Data preprocessing in MySQL

Group 6

CHING HUSAN LIU*
Data Science Degree Program
National Taiwan University
Taipei, Taiwan
r10946013@ntu.edu.tw

YU SYUAN CHUANG
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
r10921095@ntu.edu.tw

YAN TING YE
Communication Engineering
National Taiwan University
Taipei, Taiwan
r10942153@ntu.edu.tw

CHIAO YIN TENG
Bioenvironmental Systems Engineering
National Taiwan University
Taipei, Taiwan
b07602001@ntu.edu.tw

ABSTRACT

Data mining and data analysis usually cost lots of effort and are time-consuming. When we train models in machine learning, data pre-processing has a great influence on the performance of the model. It is time-consuming to write different data pre-processing programs for different datasets.

In this article, we present practical issues and common solutions when preparing and transforming datasets with SQL language. We implement common data pre-processing, such as filling blank values, removing duplicate data, statistical applications (quartiles, percentiles), one-hot encoding, image augmentation and feature correlation. Users only need to add specified parameters in the query field to get the pre-processed data directly.

KEYWORDS

data pre-processing, image augmentation, feature correlation, encoding, MySQL.

1 Data pre-processing

Figure 1 is the dataset we implement null value filling. You can see that the bottom column has the missing value “NULL”.

```sql
CREATE TABLE Book (
    ISBN int NOT NULL DEFAULT 0,
    Price int CHECK(Price>0),
    type ENUM('一般圖書', '兒童圖書', '成人書籍') NOT NULL,
    Author_ID int NOT NULL,
    Editor_ID int,
    fk_Libraryitem_id int NOT NULL,
    PRIMARY KEY (ISBN),

    FOREIGN KEY (Author_ID) REFERENCES Author(Author_ID),
    FOREIGN KEY (Editor_ID) REFERENCES Editor(Editor_ID_),
    FOREIGN KEY (fk_Libraryitem_id) REFERENCES Library_Item(ID),
    UNIQUE (fk_Libraryitem_id)
);
```

```sql
INSERT INTO Book
VALUES
(1234567890, 1000, '一般圖書', 1, 1, 1),
(1356782901, 400, '成人書籍', 2, 2, 2),
(1107893846, 400, '兒童圖書', 3, 4, 3),
(1034128394, NULL, '一般圖書', 4, 3, 4);
```

ISBN	Price
1034128394	NULL
1107893846	400
1234567890	1000
1356782901	400

4 rows in set (0.00 se

Figure 1: dataset of the missing value

1.1 Missing value

First, we calculate the prices of the other three types of book, and then update the missing value to the average. The result shows in Figure 2.

```sql
DELIMITER $$
Create procedure SUM_Price(OUT total INT)
begin
    select sum(Price) INTO total from Book;
end $$
DELIMITER ;

call SUM_Price(@sum_);
SELECT @sum_;


## FIND AVG
DELIMITER $$
Create procedure Count_Price(OUT total INT)
begin
    select COUNT(*) INTO total from Book WHERE Price IS NOT NULL;
end $$
DELIMITER ;

call Count_Price(@total_);
SELECT @total_;


SET @avg = @sum_/@total_;
select @avg;
```

```sql
## 更新NULL為指定資訊
DELIMITER $$
Create Procedure Update_info (IN avg_price INT)
    BEGIN
    UPDATE LibrarySystem.BOOK
    SET Book.Price = avg_price
    WHERE Book.Price IS NULL;
END $$
DELIMITER ;

CALL Update_info(@avg);
Select * from Book;
DROP PROCEDURE IF EXISTS Update_info;
```

ISBN	Price	type	Author_ID	Editor_ID	fk_Libraryitem_id
1034128394	600	一般圖書	4	3	4
1107893846	400	兒童圖書	3	4	3
1234567890	1000	一般圖書	1	1	1
1356782901	400	成人書籍	2	2	2

4 rows in set (0.00 sec)

Figure 2: implement of the average

For maximum value, we take the maximum price of the other three types of book, and then update the value to the maximum. The result shows in Figure 3.

```sql
## 找最大值
DELIMITER $$
Create procedure Max_num(OUT output INT)
begin
    SELECT MAX(Price) INTO output FROM Book;
    -- AS current;
end $$
DELIMITER ;

call Max_num(@max_num);
SELECT @max_num;
```

ISBN	Price	type
1034128394	1000	一般圖書
1107893846	400	兒童圖書
1234567890	1000	一般圖書
1356782901	400	成人書籍

4 rows in set (0.00 sec)

Figure 3: implement of the maximum

For calculation of mode, we set a count to calculate the price of occurrences of the other three types of book, and then update the value to the mode. The result shows in Figure 4.

```sql
## FIND 出現次數最多的值
DELIMITER $$
Create procedure MaxCount_num(OUT output INT)
begin
    SELECT Price INTO output FROM
        (SELECT DISTINCT Price , count( * ) AS count FROM Book
        GROUP BY Price  ORDER BY count DESC  LIMIT 1)
    AS current; ### 注意要加這個才可以
end $$
DELIMITER ;

call MaxCount_num(@maxcount_num);
SELECT @maxcount_num;
```

ISBN	Price	type
1034128394	400	一般圖書
1107893846	400	兒童圖書
1234567890	1000	一般圖書
1356782901	400	成人書籍

4 rows in set (0.00 sec)

Figure 4: implement of the mode

The random value in the range is to find the maximum and minimum values of the price, and then use the RAND() function to find the random value and then update the value to the missing value. The result shows in Figure 5.

```sql
## 找最大值最小值
DELIMITER $$
Create procedure MaxMin_num(OUT output_max INT, OUT output_min INT)
begin
    SELECT MAX(Price) INTO output_max FROM Book;
    SELECT MIN(Price) INTO output_min FROM Book;
end $$
DELIMITER ;

call MaxMin_num(@max_num, @min_num);
SELECT @max_num;
SELECT @min_num;


-- 產生大於或等於 column最小值 且小於或等於 column最大值 的整數亂數
DELIMITER $$
Create procedure Random_num(IN max_num INT, IN min_num INT,OUT output INT)
begin
    SELECT FLOOR(RAND() * (max_num - min_num + 1) + min_num) INTO output;
end $$
DELIMITER ;

call Random_num(@max_num, @min_num, @rand_num);
SELECT @rand_num;
```

ISBN	Price	type
1034128394	975	一般圖書
1107893846	400	兒童圖書
1234567890	1000	一般圖書
1356782901	400	成人書籍

4 rows in set (0.00 sec)

Figure 5: implement of the random value

1.2 Statistical applications

We designed a new dataset to find the quartile of the data and made a statistical table. Figure 6 is the dataset we implement quartile.

```sql
/* create table */
CREATE TABLE BOOK_ITEM (
    Category varchar(40),
    Book_ID varchar(40),
    AssessedValue INT
);

INSERT INTO BOOK_ITEM
VALUES
("漫畫","1",441),
("漫畫","2",447),
("漫畫","3",230),
("漫畫","4",496),
("漫畫","5",300),
("漫畫","6",525),
("童書","7",295),
("童書","8",379),
("童書","9",289),
("童書","10",331),
("童書","11",313),
("童書","12",220);
```

```sql
SET sql_mode = 'STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';


SELECT Category, Book_ID, AssessedValue,
    -- Bucket the assessed value for each property into quartiles based on its community
    NTILE(4) OVER (ORDER BY AssessedValue) AS Quartile
FROM BOOK_ITEM;

SELECT
    Category,
    -- Get the minimum value for the community
    MIN(AssessedValue) Minimum,
    -- Get the 1st quartile boundary for the community, which is the highest value in the 1st quartile
    MAX(CASE WHEN Quartile = 1 THEN AssessedValue END) 1Quartile,
    -- Get the median for the community, which is the highest value in the 2nd quartile
    MAX(CASE WHEN Quartile = 2 THEN AssessedValue END) Median,
    -- Get the 3rd quartile boundary for the community, which is the highest value in the 3rd quartile
    MAX(CASE WHEN Quartile = 3 THEN AssessedValue END) 3Quartile,
    -- Get the maximum value for the community
    MAX(AssessedValue) Maximum,
    -- Get a count of the total properties in the community
    COUNT(Quartile) AS Count
FROM (
    SELECT
        Category, Book_ID, AssessedValue,
        -- Bucket the assessed value for each property into quartiles based on its community
        NTILE(4) OVER (ORDER BY AssessedValue) AS Quartile
    FROM
        BOOK_ITEM
) Vals
GROUP BY
    Category
ORDER BY
    Category;
```

Figure 6: dataset of the quartile

However, when implementing the GROUP BY, an error message may occur: "Error Code: 1055." or "Error Code: 1140." The solution is to delete the function that is enabled by default in MySQL and reset the "SQL_mode" to solve the problem. The results of the final implementation of quartiles are shown in Figure 7, and the quartiles of each category are calculated according to the category of books.

Category	Book_ID	AssessedValue	Quartile
童書	12	220	1
漫畫	3	230	1
童書	9	289	1
童書	7	295	2
漫畫	5	300	2
童書	11	313	2
童書	10	331	3
童書	8	379	3
漫畫	1	441	3
漫畫	2	447	4
漫畫	4	496	4
漫畫	6	525	4

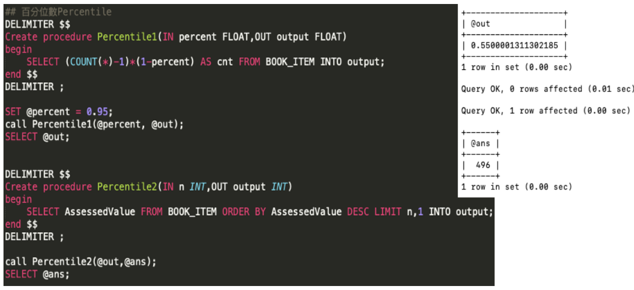
12 rows in set (0.00 sec)

Category	Minimum	1Quartile	Median	3Quartile	Maximum	Count
漫畫	230	230	300	441	525	6
童書	220	289	313	379	379	6

2 rows in set (0.00 sec)

Figure 7: implement of the quartile

For percentile, users can set the percentage by themselves, and finally output the percentile of the category of books. The result shows in Figure 8.



```
## 百分位數Percentile
DELIMITER $$
Create procedure Percentile1(IN percent FLOAT,OUT output FLOAT)
begin
    SELECT (COUNT(*)-1)*(1-percent) AS cnt FROM BOOK_ITEM INTO output;
end $$
DELIMITER ;

SET @percent = 0.95;
call Percentile1(@percent, @out);
SELECT @out;


DELIMITER $$
Create procedure Percentile2(IN n INT,OUT output INT)
begin
    SELECT AssessedValue FROM BOOK_ITEM ORDER BY AssessedValue DESC LIMIT n,1 INTO output;
end $$
DELIMITER ;

call Percentile2(@out,@ans);
SELECT @ans;
```

```
+--------------------+
| @out               |
+--------------------+
| 0.5500001311302185 |
+--------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 1 row affected (0.00 sec)

+------+
| @ans |
+------+
|  496 |
+------+
1 row in set (0.00 sec)
```

Figure 8: implement of the percentile

1.3 One-hot Encoding

We implement the one-hot encoding with coalesce to display 1 for the data that exists in this column, and display 0 for the rest. The result shows in Figure 9.

```
CREATE TABLE onehot (
    keyword varchar(40) NOT NULL,
    color varchar(40) NOT NULL
);

INSERT INTO onehot
VALUES
('foo','red'),
('bar','yellow'),
('baz','blue'),
('bazbaz','green');

SELECT DATABASE();
SELECT * FROM onehot;
SELECT keyword,
COALESCE(keyword='foo') as red,
COALESCE(keyword='baz') as blue,
COALESCE(keyword='bazbaz') as green,
COALESCE(keyword='bar')as yellow
FROM onehot;
```

```
+---------+--------+
| keyword | color  |
+---------+--------+
| foo     | red    |
| bar     | yellow |
| baz     | blue   |
| bazbaz  | green  |
+---------+--------+
4 rows in set (0.00 sec)

+---------+-----+------+-------+--------+
| keyword | red | blue | green | yellow |
+---------+-----+------+-------+--------+
| foo     |   1 |    0 |     0 |      0 |
| bar     |   0 |    0 |     0 |      1 |
| baz     |   0 |    1 |     0 |      0 |
| bazbaz  |   0 |    0 |     1 |      0 |
+---------+-----+------+-------+--------+
4 rows in set (0.00 sec)

Query OK, 1 row affected (0.01 sec)
```

Figure 9: implement of one-hot encoding

2 Image Augmentation

Image augmentation is a common technique of data preprocessing for image tasks. In this section, we propose several image augmentation functions in MySQL:

1. Horizontal / Vertical Shift

2. Horizontal / Vertical Flip

3. Crop

These functions are implemented purely using MySQL language. We implement them as stored functions. Once functions are stored in a MySQL server, users can call these functions to get the desired augmented images.

Here we only deal with grayscale images for simplicity.

2.1 Image in MySQL

Before writing these functions, we first need to understand how images are stored in MySQL. MySQL supports directly reading a PNG / JPG file and storing it as a BLOB object. However, in such a way we can not access pixel values of images directly. A PNG / JPG decoder and encoder would be necessary for decoding images and modifying them. But a PNG / JPG decoder and encoder in MySQL is not what we want to discuss, so we choose to store images in another way.

A grayscale image can be treated as a 2-D numeric array, with all entries are uint8. We can serialize such an array and then store it as a BLOB object. A BLOB object can be viewed as a binary string in MySQL. We can access and modify any element in a binary string.

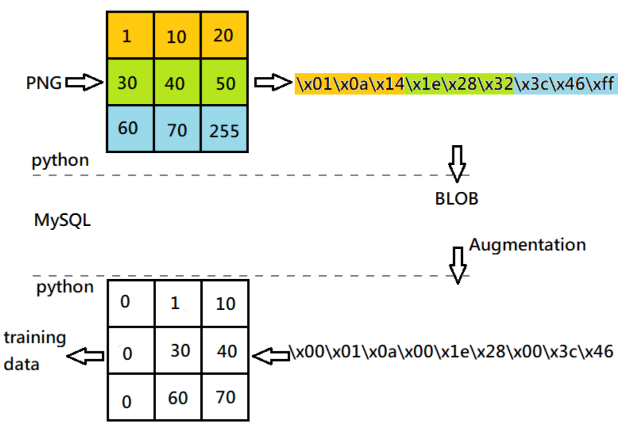


Fig. 7: Images are treated as arrays and then serialized in order to be stored in MySQL as BLOB datatype.

2.2 Function Implementation

As we mention above, any element in a BLOB object can be arbitrarily modified, which means any pixel value can be changed to the value we want. Theoretically, any kind of image augmentation can be realized in MySQL. However, we encounter some difficulties due to the lack of vector and array algorithms in

MySQL. Thus we implement shift, flip, and crop first.

The shift value, corp range are function arguments that the user can specify.

Utilizing the SUBSTRING query in MySQL to reuse the consecutive elements in original images is the main idea. Here we use right shift as an example:

```
RIGHT_SHIFT(image, w, h, s)
a_image = ‘’
pad = ‘\x00’ repeat s times
FOR i=1..h
    r = SUBSTRING(image, 0, w-s)
    row = CONCAT(pad, r)
    a_image = CONCAT(a_image, row)
RETURN a_image
```

The padding can be reused so we construct it first and then reuse it in every iteration.

Other functions are implemented in a similar way. We will discuss their complexity later.

2.3 Time Complexity Analysis

For time complexity analysis, we use the number of query calls as the measure, i.e. one query call takes one unit of time.

Here we assume that the height and the width of images are both N for simplicity. (It is reasonable because typically we have width = Θ(height) for most images.)

Horizontal shift with shift value s:

It takes O(N) to construct the padding row. We need N iterations, 2 CONCAT and 1 SUBSTRING in each iteration. The overall time complexity is Θ(N).

Vertical shift with shift value s:

It takes Θ(N) to construct the padding row. We need s CONCAT and 1 SUBSTRING. Overall time complexity is Θ(s) (or O(N)).

Horizontal flip:

We need N iterations, N CONCAT and N SUBSTRING in each iteration. Overall time complexity is $\Theta(N^2)$.

Vertical flip:

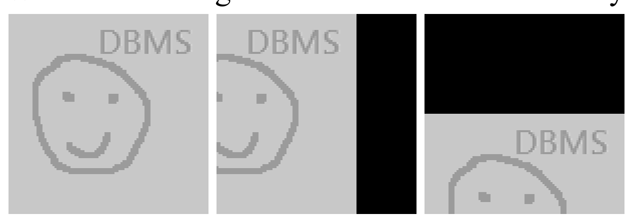
We need N iterations, 1 CONCAT and 1 SUBSTRING in each iteration. Overall time complexity is $\Theta(N)$.

Corp:

It takes Θ(N) to construct the right, left, top, and bottom padding row. We need N iterations, 1~3 CONCAT and 0~1 SUBSTRING in each iteration. Overall time complexity is $\Theta(N)$.

2.4 Result

We can see in Fig. 8 all functions work correctly.



(a) original (b) horizontal / vertical shift

(c) horizontal / vertical flip (d) corp

Fig. 8: Augmented images

To compare the execution time of our image augmentation and existing one, we execute each query 1000 times and measure the execution time. (we compare it to Pytorch's implementation.)

	h_s	v_s	h_f	v_f	c
our	8.02	2.06	700	7.69	4.89
PT	0.14	0.14	0.18	0.14	0.19

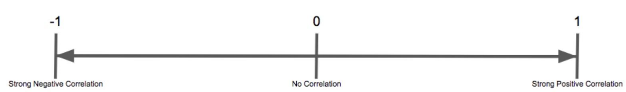
Table 1: The 1000 times execution time of each query. The h_s, v_s stands for horizontal and vertical shift; h_f, v_f stands for horizontal and vertical flip; c stands for corp. PT means using Pytorch's image augmentation implementation.

3 Feature Correlation in MySQL

The correlation coefficient is a widely used method to determine the strength of the relationship between two variables or two sets of numbers. This coefficient is a number between -1 and 1, which 1 means the strongest possible positive correlation , -1 means the strongest possible negative correlation and 0 means there is no correlation.

A positive correlation means that as one number increases the second number will also increase. A negative correlation means that as one number increases the second number decreases.

The example of possible positive correlation is : sales number and marketing spend. The example of possible negative correlation is : product price and its sales.



3.1 Correlation v.s. Causation

The possible positive correlation might not mean there is causation between two variables. For example, the strong correlation between two numbers might be caused by randomness. Also, while there might be a causation between two variables and there is a strong correlation between them, the correlation coefficient may not tell us anything about the causation between the two variables.

3.2 the Pearson Correlation Coefficient

The Pearson Correlation Coefficient is a widely used method to calculate the correlation coefficient. The original equation is:

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[n\Sigma x^2 - (\Sigma x)^2][n\Sigma y^2 - (\Sigma y)^2]}}$$

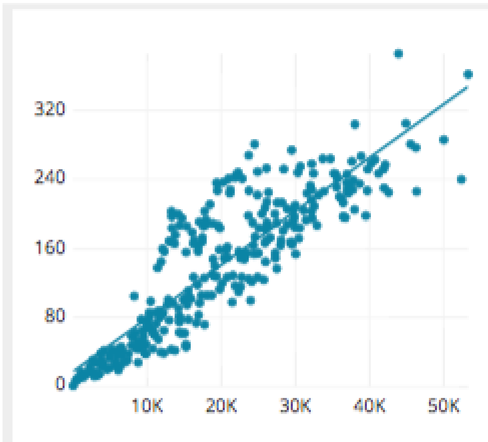
when use in MySQL:

```sql
SELECT
    (
    (tot_sum - (amt_sum * act_sum / _count)) /
    sqrt((amt_sum_sq - pow(amt_sum, 2.0) / _count) *
    (act_sum_sq - pow(act_sum, 2.0) / _count))
    )
    AS "Corr Coef Using Pearson"
```

3.3 Implement

```sql
SELECT
    ((tot_sum - (amt_sum * act_sum / _count)) /
        sqrt((amt_sum_sq - pow(amt_sum, 2.0) / _count) * (act_sum_sq - pow(act_sum, 2.0) / _count)))
    AS "Corr Coef Using Pearson"


FROM(
SELECT
    sum("Amount") AS amt_sum,
    sum("Activities") AS act_sum,
    sum("Amount" * "Amount") AS amt_sum_sq,
    sum("Activities" * "Activities") AS act_sum_sq,
    sum("Amount" * "Activities") AS tot_sum,
    count(*) as _count

FROM(
SELECT
    DATE_TRUNC('day', p.payment_date)::DATE AS "Day",
    SUM(p.amount) AS "Amount",
    COUNT(DISTINCT a.activity_id) AS "Activities"
FROM
    public.payments p
    INNER JOIN public.subscriptions s ON p.subscription_id = s.subscription_id
    INNER JOIN public.users u ON s.user_id = u.user_id
    INNER JOIN public.activity a ON a.user_id = u.user_id

GROUP BY 1) as a

) as b

GROUP BY tot_sum, amt_sum, act_sum, _count, amt_sum_sq, act_sum_sq
```

When compared to corr() , a function in PostgreSQL which can calculate the correlation coefficient directly , they have the same correlation coefficient 0.9 and its figure below.



ACKNOWLEDGMENTS

Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here.

REFERENCES

[1] Patricia S. Abril and Robert Plant, 2007. The patent holder's dilemma: Buy, sell, or troll? *Commun. ACM* 50, 1 (Jan, 2007), 36-44. DOI: https://doi.org/10.1145/1188913.1188915.

[2] Sten Andler. 1979. Predicate path expressions. In *Proceedings of the 6th. ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '79)*. ACM Press, New York, NY, 226-236. DOI:https://doi.org/10.1145/567752.567774

[3] Ian Editor (Ed.). 2007. *The title of book one* (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago.DOI:https://doi.org/10.1007/3-540-09237-4.

[4] David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY.

[5] Ordonez, C. (2011). Data set preprocessing and transformation in a database system. *Intelligent Data Analysis*, *15*(4), 613-631.

[6] Ni, J., Xie, S., Li, Z., & Jia, C. (2018, August). Optimization design method of cache extension in MySQL database. In *2018 IEEE International Conference on Mechatronics and Automation (ICMA)* (pp. 2295-2299). IEEE.

[7] JOSHI, A. P., & PATEL, B. V. (2020). Data Preprocessing: the Techniques for Preparing Clean and Quality Data for Data Analytics Process.

TEAM MEMBER WORKS

劉謦瑄: Data preprocessing、code cleaning
鄧喬尹: Encoding、video clip、written report
葉彥廷: image augmentation、 written report
莊于萱: feature correlation、written report