IRDM Course Project Part II Report

ABSTRACT

The experiment of the mordern method for retrieving information remains an important part in lost of fields. To put it into practice, it should construct the related metric firstly. After that, it should apply Word2Vec and Doc2Vec method to embed the word and sentence. Then, it applies the logistic algorithm, the XGBoost-based lambdaMART model and the neural network-based model to obtain the test-set metrics. The result shows that the mordern algorithm surpasses the traditional method in all metrics.

CCS CONCEPTS

• **Information Retrieve → Embedded systems**; *LambdaMART*; Logistic Regression; Neural Network.

KEYWORDS

embedding; learning to rank, attention, gradient descent

ACM Reference Format:

. 2018. IRDM Course Project Part II Report. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 THE METRICS FOR MEASURE RETRIEVING QUALITY

According to the requirement of the project, the NDCG and the mAP should be considered.

1.1 NDCG Metric

The NDCG can be divided into the G, DCG, and N- three parts. The G denotes the gain (using relevance score y to measure) for a certain item:

$$G(i) = \hat{(}y)(i) \tag{1}$$

Consider the sorting order factor, so that the top-ranked items gain more, and the bottom-ranked items are degraded. Based on this, it needs the DCG to measure the list relevancy, which can be expressed as:

$$DCG = \sum_{i=1}^{N} \frac{G(i)}{\log_2(i)} \tag{2}$$

According to this, DCG makes items at the front increase their influence, and items at the back decrease their influence.

To normalize the DCG so that it can make the comparison, it should introduce the IDCG, which denotes the DCG with the ideal and best rank. Then, it can define the NDCG:

$$NDCG = \frac{DCG}{IDCG} \tag{3}$$

1.2 mAP Metric

The mean precision can be defined as the area of the precision-accuracy curve with different thresholds of binary classification. Where the precision and accuracy can be defined as:

$$\begin{cases} P = \frac{TP}{TP+FP} = \frac{TP}{N} \\ A = \frac{TP}{TP+FN} \end{cases} \tag{4}$$

Then, for the threshold sequence T_i, it can obtain the area of the precision-accuracy curve as:

$$mAP = \int_0^1 AdP = \sum_{i=1}^{n} T_i A_i \tag{5}$$

However, the mAP is used for analyzing the binary classification, rather than the ranking problem. Therefore, in the preceding problem, it will be only slightly considered.

2 WORD AND SENTENCE EMBEDDING

The embedding method used for embedding words and sentences is the Word2Vec. In detail, the CBOW is applied. The thought of the algorithm is to use the surrounding words to predict the center word. According to the requirement of the problem, the embedding part can use the Gensim package. Therefore, it can summarize the step for training the word and sentence embeddings as follows:

- Read all text in the training set, which read the query text and the passage text only;
- Obtain the query-and-passage where score = 1 and merge them because only when score = 1, query and passage have the relationship;
- Concrete query, passage, and only the relevant query-and-passage as the training input;
- Delete the stopwords and the punctuation because stopwords will take impact the quality of the word embedding and the punctuation is not the word;
- Using CBOW with window size = 8 (to try to merge the context information) to train the Word2Vec model;
- Read all information, including the query, passage, query ID, passage ID, and the relevancy (only the training and validation set) from the training set, validation set, and the test set;
- Delete the stopwords, the punctuation, as well as repeat query and passage to save the memory;
- Generate the sentence embedding for query and passage, respectively by simply averaging the word embedding for each word (if the word does not occur, use the mean of the word embedding in that sentence to substitute because it will not affect the final output);

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

- Construct the look-up table (LUT) with the key of (query ID, passage ID), the value of relevancy;
- Save the sentence embedding and the corresponding LUT.

3 THE LOGISTIC ALGORITHM

3.1 Dataset and Data Preprocessing

In the logistic algorithm, the label of the dataset should be encoded as $-1, 1$, rather than $0, 1$. Therefore, it should first convert all zero to -1 in the training and validation label.

Based on this, the dataset can be expressed as:

$$D = (x, y)|y \in -1, 1 \tag{6}$$

In this problem, it can use all datasets in the training set to train the model. The validation set is not applied because the distribution of the validation set is the same as the training set.

3.2 The Cross Feature Design

In task2 and task3, it needs to send the query and the passage embedding, which may induce the shape incorrectness problem. To solve it, the cross feature design is proposed. In this algorithm, it needs to claculate the pointwise add and the pointwise multiply, then add them together, which can be expressed as:

$$x_{new} = (x_1 + x_2) \cdot x_1 \cdot x_2 = x_1^2 \cdot x_2 + x_2^2 \cdot x_1 \tag{7}$$

From the formula, it can be observed that the feature has second order crossing because it calculate the second order suqare for each feature and cross them together.

3.3 Forward Pass of the Logistic Regression

In common linear regression, it can calculate the output as:

$$L = w^T x + b \tag{8}$$

Where w, b are the trainable parameters. However, based on the definition of the logistic regression, it should output the probability. Therefore, the Sigmoid function should be considered:

$$S(z) = \frac{1}{1 + e^{-z}} \tag{9}$$

Therefore, the forward pass of the logistic regression can be summarized as:

$$P = \frac{1}{1 + e^{-(w^T x + b)}} \tag{10}$$

3.4 Backward Pass of the Logistic Regression

The loss function of the logistic regression is the negative log-likelihood function of the dataset because the optimization object is to find the value of w, b so that the output probability can be near the real probability. Based on this, the loss function can be expressed as:

$$L_f = -\sum_{i=1}^{n} lnS(yL) \tag{11}$$

To optimize the parameters, it should calculate the gradient. For convenience, it can extend the weight and the input vector as:

$$\begin{cases} w_e = [b, w]^T \\ x_e = [1, x]^T \end{cases} \tag{12}$$

After that, it can calculate the gradient using the chain derivation rule:

$$\frac{\partial L_f}{\partial w_e} = -\sum_{i=1} n(1 - S(yL))yx_e \tag{13}$$

After that, according to the gradient descent algorithm, it can calculate the new parameters as:

$$w_e \leftarrow w_e - \alpha \frac{\partial L_f}{\partial w_e} \tag{14}$$

Where α denotes the learning rate, for the most common value, the learning rate is $\alpha = 1e - 3$.

3.5 Model Performance Analysis

For the logistic model, it can obtain the loss curve over the iteration in Fig.1.

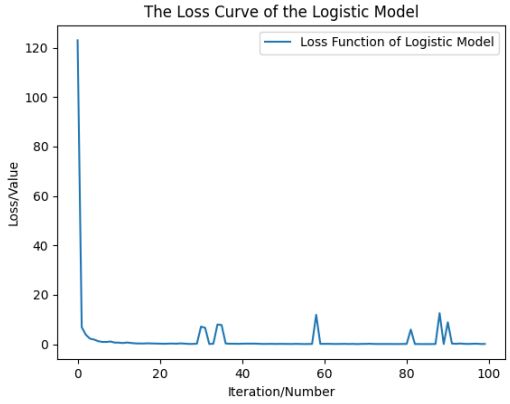

Figure 1: The Loss Curve of the Logistic Model

From this figure, it can know that the loss decreases much faster in the first 10 iterations, which states that the algorithm is correct. In addition, it can also be witnessed that in several epochs, the loss curve increases rapidly. This is because of the unreasonable sample of the training set.

In conclusion, the model performance can be guaranteed because the loss is directly linked with the NDCG metric.

4 THE XGBOOST-ITERATED LAMBDAMART ALGORITHM

4.1 Dataset and Data Preprocessing

Unlike the logistic regression, because the XGBoost is extremely sensitive to the data size. To accelerate training, it only uses the first 10000 items in the training set for training the model. This is reasonable because the distribution of the first 10000 items is almost the same as the whole distribution.

The data preprocessing method is the same as the logistic regression.

4.2 The Parameters Choice

According to the requirement of the project, the LambdaMART algorithm does not need to implement from the ground, which can be assisted by the XGBoost algorithm. Therefore, the task is to fine-tune the model. Firstly, it should select the fine-tune parameters:

- The objective, this parameter is crucial, but it can not be changed because only when it is set to rank:ndcg that it can apply the LambdaMART algorithm;
- The eta, which means the learning rate. Unlike the learning rate in a neural network or logistic regression, the learning rate of the XGBoost should be extremely large;
- The max depth, which affects the tree construction so that the classification and the ranking effectiveness would be changed.

Unlike the traditional method, the gradient descent algorithm can be applied to fine-tune the model.

4.3 The Gradient Descent-Based Fine-Tune Method

Given two group of the adjustable parameters $g_1 = (\eta_1, N_1)$ and $g_1 = (\eta_2, N_2)$, it can calculate the gradient according to the definition of the gradient:

$$\nabla g = (\frac{L_{f21} - L_{f11}}{\eta_2 - \eta_1}, \frac{L_{f22} - L_{f12}}{N_2 - N_1}) \tag{15}$$

Where L_f here should be the loss function, L_{fij} denotes the weight of the loss function, which can be expressed as:

$$\begin{cases} L_{fi1} = 0.01 L_f \\ L_{fi2} = 0.99 L_f \end{cases} \tag{16}$$

This factor must be considered, because the scale of the variable is too large, which will cause significant gradient explosion and gradient disappearance. Therefore, it is necessary to consider weighting to ensure its stable convergence. The weighted coefficient represents the relationship between the scales of variables. The scale of $\eta_i$is about N_i is about one percent of the scale, so its coefficients are 0.01 and 0.99 respectively.

As for the loss function, it can apply the MSE loss function, because the training NDCG can be obtained while the label is given. According to the definition of the NDCG, the value should near 1.0. Therefore, it can apply it to force the algorithm to output the value that is near NDCG of 1.0, which can be expressed as:

$$L_f = (NDCG - 1.0)^2 \tag{17}$$

After calculating the gradient, it needs to update the parameters. The process is like the logistic regression:

$$\begin{cases} \eta_2 \leftarrow \eta_2 - \beta \nabla g_0 \\ N_2 \leftarrow N_2 - \beta \nabla g_1 \end{cases} \tag{18}$$

Where β denotes the learning rate of the gradient descent algorithm, rather than the XGBoost itself. In addition, the η_1 and N_1 should be updated to the original value of η_2 and N_2 respectively.

4.4 The Model Performance

Like the logistic regression, it can obtain the loss curve based on different metric parameters ("ndcg-", "ndcg", "map-", and "map") over the iteration respectively, shown in Fig.

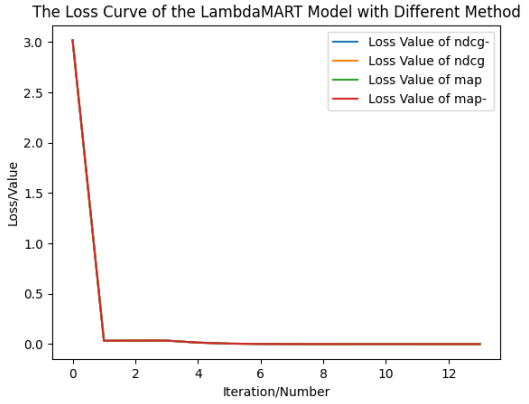

Figure 2: The Loss Curve of the LambdaMART Algorithm with Different Methods

From the figure, it can know that the loss curve can be seen as a similar trend in the logistic algorithm. However, the iteration number is considerably lower than that in the logistic algorithm. As for the optimal parameters, that is:

$$\begin{cases} \eta = 0.0012218448943188385 \\ N = 10 \end{cases} \tag{19}$$

It achieves only $1e-7$ of the NDCG loss function. Because the dataset is relatively large, so it will not reach overfitting.

In conclusion, the model performance is promising because no overfitting is discovered and only 15 iterations is made for convergence of the algorithm.

5 NEURAL NETWORK-BASED METHOD

5.1 Dataset and Preprocessing

In this problem, the whole dataset of the training set is used with the same preprocessing steps as the logistic regression. However, it should transfer the data into the GPU so that the training can be accelerated effectively.

5.2 Literature Review

After reviewing the literature, it can classify the model into 3 different types:

- The common neural network, such as SENet[4], ResNet[3], etc. These networks aim to output the value that is near the true value. However, these networks can not implement the rank;
- The context-based network, such as the BERT[1], Transformer[6], etc. These networks can extract the context information while it can apply to the learning to rank task. However, these network needs the sequence length while the dataset has no such feature;

- The object-detection network, such as YOLOV4[2], SSD[5], etc. These networks are able to output lots of bounding boxes with the different classes, which can apply to the learning to rank task. However, these networks need the 2D input of the images, rather than the 1D vector.

Integrating the characteristics of these different networks, it can design the unique model with the vector-based input, with the attention mentioned in the common neural network and BERT, the similar structure with the BERT, and the multiple outputs with the object-detection network.

5.3 Loss Function Design

The backbone model is the BERT model, with the output length is 1000 because there are 1000 passages for one query in the test set. The meaning of the output is the relevancy score for each item.

Therefore, it should set the objective as the NDCG value like the XGBoost. Consider the real label is sparse and the value is equal to 1, to represent this, it can obtain the IDCG as:

$$IDCG = 1 + \frac{1}{log_2(3)} + \frac{1}{2} + ... + \frac{1}{log_2(i+1)} \tag{20}$$

The total number of items for calculating IDCG is considerably less than that to calculate the NDCG.

Likewise, the nature of the NDCG Loss is the MSE Loss function, which is the same as that mentioned in the XGBoost model.

5.4 Network Structure Design

5.4.1 The Backbone. The encoder layer in BERT consists of the layer norm layer, the attention layer, and the feedforward layer, which can be seen in Fig.3.

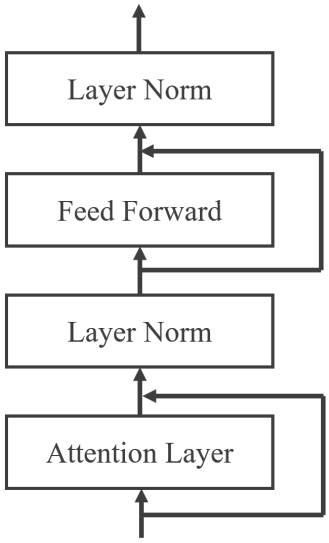

Figure 3: The Basic Component of One Encoder Layer

After stacking lots of encoder layers, it can obtain the final represent vector as the output.

5.4.2 The Activation Function. The original BERT applies the ReLU activation function for each layer. However, this activation function is not enough for extracting features. Therefore, it can refer to the YOLO V4 network, which uses the Swish activation function, which is expressed as:

$$SiLU(x) = x \cdot S(x) \tag{21}$$

The activation function can be seen as the smooth of the ReLU function, which prevents the dead neuron problem.

5.4.3 The Bottle Neck CSP Layer. In traditional BERT, the feed-forward layer is just to merge the feature. However, because in this problem, the input is the 1D vector, which means, only the multi-layer perceptron can be applied, the feature merging in BERT is not enough. Therefore, it can substitute the feed-forward layer for the bottleneck CSP structure learned from the YOLO V4 network. It can draw the structure as Fig.4.

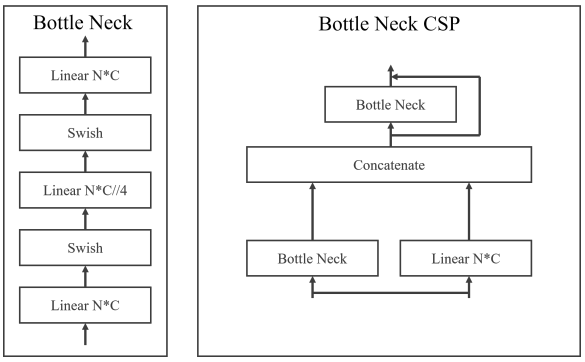

Figure 4: The Structure of the Bottle Neck CSP Layer

In this figure, C denotes the channel of the network, and $N = 1000$ denotes the batch size. From this, it can know that the bottleneck CSP can extract more features than using the simple bottleneck structure and the simple feed-forward layer because it can force the network to learn the representation otherwise the loss would not be decreased. In addition, the bottleneck structure is like the dimensional reduction and recovering process, which means, the network is learned to preserve more useful information, which enhances the feature extraction ability.

5.4.4 The Attention Layer. In the original BERT, the attention layer is called multi-head attention. However, this attention needs the sequence length while the dataset has no such feature. Therefore, it can apply another kind of attention method, that is the weight method learned from SENet.

The method plot can be seen in Fig.5.

Just as in the figure illustration, the layer has the squeeze and excitation parts. Where the information first uses the linear layer to squeeze to the shape of $(N, 1)$. Then, using the activation function and transpose method, it can change the shape of $(1, N)$. After the last linear layer and the Sigmoid function, it can provide the weight of each item in the batch, which denotes the importance of each item. Ultimately, it should perform the pointwise multiply to multiply the weight of all items.

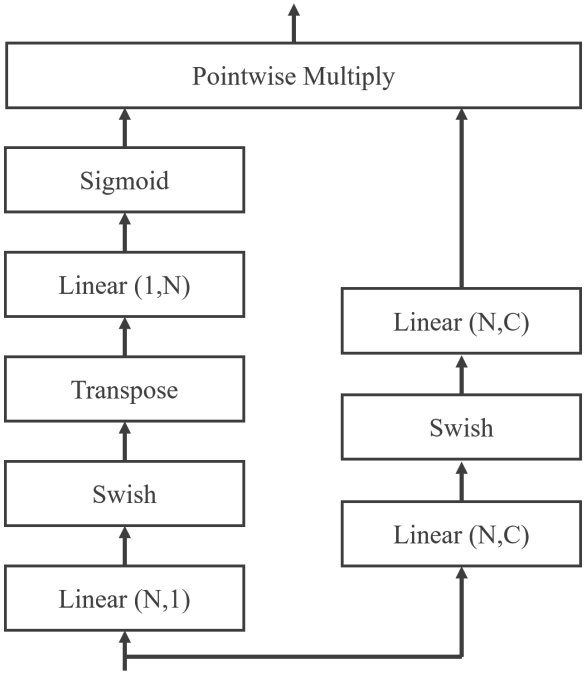

Figure 5: The Attention Layer

The use of attention enhances the importance of the item, which can have a positive effect on the rank because a higher rank means higher importance.

5.5 Model Performance

After training for one epoch, the average loss if 0.07, which is more than that in XGBoost, but less than the logistic regression. However, after training for 2 epochs, the average loss is less than $1e-6$, which is considerably less than that in XGBoost. Therefore, the model performance can be guaranteed in this sense.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[2] Rongli Gai, Na Chen, and Hai Yuan. 2021. A detection algorithm for cherry fruits based on the improved YOLO-v4 model. *Neural Computing and Applications* (2021), 1–12.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[4] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).