Received: 10 September 2021 | Revised: 18 October 2021 | Accepted: 11 November 2021 | CAAI Transactions on Intelligence Technology

DOI: 10.1049/cit2.12069

ORIGINAL RESEARCH

A BiLSTM cardinality estimator in complex database systems based on attention mechanism

Qiang Zhou[1] | Guoping Yang[2] | Haiquan Song[3] | Jin Guo[1] | Yadong Zhang[1] | Shengjie Wei[4] | Lulu Qu[2] | Louis Alberto Gutierrez[5] | Shaojie Qiao[2]

[1]School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China

[2]School of Software Engineering, Chengdu University of Information Technology, Chengdu, China

[3]School of Energy Power and Mechanical Engineering, North China Electric Power University, Baoding, China

[4]Digital Media Art, Key Laboratory of Sichuan Province, Sichuan Conservatory of Music, Chengdu, China

[5]Department of Computer Science, Rensselaer Polytechnic Institute, New York, New York, USA

Correspondence

Shaojie Qiao, School of Software Engineering, Chengdu University of Information Technology, Chengdu, China.
Email: sjqiao@cuit.edu.cn

Funding information

National Natural Science Foundation of China, Grant/Award Numbers: 61772091, 61802035, 61962006, 61962038, U1802271, U2001212, 62072311; Sichuan Science and Technology Program, Grant/Award Numbers: 2021JDJQ0021, 22ZDYF2680; CCF-Huawei Database System Innovation Research Plan, Grant/Award Number: CCF-HuaweiDBIR2020004A; Digital Media Art, Key Laboratory of Sichuan Province, Sichuan Conservatory of Music, Chengdu, China, Grant/Award Number: 21DMAKL02; Chengdu Major Science and Technology Innovation Project, Grant/Award Number: 2021-YF08-00156-GX; Chengdu Technology Innovation and Research and Development Project, Grant/Award Number: 2021-YF05-00491-SN; Natural Science Foundation of Guangxi, Grant/Award Number: 2018GXNSFDA138005; Guangdong Basic and Applied Basic Research Foundation, Grant/Award Number: 2020B1515120028; Science and Technology Innovation Seedling Project of Sichuan Province, Grant/Award Number: 2021006; College Student Innovation and Entrepreneurship Training Program of Chengdu University of Information Technology, Grant/Award Numbers: 202110621179, 202110621186

[Correction added on 29 December 2021, after first online publication: The following change need to be noted.
1. The affiliation of Shengjie Wei is updated as below:
[4]Digital Media Art, Key Laboratory of Sichuan Province, Sichuan Conservatory of Music, Chengdu, China
2. The affiliation 4 is deleted and the affiliation designators are updated.]

Abstract

An excellent cardinality estimation can make the query optimiser produce a good execution plan. Although there are some studies on cardinality estimation, the prediction results of existing cardinality estimators are inaccurate and the query efficiency cannot be guaranteed as well. In particular, they are difficult to accurately obtain the complex relationships between multiple tables in complex database systems. When dealing with complex queries, the existing cardinality estimators cannot achieve good results. In this study, a novel cardinality estimator is proposed. It uses the core techniques with the BiLSTM network structure and adds the attention mechanism. First, the columns involved in the query statements in the training set are sampled and compressed into bitmaps. Then, the Word2vec model is used to embed the word vectors about the query statements. Finally, the BiLSTM network and attention mechanism are employed to deal with word vectors. The proposed model takes into consideration not only the correlation between tables but also the processing of complex predicates. Extensive experiments and the evaluation of BiLSTM-Attention Cardinality Estimator (BACE) on the IMDB datasets are conducted. The results show that the deep learning model can significantly improve the quality of cardinality estimation, which is a vital role in query optimisation for complex databases.

KEYWORDS

attention, BiLSTM, cardinality estimation, complex database systems, query optimiser, Word2vec

1 | INTRODUCTION

The performance of a query optimiser depends on cardinality estimation. In order to select the best execution plan, the query optimiser should have a good estimation of the cardinality with high prediction accuracy as well as low execution time. However, the estimations of cardinality in regard to complex traditional relational databases usually have huge errors. To the best of our knowledge, the biggest challenge of cardinality estimation is how to obtain the relationships between multiple tables and how to deal with complex query statements in complex database systems is another difficult task.

Currently, Multi-Set Convolutional Network (MSCN) [1] is a state-of-the-art method that has a powerful query optimisation performance. It can identify the relations between multiple tables and can accurately estimate cardinality at a faster speed. However, the encoding method of MSCN is too simple to encode a specific query. If the query structure is complex and involves nesting predicates, MSCN cannot encode it. In addition, MSCN cannot process complex predicates, for example, 'LIKE %XXX%', when complex predicates are encountered.

In the past decades, machine learning has extensively been studied and even widely applied in various fields [2–4]. Recently, the database community is exploring the way of combining complex database systems with machine learning or deep learning techniques. Therefore, the existing work based on machine learning approaches to solve classic database problems is initiated, for example, parameter tuning, query optimisation, and even indexing.

Cardinality estimation is a supervision problem. The input layer is the encoding feature of the query, while the output layer is the estimated cardinality. There are many advanced methods that only consider a single table or simple query predicates. These technologies are too special and have no generalisation capability to meet all kinds of query tasks in a big data era. The existing cardinality estimation methods are not perfect since most of them aim to find a better model than the common baselines.

In order to settle the problems in the existing estimation techniques, we have made the following original contributions in this study:

- We propose a cardinal estimator based on **B**iLSTM-**A**ttention [5], which can obtain the relations between multiple tables, semantic information of query and deal with complex predicates. For the input query, our model has a powerful generalisation ability and can handle all kinds of queries. With the combination of **B**iLSTM and **A**ttention mechanism, the model can fully obtain the semantic information of queries.
- We use Word2wec [6] as an embedding model for complex string predicates so that these complex predicates can be represented by vectors.
- We propose a sampling strategy. In order to make the model learn data distribution, we sample the data (samples) in the database and compress the samples into bitmaps. We integrate the vector obtained by BiLSTM attention with the sample bitmap to make the final vector contain all the information of the query.

2 | RELATED WORK

Recently, the researchers have applied the deep learning methods [7, 8] in query optimisation. For the selection of join order, this kind of problem needs to constantly interact with the intermediate results of the current query. Reinforcement learning [9–11] is used to find the best query plan. Currently, a lot of work uses join enumeration to find the best join order, and these joins often have complex relationships. For cardinality estimation, we need to use supervised learning to estimate cardinality. Cardinality estimation is a vital role in query optimisation, which can degrade the performance of queries. The first work of learning-based cardinality estimation [12] is to classify queries according to join conditions, predicates and etc. Then, the model is trained by the value of predicates, however, the model is invalid for training unknown structured queries. The existing cardinality estimation method either constructs a supervised model which can map the features of query Q to its cardinality label [1, 13], or learns the unsupervised model of P_T (P_T is the joint probability distribution of table T) to support the calculation of the probability of any query Q on table T [14–16]. The cardinality estimation method based on deep learning greatly improves the accuracy of estimation, but it usually leads to a lot of time and computing resources consumption. Liu et al. [17] proposed the ML method to estimate cardinality, however, they do not focus on the joins of tables, which is the important challenge in cardinality estimation. Currently, MSCN is the advanced cardinality estimation method, but MSCN only pays attention to the estimation results, and the data of training set is too simple to deal with complex predicates.

Our method is based on the estimation of sampling results. The cardinality or bitmap obtained from the sample is embedded into the training signal. Most methods sample each table and try to estimate the cardinality of multi-table joins [18–21]. Although these methods exhibit a good performance in single-table estimation, they are not good at capturing the relationship between multiple tables so that the results are not good. 0-tuple problems often occur. Although, Muller et al. [22] reduce the 0-tuple problem of join predicates (the calculation cost is very high), but it is still unable to solve the problem of single predicate giving zero results. Our proposed estimation method in the case of 0-tuple can cope with the over-sampling problem even including the estimation of real correlated samples, which still cannot deal with the case of 0-tuple. Qiao et al. [23] propose a convolution estimation method for vertical scanning (called VSCNN). This method uses three different convolution kernels with different widths to scan the condition clauses from top to bottom to obtain the information in the query, but it is not compatible with discrete long string predicates because it cannot recognise the sequence of words in time well. In [24], a series of new pre-training techniques, such as Word2vec [6] are surveyed. Our work is based on the BiLSTM model [25], which uses NLP tools, vocabulary resources and LSTM units to achieve the same results as that of the advanced techniques.

3 | BACKGROUND KNOWLEDGE AND PRELIMINARIES

In this section, we briefly introduce the model structure and workflow and analyse the internal structure and principle of these technologies. In the proposed BACE model, we will use Word2vec and LSTM as a preparation.

3.1 | Word2vec model

The NLP team has developed two popular Word2vec [6] models, namely *Skip-gram* and *CBOW*, which perform unsupervised learning on word sequences with a loss function, which is similar to the classification cross entropy of the *Softmax* function. For Word2vec, the traditional *Softmax* is not applicable. There are two methods below to achieve the classification effect of *Softmax* in the Word2vec task: 1) negative sampling [26] and 2) hierarchical softmax [27]. In this study, we use negative sampling because it obtains results quickly.

CBOW and *Skip − gram* have simple structures of three layers, which are shallow neural network models. In both methods, each pair contains two dense vector representations: one is a center word matrix denoted by W in Figure 1, and the other is a context word matrix represented by W' in Figure 1. Word2vec does not pay attention to the output result; it only pays attention to the weight matrix generated in the intermediate process, and the weight matrix contains the vector representation of the word.

Table 1 shows the five most important Word2vec hyperparameters: They include 1) the dimensions of vector d; 2) the threshold of sliding window size L; 3) negative sampling index α; 4) the number of negative samples used by each pair N; and 5) the initial learning rate β. In addition, word2vec needs a down-sampling threshold and a minimum pair appearance threshold. We set the down-sampling rate to 10^{-5} and the minimum entity occurrence rate to 1.

3.2 | LSTM model

The *Long short-term memory* (*LSTM*) network is an improved version of recurrent neural network (*RNN*), which has no vanishing gradients or exploding gradients. In particular, *LSTM* can get long-term context relations of dependencies. We use a bidirectional *LSTM* (*BiLSTM*) to capture context sequence information. In the embedding layer, each word embedding vector is fed to the *LSTM* unit, and each *LSTM* unit is composed of hidden units with the size of h. For *BiLSTM*, we concatenate each output of *LSTM*, and we obtain a vector representation, which has the length of $2h$. *LSTM* treats the input sequence as a pair ($embed^{<i>}$, $y^{<i>}$). For each pair ($embed^{<i>}$, $y^{<i>}$) and each time step t, *LSTM* saves memory vector $mr^{<t>}$ and hidden vector $hd^{<t>}$. These vectors are used to update the cell state and generate the target output $y^{<i>}$ which is referred to the former state. The details of processing at time slice t are shown below:

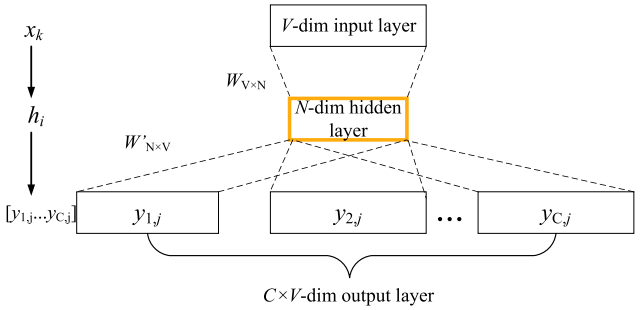

FIGURE 1 The architecture of Word2vec

TABLE 1 Word2vec hyperparameters

Parameter	Value	Description
D	100	The number of parameters
L	5	Threshold of distance between two marked words
A	3/4	Negative sampling index
N	5	Number of negative samples
B	0.025	Learning rate

$$u_g = \sigma(W_u * hd^{<t-1>} + I_u) \quad (1)$$

$$f_g = \sigma(W_f * hd^{<t-1>} + I_f) \quad (2)$$

$$o_g = \sigma(W_o * hd^{<t-1>} + I_o) \quad (3)$$

$$c_g = tanh(W_c * hd^{<t-1>} + I_c) \quad (4)$$

$$m^{<t>} = f_g \odot mr^{<t-1>} + u_g \odot c_g \quad (5)$$

$$hd^{<t>} = tanh(o_g \odot mr^{<t>}) \quad (6)$$

where σ is the sigmoid function, W_u, W_f, W_o, W_c and I_u, I_f, I_o, I_c are the weight matrix and projection matrix of the cyclic unit. The computational gates of LSTM cells u_g, f_g, o_g and c_g play a vital role in getting key properties of the computational vectors. They can be saved in the memory vector mr if needed. The forgetting gate f_g decides to discard some information from the former memory vector $mr^{<i-1>}$. The updated information is saved as the new memory vector $mr^{<i>}$ by the update gate c_g, while the update gate uses the former memory vector $mr^{<i-1>}$ and the input gate u_g. Finally, the o_g output gate probes the information from the new memory vector $mr^{<i>}$ to the hidden vector $hd^{<i>}$.

4 | BiLSTM-ATTENTION CARDINALITY ESTIMATOR

In this section, we will discuss the *BACE* model in detail. As shown in Figure 2, this model consists of seven steps:

(1) Input layer: input the query statement into the training model;

(2) Bitmap layer: input the sample bitmap corresponding to each query statement, and average the vectors passing through the multi-layer perceptions (MLP);
(3) Embedding layer: mapping query statements into word vectors;
(4) *LSTM* layer: transform the word vector from Step (3) into the high-level feature vector through bidirectional LSTM;
(5) Attention layer: generate weight matrix, while the word embedding of each time step are multiplied by the weight matrix. The results are the sentence level feature matrix;
(6) Output layer: combine the vectors generated by Step (2) and Step (5) and input them into the MLPs to obtain the final results.
(7) Loss function: measure the quality of the cardinality estimator, and perform gradient descent according to the loss value to obtain the optimal solution.

In order to show the importance of the proposed model, we give a running example as shown in Figure 3. The optimiser mainly includes three components: cardinality estimator, cost model and connection order selection. The component in red rectangle is the proposed model, which can make a more accurate cardinality estimation of SQLs and then other parts use this estimation to generate the optimal execution plan tree.

4.1 | Bitmap layer

The basic idea of the proposed model is to get the rich training data, and the method is to obtain the information from the real-world-based table samples. In this study, we sample each table. Here, we set the number of samples to 1000. These samples are used to measure the hitting probability of the predicates of queries. The bitmap is used to store sample information. The sample bitmap is a <0–1> vector with a fixed size, while each bit represents whether the sample matches the predicate. If the record hits the predicate, the corresponding bit is 1; otherwise, it is 0.

We represent a query as $q \in Q$, and B_q represents the bitmap contained in a query q. If a query q contains n predicate nodes of type *'Seq Scan'*, n bitmaps will be generated, which are represented by B_q^1, B_q^2,…, B_q^n, respectively. We input the bitmap samples corresponding to each predicate node of type *'Seq Scan'* into n different *MLPs*, and then we can obtain n different vectors denoted by $MLP(B_q^i)$. Finally, we compute the weighted average of $MLP(B_q^i)$. The bitmap layer can be expressed by Equation (7):

$$w_B = \frac{1}{n}\sum_{i=1}^{n} MLP(B_q^i) \tag{7}$$

4.2 | Embedding layer

Given a query $Q = (x_1, x_2,\ldots, x_T)$ which is composed of T keywords or symbols, and each word x_i is transformed into a context relevant word vector $embed_i$. For each word x_i in Q, we first calculate the embedding matrix $W^E \in \mathbb{R}^{d_w|V|}$. V represents a fixed word list, while d_w represents the dimension of the word vector. The weight W^E is a parameter matrix based on learning, and d_w is a hyperparameter. We encode a word x_i into a word embedded $embed_i$, which is the product of matrix vectors as shown in Equation (8).

$$embed_i = W^E * v^i \tag{8}$$

where v^i is a vector of length $|V|$, and its value equals 1 at index $embed_i$, and its value is 0 at all other locations. Then, the SQL is transformed into the query embedding layer $embs = (embed_1, embed_2,\ldots, embed_T)$.

4.3 | BiLSTM layer

Hochreiter and Schmidhuber [28] first prove that the LSTM cell can solve gradient vanishing. Introducing an adaptive gating mechanism is a basic idea. It determines the degree where LSTM cells remain in the former state and stores the retrieved features of the current input vector. In this study, we chose the variant of the LSTM proposed by Graves et al. [29].

Generally speaking, a recurrent neural network based on LSTM is composed of three components: an input gate i_t that contains weight matrices W_{x_i}, W_{hd_i}, W_{cl_i} and b_i; a forgetting gate f_t that contains weight matrices W_{x_f}, W_{hd_f}, W_{cl_f} and b_f; one output gate o_t that contains weight matrix W_{x_o}, W_{hd_o}, W_{cl_o}, and b_o. We use the current input x_i, the state of hd_{i-1} generated in the previous step, and the current state of cell cl_{i-1} to determine whether to input, forget the previously stored memory, and then output the generated state.

$$i_t = \sigma(W_{x_i} * x_t + W_{hd_i} * hd_{t-1} + W_{x_i} * x_t + W_{cl_i} * c_{t-1} + b_i) \tag{9}$$

$$f_t = \sigma(W_{x_f} * x_t + W_{hd_f} * h_{t-1} + W_{cl_f} * cl_{t-1} + b_f) \tag{10}$$

$$g_t = tanh(W_{hd_c} * hd_{t-1} + W_{x_{cl}} * x_t + W_{cl_{cl}} * cl_{t-1} + b_{cl}) \tag{11}$$

$$cl_t = f_t * cl_{t-1} + i_t * g_t \tag{12}$$

$$o_t = \sigma(W_{hd_o} * hd_{t-1} + W_{x_o} * x_t + W_{cl_o} * cl_t + b_o) \tag{13}$$

$$hd_t = o_t * tanh(cl_t) \tag{14}$$

However, it is beneficial for understanding the context because it can model the sequence of queries. Standard *LSTM* networks process sequences in the chronological order, ignoring subsequent contexts. The bidirectional *LSTM* network extends the unidirectional *LSTM* network by introducing a second layer, in which the information in the hidden connections flow in a reverse chronological order. Therefore, the model can utilise the previous and subsequent information.

In this study, we use *BiLSTM*. As shown in Figure 2, the network consists of two sub-networks which work for left

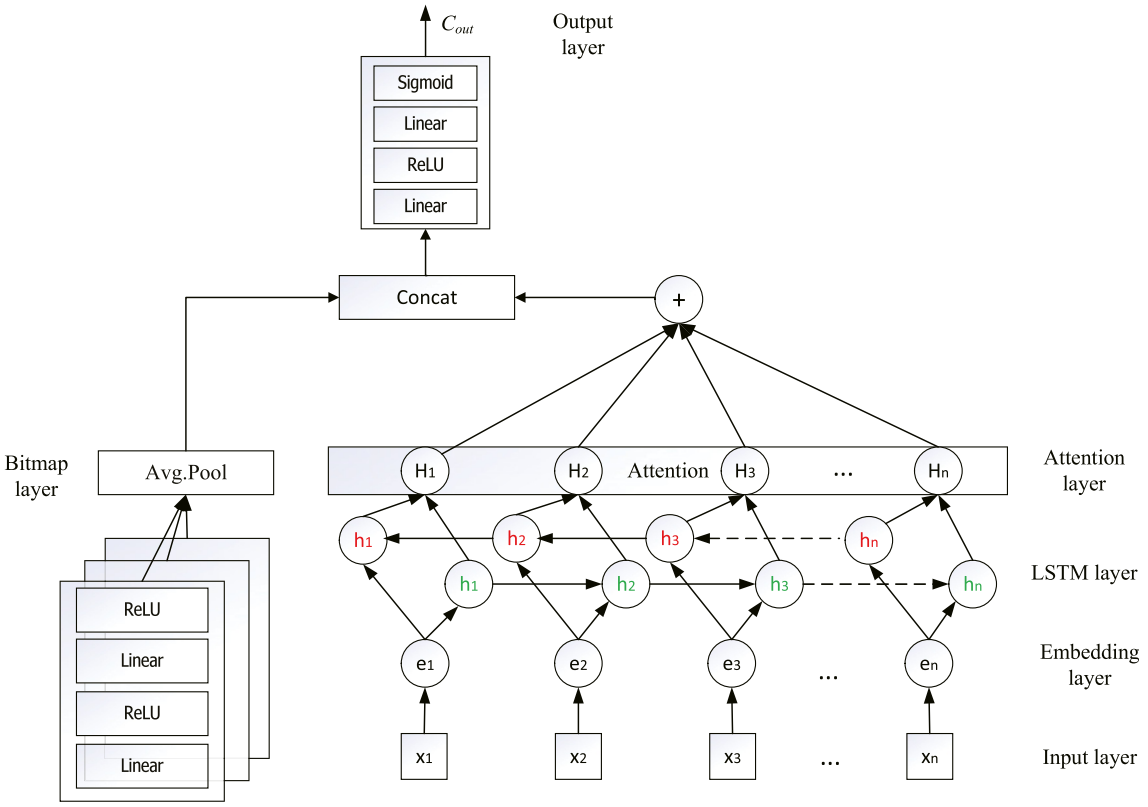

FIGURE 2 The working mechanism of BiLSTM-Attention cardinality estimator

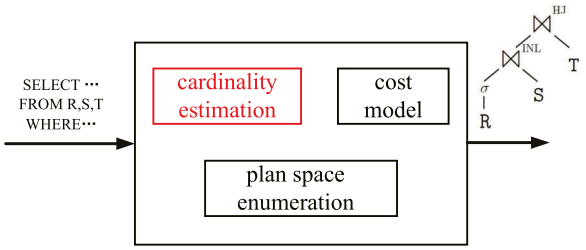

FIGURE 3 The running example of BiLSTM-Attention Cardinality Estimator (BACE)

sequence context and right sequence context, forward and backward, respectively. The output of the ith word is depicted by the following Equation (15).

$$h_i = [\overrightarrow{h_i} \oplus \overleftarrow{h_i}] \tag{15}$$

4.4 | Attention layer

In this section, we propose an attention mechanism for the cardinality estimator. Because the information of the *WHERE* clauses is difficult to extract, we add an attention layer on the basis of BiLSTM. As shown in Figure 2, the attention layer follows the LSTM layer. As an important component, the attention layer can perform a global analysis on the information processed by BiLSTM and automatically find the part with a high attention score in the *WHERE* clauses. Let H be a matrix composed of output vectors $[h_1, h_2, \ldots, h_T]$, where T is the length of the query, and the representation of a query r consists of the weighting sum of the output vectors.

$$M = tanh(H) \tag{16}$$

$$\alpha = softmax(w^T * M) \tag{17}$$

$$r = H * \alpha^T \tag{18}$$

where $H \in \mathbb{R}^{d^w} \times T$, d^w is the number of dimensions w.r.t. the word vector, W is the training parameter vector, and w^t is the transpose. We obtain the final query representation for estimation from Equation (19).

$$h^* = tanh(r) \tag{19}$$

4.5 | Output layer

Before inputting into this layer, as shown in Equation (20), we need to merge the vector w_B from *Bitmap Layer* with the vector h^* from *Attention Layer* to obtain a new vector C_{input}. The vector contains not only the sample information of the query node but also the semantic information of the query. We put the final input vector C_{input} into a new *MLP* and obtain the final output C_{out}, as shown in Equation (21).

$$C_{input} = Concat(w_B, h^*) \tag{20}$$

$$C_{out} = MLP_{out}(C_{input}) \tag{21}$$

All *MLPs* are a double layer structure of fully connected network. The other *MLPs* use the double layer *ReLU* activation function, except for the output layer. For the output layer *MLP*, we use the *Sigmoid* activation function in the last layer of the output layer, and we need a scalar, which ranges in [0,1] because the cardinality is a scalar. We can use the inverse process of normalisation to restore C_{out} to real cardinality. In hidden layers, we use the *ReLU* activation function. Because they have a strong empirical performance and quick convergence speed.

4.6 | Loss function

We use *q*-error as an evaluation indicator by defining in Equation (22) and the loss function given in Equation (23). *q*-error can measure the relative gap between the estimated value and the true value. In Equation (22) and Equation (23), C_{out} represents the estimated value and C_{label} represents the true value.

$$q-error = \max\left(\frac{C_{out}}{C_{labe}}, \frac{C_{label}}{C_{out}}\right) \tag{22}$$

$$Loss = \log \sum_{i=0}^{n} \max\left(\frac{C_{out}}{C_{labe}}, \frac{C_{label}}{C_{out}}\right) \tag{23}$$

5 | EXPERIMENTS

5.1 | Experimental setup and datasets

We use two different query workloads: 1) the Synthetic workload is generated by the same query generator as the training data (using a different random seed), which contains 5000 unique queries (conjunctions), equality predicates and range predicates. These predicates are on non-key columns, which have zero to two joins; 2) JOB-Light, the workload benchmark (JOB) [30] derived from JOB contains 70 of the original 113 queries. However, JOB-Light does not contain any string predicates, only one to four join queries. The three data sets and training data SQL used in our experiment are generated by using the intertable relationship graph of IMDB, including primary/foreign key relationship and one to many or many to one. Then, we use the SQL analysis tool of a relational database to analyse the cardinality of each SQL. TPC-H is an excellent benchmark, but it is not a real dataset in the real world and cannot reflect the specific relationship between some data. We use IMDB-based datasets, which can make our model more inclined to the real environment rather than some synthetic environment.

The most popular complex relational databases are used in comparison experiments, that is, Oracle 10g, PostgreSQL 9.5, MySQL 8.0 and SQL Server 2016. Industrial database systems refer to the above four relational databases. Of course, this model can also be applied to other relational database optimisers, such as DB2. The complexity of database systems lies in their query optimisers. The traditional database optimisers are based on statistics or sampling, combined with some complex algorithms for query optimisation. However, the traditional database statistical method is very effective for a single table, but it cannot capture the relationship when multiple tables are connected. Therefore, it is necessary to apply the learning-based method to database systems. In this research study, in order to show that our model is effective, we used several advanced cardinality estimation methods for comparison. Black-Box Approach (BBA) [12] can make an accurate estimate of the previously learned query. It divides the query structure to estimate the cardinality. However, if this method is used to identify unseen or complex structures, the effect is very poor. IBJS [31] relies too much on the index structure and is data-driven. If the indexing scheme is bad or there is no index at all and the quality of the sample data is not good, the estimated quality will be bad. The above methods are all traditional methods. It can be seen that they are all estimates based on statistical data, relying too much on data, and making independent assumptions. The learning-based cardinality estimation scheme does not need to make independent assumptions, and it can automatically learn the relationship between data and queries. VSCNN [23] uses three different convolution kernels to scan the *WHERE* clauses vertically. Although VSCNN [23] can extract local semantic information, it cannot control the time sequence. We use 90% of the query statements in the dataset as the training set and the remaining statements as the validation set.

We use PyTorch [32] as the basic framework for deep learning, and the model is trained on CUDA [33]. All experiments are conducted on a machine with AMD CPU R7-5800X, 32GB Memory, 64G SSD, GeForce RTX 3090 with 24GB and CentOS 6.5 operating system.

5.2 | Regularisation

The dropout technology was proposed by Hinton et al. [34]. It can reduce the probability of over-fitting, mainly because it allows neurons to not be overly dependent on a certain data. The target is achieved by discarding some neurons immediately during forward propagation. We use the dropout technology in the embedding layer, BiLSTM layer and output layer. For the update of new data, L_1-norm is greatly affected, while L_2-norm is slightly affected. L_1-norm tends to make coefficients sparse, but L_2-norm is the opposite. In a word, L_2-norm is more steady than L_1-norm, and this is the reason why we choose L_2-norm.

5.3 | Experimental results

Table 2 shows the *q*-error of our model and traditional relational databases on the *Synthetic* workload, and we also compare the *q*-error of BBA [12] and IBJS [31] with the proposed method. In order to show the importance of the

TABLE 2 *q*-error of cardinality estimators on synthetic workloads

Synthetic	Mid	90th	95th	99th	Max	Mean
PostgreSQL	1.68	9.61	24.0	453	369 911	149
MySQL	2.17	21.7	49.6	623	460 095	349
Oracle	1.98	12.3	41.9	463	544 995	369
SQL server	2.16	16.9	51.9	291	513 011	369
BBA	2.33	20.01	48.3	544	484 266	303
IBJS	1.11	10.26	37.1	289	294 920	128
VSCNN	1.32	4.21	7.49	33.95	1372	3.12
BACE (NoSample)	1.31	5.96	15.2	111.4	2740	11.9
BACE (NoAtt)	1.23	4.52	9.2	35.4	540	5.09
BACE	1.22	4.16	7.41	27.1	340	3.01

TABLE 3 *q*-error of cardinality estimators on job-light workloads

JOB-light	Mid	90th	95th	99th	Max	Mean
PostgreSQL	7.83	165	1112	1912	3507	169
MySQL	9.65	313	685	2256	2498	149
Oracle	8.32	374	966	2761	3361	157
SQL server	9.13	354	703	2550	3391	151
BBA	8.03	157.2	621.1	2311	3006	151
IBJS	1.59	157	3188	14 309	15 715	590
VSCNN	3.82	79.4	362.5	965.1	1118.2	60.1
BACE (NoSample)	5.82	99.56	709.2	1103	1726	84.9
BACE (NoAtt)	4.23	61.52	351.2	991.4	1024	51.3
BACE	3.35	53.16	342.41	961	973	42.02

techniques of sampling and attention, we also take these two factors as variables in experiments. According to Table 2, IBJS [31] has the best median estimation. BACE completely defeats relational databases. BACE wins other methods in the other percentile of *q*-error. Although IBJS performs well on the median *q*-error, it relies too much on the index structure and is data-driven. If the indexing scheme is bad or there is no index at all and the quality of the sample data is not good (0-tuple problem), the estimated quality will be bad. If there is no proper index or even no index, its performance will become very poor. The mean *q*-error of our model is 42.2 times smaller than that of IBJS [31]. All *q*-error measurements of BACE are better than that of VSCNN [23] on the *Synthetic* workload. Because the *Synthetic* workload is simple and BACE has an attention mechanism that can consider abstract information more efficiently, which consider the words in the whole *WHERE* clauses all at once. From the mean value in the table, we can see that sampling and attention have a profound impact on the model, in which the sampling method is more important than attention. The median, 90th, 95th and 99th in the experimental part are percentiles. Referring to the definition of box plot, there are several data that are sorted from small to large. These four numbers represent the data in the corresponding percentile, respectively. Comparing these four numbers, we can observe the distribution of cardinality.

In order to test the generalisation of the model, we use the *JOB-Light* workload with more joins. Table 3 shows the *q*-error of our model compared with that of other competitors on the *JOB-Light* workload. IBJS [31] has the best median estimation. BACE beat other models on other *q*-errors. The mean *q*-error of BACE is 14.1 times smaller than that of IBJS. Because our model has memory structure, it can deal with multiple joins of tables and complex predicates of the workload. All *q*-error measurements of BACE are better than that of VSCNN [23] on the *JOB-Light* workload. VSCNN [23] uses three different convolution kernels to extract information from the *WHERE* clauses, but it is difficult to handle the order information of the words in the *WHERE* clauses. However, BACE uses the BiLSTM network structure, which can take the order of words from two directions into consideration.

TABLE 4 *q*-error of cardinality estimators on JOB workloads with strings

JOB-String	Mid	90th	95th	99th	Max	Mean
PostgreSQL	183	8313	34 213	106 010	670 010	10 426
MySQL	104	28 147	213 471	1630 619	2487 611	60 229
Oracle	119	55 446	179 106	697 790	927 648	34 493
SQL server	174	60 432	231 045	552 190	432 609	52 700
MSCN	NAN	NAN	NAN	NAN	NAN	NAN
MSCN (String)	16.6	87.4	207.3	792.7	851.1	73.5
VSCNN	15.2	89.1	199.5	760.2	803.7	69.6
BACE	17.2	92.1	191.5	732.2	783.7	59.1

Table 4 shows the performance of our model on the *JOB* workload. The experimental results show that the *q*-error of the traditional database system is very large, even hundreds of thousands. MSCN [1] cannot handle string type of values and complex predicates because it has no related natural language processing structure. We use 'NAN' in the table to represent the *q*-error of MSCN [1]. We extend MSCN [1] by using a *skip-gram* model to embed string type of values, which improves its generalisation ability. BACE works well to have good estimations in all experiments but except for the median and 95th. MSCN (string) represent that MSCN [1] can handle strings by improving, but the *q*-error is higher than that of proposed model. BACE can handle strings well because it can represent sparse strings in the vector form and capture the complex semantic information in SQL statements. In addition to mid and 90th, other *q*-errors of BACE are better than those of VSCNN. Because the *JOB* workload with strings is very complex, BACE's powerful ability to extract information may

cause the over-fitting problem. In most cases, BiLSTM and the attention mechanism bring benefits to BACE, which can extract the information of the *WHERE* clause more accurately.

Figure 4 shows the loss of training the models. We compare the proposed models without sampling and without attention mechanism. We can see that the curve of the model without only sampling (NoSample) declines slowly, accompanied by big fluctuations. The model without only attention mechanism (NoAtt) has faster convergence speed and less fluctuation. The model with both sampling and attention has the fastest curve decline and the fastest convergence speed.

Figure 5 shows the prediction time of different models. Our model predicts that the average time required for cardinality estimations is 0.41 ms. We show the comparison of prediction time of each model in the training set and validation set. Our model takes more time than MSCN [1] and traditional databases but less time than tree structure estimator (i.e. TLSTM [35] and TPOOL [35]). The tree structure model can deal with complex predicates easily. The prediction time of the validation set of TLSTM is 1.52 times higher than that of BACE, and the prediction time of the validation set of TPOOL is similar to that of BACE. This is because the memory unit is used in our model, and the prediction time will be higher. Similarly, the more complex tree memory units used by TLSTM and TPOOL will bring more time consumption. We found that the evaluation time of BACE is shorter than the training time and verification time, but the evaluation time decreases little compared with that of MSCN [1], PostgreSQL and TLSTM.

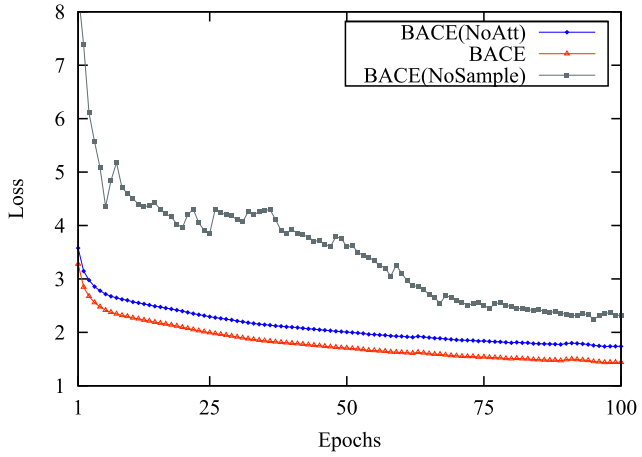

FIGURE 4 Loss comparison of cardinality estimators under different conditions

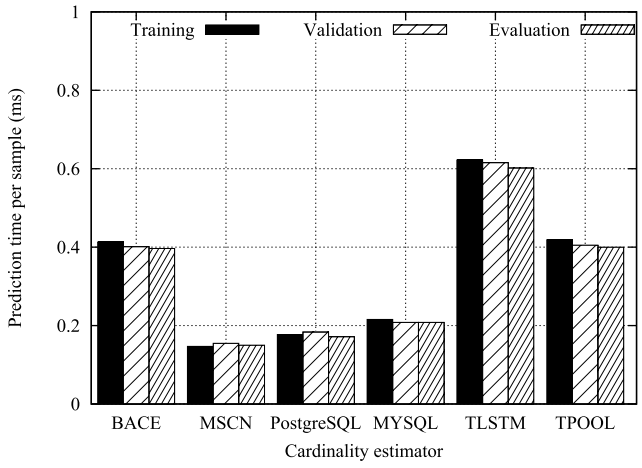

FIGURE 5 Prediction time comparison of different cardinality estimators

6 | CONCLUSION

In this study, we propose a cardinality estimator based on BiLSTM-Attention. We first use Word2vec technology to transform the query into word vectors, then use BiLSTM to model the word vector sequence, input the result at each time slice into the attention network, and combine it with the sample bitmap to obtain a new vector and finally estimate the cardinality according to the word vector. We solve the problem that it is difficult to obtain the relations of multiple table joins, and our model can cope with the complex string type of predicates. In summary, we believe that machine learning has a bright future in handling all kinds of query optimisation tasks.

ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China under grant nos. 61772091, 61802035, 61962 006, 61962038, U1802271, U2001212, and 62072311; the Sichuan Science and Technology Program under grant nos. 2021JDJQ0021 and 22ZDYF2680; the CCF-Huawei Database System Innovation Research Plan under grant no. CCF-HuaweiDBIR2020004A; Digital Media Art, Key Laboratory of Sichuan Province, Sichuan Conservatory of Music, Chengdu, China under grant no. 21DMAKL02; the Chengdu Major Science and Technology Innovation Project under grant no. 2021-YF08-00156-GX; the Chengdu Technology Innovation and Research and Development Project under grant no. 2021-YF05-00491-SN; the Natural Science Foundation of Guangxi under grant no. 2018GXNSFDA138005; the Guangdong Basic and Applied Basic Research Foundation under grant no. 2020B1515120028; the Science and Technology Innovation Seedling Project of Sichuan Province under grant no 2021 006 and the College Student Innovation and Entrepreneurship Training Program of Chengdu University of Information Technology under grant nos. 202 110 621 179 and 202 110 621 186.

CONFLICT OF INTEREST

The authors declared that they have no conflicts of interest to this work.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Shaojie Qiao https://orcid.org/0000-0002-4703-780X

REFERENCES

1. Kipf, A., et al.: Learned cardinalities: estimating correlated joins with deep learning. In: 9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13–16, 2019, Online Proceedings (2019). [Online]. www.cidrdb.org. http://cidrdb.org/cidr2019/papers/p101-kipf-cidr19.pdf
2. Ya-nan, L., et al.: Survey on target site prediction of human mirna based on deep learning. Computer Sci. 48(1), 209–216 (2021)
3. Yan, Z., Tian-rui, L.: Review of comment-oriented aspect-based sentiment analysis. Computer Sci. 47(6), 194–200 (2020)
4. Miroshkina, M.V., Briskin, E.S.: On the optimal modes of controlled transfer of walking propulsion devices. J. Artif. Intell. Technol. 1(3), 174–179 (2021)
5. Zhou, P., et al.: Attention-based bidirectional long short-term memory networks for relation classification. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany. Short Papers, vol. 2. The Association for Computer Linguistics (2016). https://doi.org/10.18653/v1/p16-2034. [Online]
6. Mikolov, T., et al.: Efficient estimation of word representations in vector space. In: Bengio, Y., LeCun, Y. (eds.) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings (2013). [Online]. http://arxiv.org/abs/1301.3781
7. Cronj, L., Sanders, I.: Semiautomated class attendance monitoring using smartphone technology. J. Artif. Intell. Technol. 1(1), 9–20 (2020)
8. Regazzoni, F., et al.: Protecting artificial intelligence ips: a survey of watermarking and fingerprinting for machine learning. CAAI Trans. Intell. Technol. 6(1), 180–191 (2021)
9. Krishnan, S., et al.: Learning to optimise join queries with deep reinforcement learning. CoRR, abs/1808.03196 (2018). [Online]. http://arxiv.org/abs/1808.03196
10. Marcus, R., Papaemmanouil, O.: Deep reinforcement learning for join order enumeration. In: Bordawekar, R., Shmueli, O. (eds.) Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, aiDM@SIGMOD 2018, Houston, TX, USA, June 10, 2018, pp. 3:1–3:4. ACM (2018). https://doi.org/10.1145/3211954.3211957. [Online]
11. Ortiz, J., et al.: Learning state representations for query optimisation with deep reinforcement learning. In: Schelter, S., Seufert, S., Kumar, A. (eds.) Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning, DEEM@SIGMOD 2018, Houston, TX, USA, June 15, 2018, pp. 4:4–4:1. ACM (2018). https://doi.org/10.1145/3209889.3209890. [Online]
12. Malik, T., Burns, R.C., Chawla, N.V.: A black-box approach to query cardinality estimation. In: Third Biennial Conference on Innovative Data Systems Research, CIDR 2007, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings, pp. 56–67 (2007). [Online]. www.cidrdb.org. http://cidrdb.org/cidr2007/papers/cidr07p06.pdf
13. Akdere, M., et al.: Learning-based query performance modelling and prediction. In: Kementsietsidis, A., Salles, M.A.V. (eds.) IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012, pp. 390–401. IEEE Computer Society (2012). https://doi.org/10.1109/ICDE.2012.64. [Online]
14. Hilprecht, B., et al.: Deepdb: learn from data, not from queries! Proc. VLDB Endow. 13(7), 992–1005 (2020). [Online]. http://www.vldb.org/pvldb/vol13/p992-hilprecht.pdf
15. Yang, Z., et al.: Deep unsupervised cardinality estimation. Proc. VLDB Endow. 13(3), 279–292 (2019). [Online]. http://www.vldb.org/pvldb/vol13/p279-yang.pdf
16. Zhu, R., et al.: FLAT: fast, lightweight and accurate method for cardinality estimation. CoRR. abs/2011.09022 (2020). [Online]. https://arxiv.org/abs/2011.09022
17. Liu, H., et al.: Cardinality estimation using neural networks. In: Gould, J., Litoiu, M., Lutfiyya, H. (eds.) Proceedings of 25th Annual International Conference on Computer Science and Software Engineering, CASCON 2015, Markham, Ontario, Canada, 2-4 November, 2015, pp. 53–59. IBM/ACM (2015). [Online]. http://dl.acm.org/citation.cfm?id=2886453
18. Chen, Y., Yi, K.: Two-level sampling for join size estimation. In: Salihoglu, S., et al. (eds.) Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017, pp. 759–774. ACM (2017). https://doi.org/10.1145/3035918.3035921. [Online]
19. Estan, C., Naughton, J.F.: End-biased samples for join cardinality estimation. In: Liu, L., et al. (eds.) Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA, p. 20. IEEE Computer Society (2006). https://doi.org/10.1109/ICDE.2006.61. [Online]
20. Vengerov, D., et al.: Join size estimation subject to filter conditions. Proc. VLDB Endow. 8(12), 1530–1541 (2015). [Online]. http://www.vldb.org/pvldb/vol8/p1530-vengerov.pdf
21. Wu, W., Naughton, J.F., Singh, H.: Sampling-based query re-optimisation. In: Özcan, F., Koutrika, G., Madden, S. (eds.) Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26–July 01, 2016, pp. 1721–1736. ACM (2016). https://doi.org/10.1145/2882903.2882914. [Online]
22. Müller, M., Moerkotte, G., Kolb, O.: Improved selectivity estimation by combining knowledge from sampling and synopses. Proc. VLDB Endow. 11(9), 1016–1028 (2018). [Online]. http://www.vldb.org/pvldb/vol11/p1016-muller.pdf
23. Qiao, S., et al.: Cardinality estimator: processing SQL with a vertical scanning convolutional neural network. J. Comput. Sci. Technol. 36(4), 762–777 (2021). https://doi.org/10.1007/s11390-021-1351-7. [Online]
24. Zhou-jun, L., Yu, F., Xian-jie, W.: Survey of natural language processing pre-training techniques. Computer Sci. 47(3), 162–173 (2020)
25. Zhang, S., et al.: Bidirectional long short-term memory networks for relation classification. In: Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation, PACLIC 29, Shanghai, China, October 30–November 1, 2015. ACL (2015). [Online]. https://www.aclweb.org/anthology/Y15-1009/
26. Mnih, A., Teh, Y.W.: A fast and simple algorithm for training neural probabilistic language models. In: Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26–July 1, 2012. icml.cc/Omnipress (2012). [Online]. http://icml.cc/2012/papers/855.pdf
27. Mnih, A., Hinton, G.E.: A scalable hierarchical distributed language model. In: Koller, D., et al. (eds.) Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008, pp. 1081–1088. Curran Associates, Inc. (2008). [Online]. https://proceedings.neurips.cc/paper/2008/hash/1e056d2b0ebd5c878c550da6ac5d3724-Abstract.html
28. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. 9(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735. [Online]
29. Graves, A., Mohamed, A., Hinton, G.E.: Speech recognition with deep recurrent neural networks. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26–31, 2013, pp. 6645–6649. IEEE (2013). https://doi.org/10.1109/ICASSP.2013.6638947. [Online]
30. Leis, V., et al.: How good are query optimisers, really? Proc. VLDB Endow. 9(3), 204–215 (2015). [Online]. http://www.vldb.org/pvldb/vol9/p204-leis.pdf
31. Leis, V., et al.: Cardinality estimation done right: index-based join sampling. In: 8th Biennial Conference on Innovative Data Systems Research,

CIDR 2017, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings (2017). [Online]. www.cidrdb.org. http://cidrdb.org/cidr2017/papers/p9-leis-cidr17.pdf
32. Zimmer, R., et al.: Technical report: supervised training of convolutional spiking neural networks with pytorch. CoRR. abs/1911.10124 (2019). [Online]. http://arxiv.org/abs/1911.10124
33. Al-Mouhamed, M.A., ul Hasan Khan, A., Mohammad, N.: A review of CUDA optimisation techniques and tools for structured grid computing. Computing. 102(4), 977–1003 (2020). https://doi.org/10.1007/s00607-019-00744-1. [Online]
34. Hinton, G.E., et al.: Improving neural networks by preventing co-adaptation of feature detectors. CoRR. abs/1207.0580 (2012). [Online]. http://arxiv.org/abs/1207.0580
35. Sun, J., Li, G.: An end-to-end learning-based cost estimator. Proc. VLDB Endow. 13(3), 307–319 (2019). [Online]. http://www.vldb.org/pvldb/vol13/p307-sun.pdf

How to cite this article: Zhou, Q., et al.: A BiLSTM cardinality estimator in complex database systems based on attention mechanism. CAAI Trans. Intell. Technol. 7(3), 537–546 (2022). https://doi.org/10.1049/cit2.12069