CEDA: Learned Cardinality Estimation with Domain Adaptation

Zilong Wang*
Beijing Jiaotong University, China
zilongwang@bjtu.edu.cn

Qixiong Zeng*
Beijing Jiaotong University, China
zengqixiong@bjtu.edu.cn

Ning Wang#
Beijing Jiaotong University, China
nwang@bjtu.edu.cn

Haowen Lu
Beijing Jiaotong University, China
haowenlu@bjtu.edu.cn

Yue Zhang
Beijing Jiaotong University, China
yuezhang@bjtu.edu.cn

ABSTRACT

Cardinality Estimation (CE) is a fundamental but critical problem in DBMS query optimization, while deep learning techniques have made significant breakthroughs in the research of CE. However, apart from requiring sufficiently large training data to cover all possible query regions for accurate estimation, current query-driven CE methods also suffer from workload drifts. In fact, retraining or fine-tuning needs cardinality labels as ground truth and obtaining the labels through DBMS is also expensive. Therefore, we propose **CEDA**, a novel domain-adaptive CE system. **CEDA** can achieve more accurate estimations by automatically generating workloads as training data according to the data distribution in the database, and incorporating histogram information into an attention-based cardinality estimator. To solve the problem of workload drifts in real-world environments, **CEDA** adopts a domain adaptation strategy, making the model more robust and perform well on an unlabeled workload with a large difference from the feature distribution of the training set.

PVLDB Reference Format:
Zilong Wang, Qixiong Zeng, Ning Wang, Haowen Lu, and Yue Zhang. CEDA: Learned Cardinality Estimation with Domain Adaptation. PVLDB, 16(12): 3934 - 3937, 2023.
doi:10.14778/3611540.3611589

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Donny-350/CEDA.

1 INTRODUCTION

The query optimizer is an important component of a DBMS that aims to improve query execution efficiency based on cardinality estimation [4]. The errors of cardinality estimation (CE) can have a significant impact on query performance. Traditional DBMSs like PostgreSQL use histograms for CE while making the assumption of independence across columns. Especially when estimating the cardinality of multi-table join queries, the q-error can be more significant. With the application of machine learning in the DB community, researchers have begun to explore how to use machine learning to solve the problems of CE errors.

Existing deep learning-based CE methods can be classified as data-driven and query-driven. Generally speaking, query-driven models are more efficient than data-driven models in terms of inference time [6]. Data-driven model Naru [9] treats CE as a density estimation problem and learns the joint data distribution of each data point. However, it suffers from long training and inferencing time, as well as high resource consumption when processing complex queries. Query-driven models LW-NN/XGB [2] and MSCN [3] can compete with traditional DBMSs in terms of inference time. Especially, MSCN is effective for cardinality estimation of multi-table join while LW-NN/XGB can only process the query with single table. However, all the query-driven models suffer from workload drifts, i.e., they tend to perform poorly when the feature distribution of the actual workload significantly differs from that of the training workload. To improve their generalization ability, these models need to be fine-tuned or retrained, which will consume significant computing resources and time. Furthermore, the training workload is critical for query-driven models. For accurate estimation, a large training dataset covering as many query regions as possible is helpful. In practical DBMS, the overhead of query optimization should be small, so improving the generalization ability of the CE model to obtain accurate CE results on unlabeled workload is a key challenge.

To address the aforementioned challenges, we propose to tackle the problem from two aspects. 1) Workload generation, for generating queries, which can cover nearly all possible query regions in the database. It is able to solve the cold-start problem while enabling the trained model to have good adaptability. Existing workload generation methods, such as random generation [5] and template-based generation [1], have not taken the data distribution of database into account, so it is difficult to cover all possible query regions. In addition, for different databases will have different workloads depending on user needs, it is important to customize workload for specific user demands. 2) Domain adaptation, a transfer learning technique, which can improve the generalization performance of the model when the training set and the real workload come from different distributions. Although Domain adaptation has achieved great success in entity resolution [7], we are the first to explore domain adaptation in CE for addressing workload drifts.

In this paper, we propose a query-driven CE approach based on database metadata and domain adaptation. We also developed an operational system, CEDA, and integrated the entire lifecycle of our CE approach into an end-to-end system. Specifically, it includes generating workload to solve the cold start problem, training

* The first two authors contribute equally to this paper.
# Ning Wang is the corresponding auther.

Proceedings of the VLDB Endowment, Vol. 16, No. 12 ISSN 2150-8097.
doi:10.14778/3611540.3611589

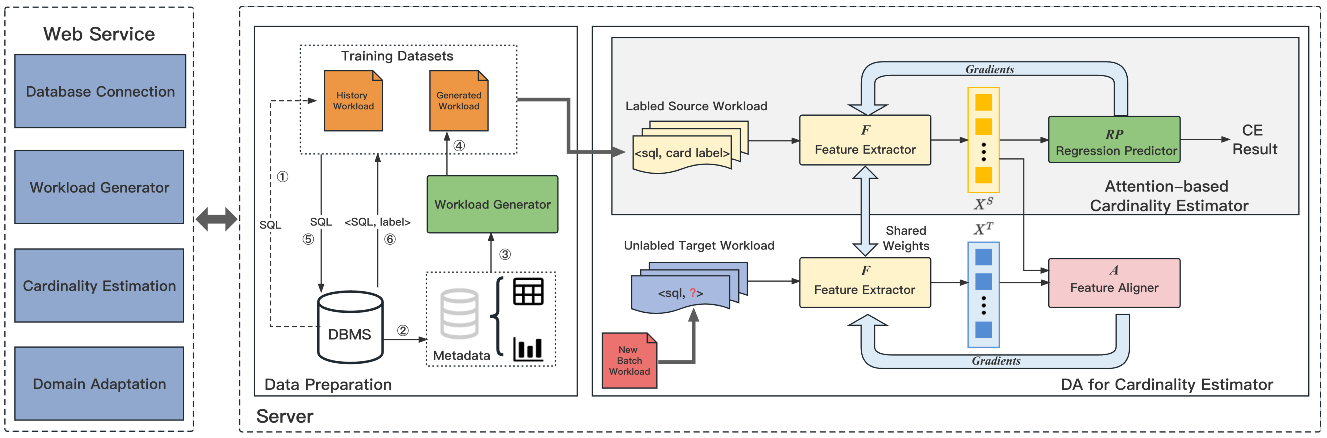

Figure 1: The Architecture of CEDA.

lightweight CE model, and domain adaptation for workload drifts. CEDA has the following new features:

(1) CEDA has a customizable workload generation module for overcoming the cold-start problem and enhancing the adaptability of the model for complex scenarios.
(2) CEDA is equipped with an attention-based cardinality estimator, which can produce accurate estimation results by incorporating histogram information in the database.
(3) CEDA adopts the domain adaptation strategy, making the model more robust and perform well on a real workload with a large difference from the feature distribution of the training set.

2 SYSTEM OVERVIEW

Our cardinality estimation system CEDA is composed of two main components: web service and server, as shown in Figure 1.

Web Service. To assist users in performing CE for SQL queries, we provide four main functions in the web interface: 1) Connecting to the database; 2) Generating customizable workload according to the data feature distribution in the database; 3) Performing accurate cardinality estimation and providing visualized results; 4) Using domain adaptation techniques to process workload drifts.

Server. An end-to-end cardinality estimation pipeline is supported by three modules:

(1) Data Preparation module is utilized to overcome the cold-start problem for the query-driven CE model. In general, the CE model can use historical database workload as the initial training corpus. To accommodate complex and changeable database business scenarios, the workload generator of CEDA can be customized to generate workload based on user specifications. Specifically, it can generate a large number of queries according to the data features obtained from the metadata (e.g., tablename, attname, histogram_bounds, most_common_vals, etc.) to form the final corpus for CE model training.

(2) Attention-based Cardinality Estimator module is designed to provide accurate cardinality estimation. We use materialized samples and histograms to mitigate the impact of data skewness on training and improve the generalization of our model. The introduction of the attention mechanism enables a more accurate representation of the predicate vector. For reducing time costs, the cardinality estimator module can be trained offline.

(3) DA module is aimed at addressing the issue of workload drifts. The performance of the CE model is often compromised when there is a significant difference in the feature distributions between the source and target workload. To address this problem, we adopt a domain adaptation strategy, introducing a domain classifier and employing the Gradient Reversal Layer (GRL) to enable adversarial training between the feature extractor and domain classifier, which helps to align the feature distributions of the source and target workload.

3 DEMONSTRATION OVERVIEW

In this section, we will demonstrate how to easily use the CEDA system for an end-to-end cardinality estimation. Figure 2 displays a screenshot of the CEDA front-end system. Users can navigate to the corresponding functional page by clicking on the left-hand navigation bar.

Step 1 (Database Connection.) Users need to input the corresponding database connection information on the "Database Detail" page (see Fig.2-1) and click the "Connect" button, and then the detailed information of the database will be displayed on the page.

Step 2 (Workload Generation.) When there is no enough historical workload available for training, CEDA can generate a comprehensive workload according to user needs on the "Workload Generator" page (see Fig. 2-2). To avoid an explosion in the number of predicate and join combinations during SQL generation, users can select the number of SQL statements, the maximum number of joins, and the maximum number of predicates in the "Basic Settings" section of the page. Customized services are also provided for different user requirements under different business scenarios by selecting Table Name and Column Name in the "More Settings" section. If the user does not specify anything, the workload will be generated based on metadata in database. After clicking the "Generate" button, the workload will be generated and shown in

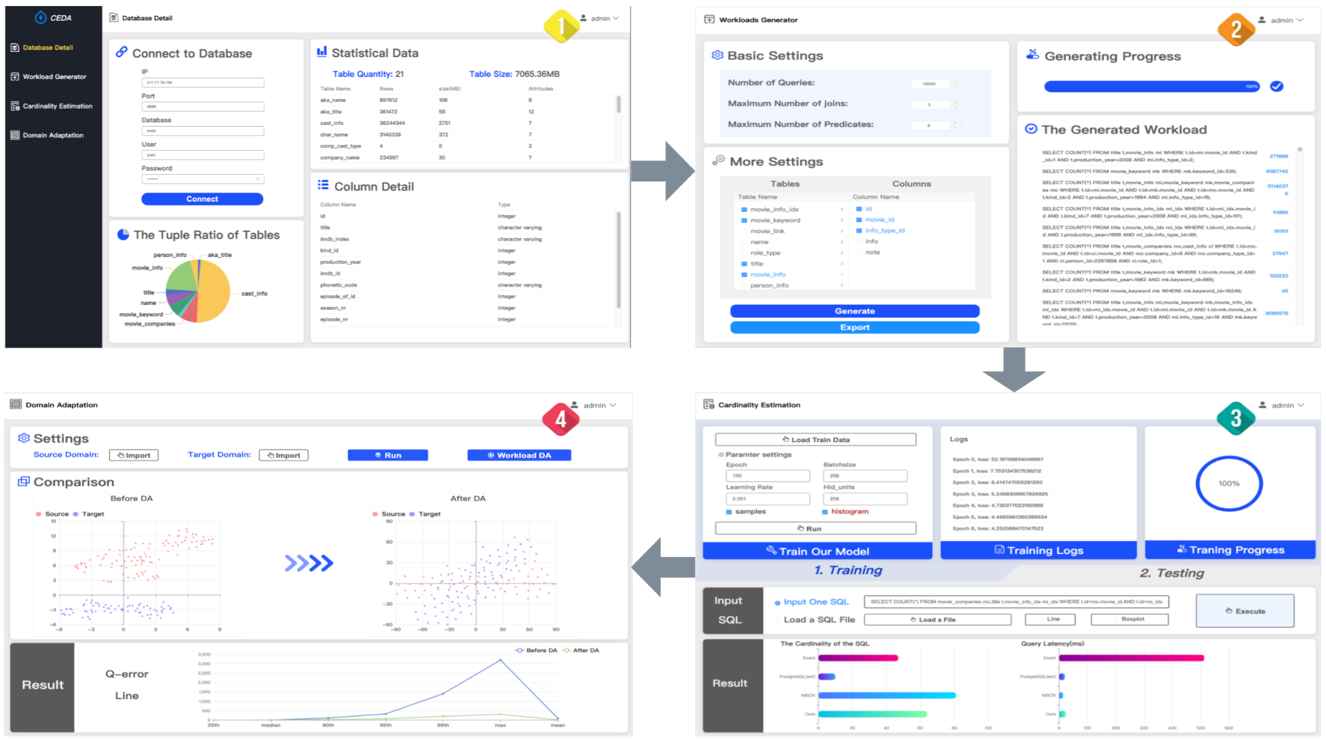

Figure 2: A Screenshot of CEDA.

"The Generated Workload" list, while the generating progress is displayed in the upper-right corner of the page. Users can also click the "Export" button to export the workload generated.

Step 3 (Model Training & Cardinality Estimation.) Users can import training data by clicking the "Load Train Data" button on the "Cardinality Estimation" page (see Fig.2-3). In addition to importing the workload generated in Step 2, users can also import their own workload. After setting epoch, batch size, learning rate, and hidden units, users can control whether to add the materialized samples and histogram information for model training by selecting the "samples" and "histogram" checkboxes. After clicking the "Run" button, the training logs and progress will be displayed in real-time on the "Training Logs" and "Training Progress" sections. Once the training is completed, users can view the visualized results of cardinality estimation for single or batch SQL statements. Users can input an SQL statement in the text box and click the "Execute" button to view the true value and estimated cardinalities by various methods in the "Result" section, as well as the estimation time by each method. In addition, users can import SQL files and view the q-error line chart or q-error box plot by selecting the corresponding options.

Step 4 (Domain Adaptation.) Users can perform domain adaptation on two workloads with significant differences in feature distributions on the "Domain Adaptation" page (see Fig.2-4). First, users import the source and target workload at the "Source Domain" and "Target Domain" respectively. To show the changes in feature distributions before and after alignment, t-SNE [8] is used to map the features of the source and target domain to a two-dimensional space and is displayed in the "Comparison" section. By clicking the "Run" button, users can observe the difference between feature distributions of the source and target domain before domain adaptation. Due to workload drifts, the q-error of cardinality estimation may increase significantly when the model trained on the source domain is directly applied to the target domain. The feature distributions will be aligned by clicking the "Workload DA" button and displayed in the "After DA" section. Meanwhile, users can see a noticeable decrease in overall q-error after performing workload domain adaptation.

4 CORE TECHNIQUES

In this section, we will introduce the key techniques of DA for attention-based cardinality estimator. As shown in Figure 3, it consists of three key components: Feature Extractor, Cardinality Predictor and Domain Classifier.

Feature Extractor(*F*). Feature Extractor is used to extract features and learn representations for SQL queries, which consists of MLP and attention mechanism. Firstly, we divide the SQL query into three parts: table names, join relations, and predicates. Then, the table names and join relations are encoded independently using a two-layer MLP with ReLU as the activation function to get their feature representations, respectively. To mitigate the impact of data skewness on training, histogram information is introduced, and the histogram vector is obtained by matching the relevant columns of the predicate with the boundary value set of their corresponding

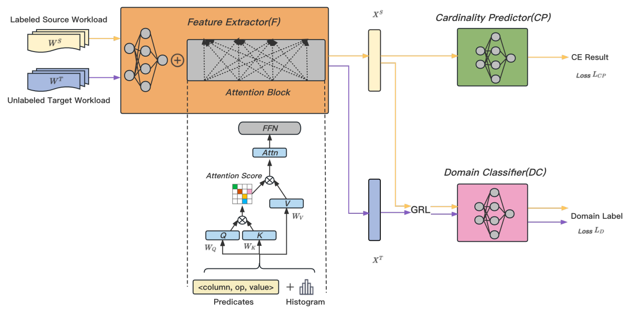

Figure 3: The Structure of DA for Attention-based Cardinality Estimator.

histograms. In order to improve estimation accuracy for predicates never seen before, we obtain the vector representation of the predicate by concatenating the vector of triple <column, op, value> and the vector of histogram to help the model better understand the data distribution. Furthermore, to better capture the correlation among predicates, we employ an attention mechanism layer that takes the predicate vectors as input, and computes the similarity between each predicate pair. The similarities are then converted into weight coefficients, which are used to compute the weighted sum of all predicates through a fully connected layer for a comprehensive and accurate representation of the Where clause. Finally, after concatenating the table vector, join vector and predicates vector obtained, we feed them into another fully connected layer to get the accurate vector representation $\boldsymbol{X}$ of the SQL statement.

Cardinality Predictor(*CP*). ***CP*** is a regression predictor composed of two fully connected layers, with the final layer utilizing a Sigmoid activation function. It takes the SQL feature vector $\boldsymbol{X}$ obtained from $\boldsymbol{F}$ as input and predicts a scalar value between 0 and 1, which is then denormalized to obtain the cardinality value. The q-error comparison results in Table 1 demonstrate that our model is effective in giving more accurate cardinality estimation compared with existing methods. In these comparison methods, LW-NN cannot handle the cardinality estimation of multiple tables.

Domain Classifier(*DC*). ***DC*** is a domain discriminator consisting of two fully connected layers, with the ReLU and LogSoftmax activation functions used in the middle and output layer, respectively. It takes the SQL feature vector $\boldsymbol{X}$ extracted from $\boldsymbol{F}$ as input and determines whether $\boldsymbol{X}$ belongs to the source or target domain. The output of ***DC*** can be used to compute the adaptation loss and adjust the parameters of $\boldsymbol{F}$ for domain adaptation purposes.

Adversarial-based Training. Our model employs adversarial training for domain adaptation, in which only the labeled data from the source domain is used to train $\boldsymbol{F}$ and ***CP***, without involving any labeled data from the target domain. During the training process, we use the q-error loss function L_{CP} to optimize $\boldsymbol{F}$ and ***CP***. Here, we denote the target value as y and the predicted value as $\hat{y}$.

$$L_{CP} = \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{max(y_i, \hat{y}_i)}{min(y_i, \hat{y}_i)} \tag{1}$$

Simultaneously, $\boldsymbol{F}$ and ***DC*** are trained using mixed data from the source and target domain to enable $\boldsymbol{F}$ to learn domain-invariant feature representations. In this process, we use a GRL to confuse domain labels as much as possible, making it difficult for ***DC*** to differentiate between the source and target domain. The adaptive loss function L_D is used to optimize $\boldsymbol{F}$ and ***DC***, resulting in a good performance in the target domain. The quantities N_s and N_t denote the number of samples in the source domain and target domain, respectively.

$$L_D = -\frac{1}{N_s} \sum_{i=1}^{N_s} log(DC(F(X_s^i))) - \frac{1}{N_t} \sum_{j=1}^{N_t} log(1 - DC(F(X_t^j))) \tag{2}$$

The loss function for the overall model in the adversarial training is expressed as:

$$L = L_{CP} - L_D \tag{3}$$

We conducted experiments on two workloads with significantly different feature distributions and compared the q-error before and after domain adaptation in Table 2. We can see that DA can reduce q-error and improve the accuracy of cardinality estimation results on the target domain significantly.

Table 1: Estimation errors on two Real-World datasets.

Dataset	Estimator	25th	median	90th	95th	99th	max	mean
IMDB	PostgreSQL	2.38	6.79	162.80	825.94	2983.92	3534.75	162.15
	LW-NN	-	-	-	-	-	-	-
	MSCN	1.74	3.28	34.62	174.77	670.14	673.46	35.56
	Ours	**1.43**	**3.09**	**20.00**	**57.48**	**339.82**	**469.28**	**18.94**
Forest	PostgreSQL	1.10	1.35	11.00	29.97	299.02	7719.00	22.16
	LW-NN	1.24	1.78	5.12	9.39	34.21	**186.83**	3.52
	MSCN	1.10	1.5	4.58	6.62	20.01	187.00	3.01
	Ours	**1.09**	**1.34**	**3.37**	**5.05**	**14.30**	214.0	**2.36**

Table 2: Comparison of q-error before and after Domain Adaptation.

	25th	median	90th	95th	99th	max	mean
NoDA	2.47	8.36	117.48	334.26	1406.90	3200.11	93.00
DA	**1.78**	**4.87**	**30.18**	**68.60**	**203.67**	**314.63**	**17.61**

REFERENCES

[1] Nicolas Bruno, Surajit Chaudhuri, and Dilys Thomas. 2006. Generating queries with cardinality constraints for dbms testing. *IEEE Transactions on Knowledge and Data Engineering* 18, 12 (2006), 1721–1725.

[2] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek Narasayya, and Surajit Chaudhuri. 2019. Selectivity estimation for range predicates using lightweight models. *Proceedings of the VLDB Endowment* 12, 9 (2019), 1044–1057.

[3] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. 2018. Learned cardinalities: Estimating correlated joins with deep learning. *arXiv preprint arXiv:1809.00677* (2018).

[4] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How good are query optimizers, really? *Proceedings of the VLDB Endowment* 9, 3 (2015), 204–215.

[5] Donald R Slutz. 1998. Massive stochastic testing of SQL. In *VLDB*, Vol. 98. Citeseer, 618–622.

[6] Ji Sun, Jintao Zhang, Zhaoyan Sun, Guoliang Li, and Nan Tang. 2021. Learned cardinality estimation: A design space exploration and a comparative evaluation. *Proceedings of the VLDB Endowment* 15, 1 (2021), 85–97.

[7] Jianhong Tu, Ju Fan, Nan Tang, Peng Wang, Chengliang Chai, Guoliang Li, Ruixue Fan, and Xiaoyong Du. 2022. Domain adaptation for deep entity resolution. In *Proceedings of the 2022 International Conference on Management of Data*. 443–457.

[8] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[9] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep unsupervised cardinality estimation. *arXiv preprint arXiv:1905.04278* (2019).