Learning to Rank for Information Retrieval

IDENTIFIERS

Zhen Qian
S3888611

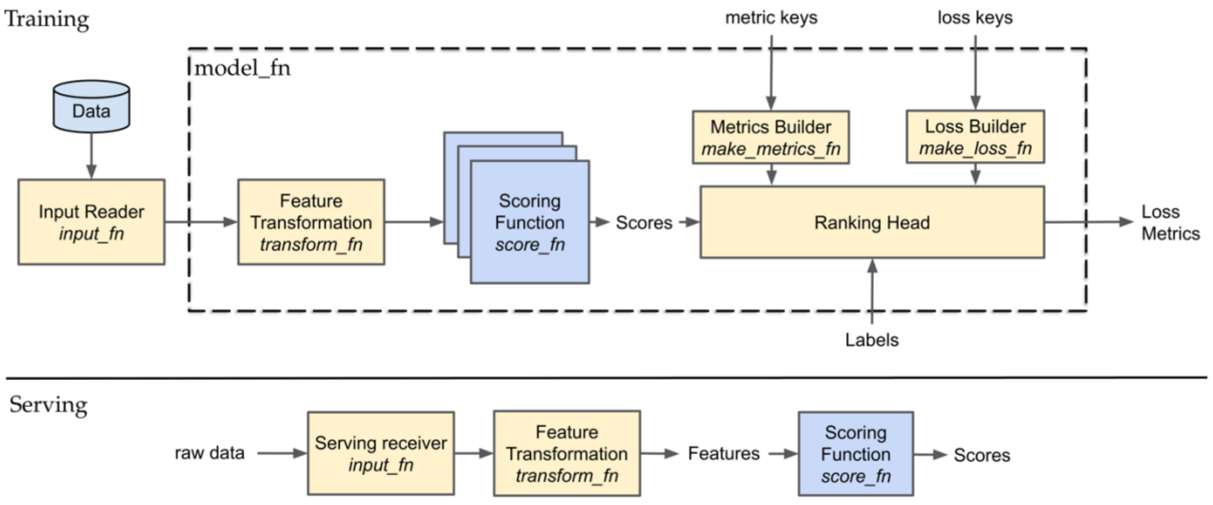
OVERVIEW

The purpose of this project is to find a ranking model to estimate the relevance between query-document pairs. In this submission, the function used to score each query-document pair is the feedforward neural network model. The loss function used to optimise the scoring function is listwise loss function. The model is built, trained, and evaluated with the python library of Tensorflow and Tensorflow Ranking.

MODEL DESCRIPTION

The architecture of the ranking model used for this project is adapted from the architecture showed in the tutorial for Tensorflow Ranking library (Github Tensorflow Ranking, 2021).

Figure 1: TensorFlow Ranking Architecture



Source: https://github.com/tensorflow/ranking/blob/master/tensorflow_ranking/examples/handling_sparse_features.ipynb

As showed in the image above, the training process of the ranking model starts from building an input reader to convert the raw data into a form that Tensorflow can recognize and take in. The data needs to be transformed to dense tensors before putting it into the scoring function. Metrics function is built to evaluate the score function and loss function is built to optimise the score function. In the testing process, need to build an input function for the raw data as well. The dataset will be transformed before putting into the trained model. The scores for testing data will be derived from the trained model.

The hyperparameters that have been taken into experiments in this project are hidden layer dimensions, learning rate, and dropout rate. Hidden layer dimensions are simply the number of hidden layers for the neural networks and the number of nodes for each hidden layer. Learning rate measure the speed of the learning process. It represents to what extent the old weights of inputs are overwritten by new weights in case of neuro networks. Dropout rate represents to what extent the weights of some inputs are ignored during the training process. By doing this, each input can keep independent to some extent.

EXPERIMENTS

The scoring function used in this project is a feedforward neural network model. It estimates the weights for all inputs and calculate the sum product of the weights and inputs. The loss function kicks in and calculates the error (also known as loss) between the calculated results and the actual outputs. This error is fed back to the network and the weights will be adjusted, so that the error can be reduced. Neural networks have been widely used in learning to rank. One of the advantages of neural networks is that it performs well when dealing with high dimensional inputs. (Ai, Qingyao et al., 2019) The raw data provided with this project includes query-document pairs where each query has multiple documents correspond to it and each document has 46 features. It is believed that neural networks would be a good choice for handling data with such dimensions and shapes.

The evaluation metrics used for this project is NDCG scores and the loss function used is a built-in function called 'approx_ndcg_loss', simply because they can evaluate optimize the model so that the estimation on test data can get the best NDCG score.

The model used for estimation was trained and evaluate using Tensorflow and Tensorflow Ranking libraries. There are basically 7 steps to build the entire ranking model. I mainly followed the steps as specified in the demo tutorial for the tensorflow-ranking library. The details of what I did for each step are shown below.

1. Transform the raw data to a form that tensorflow can take in.

Some codes are borrowed from the libsvm example (Github Tensorflow Ranking, 2021) to complete this step. Tensorflow requires that the inputs must be a function to generate tensor datasets. It takes several sub-steps to turn raw data stored in a tsv file to a function. I adapted the method used in the example provided by Tensorflow Ranking library to transform our raw data. Similarly, I built a function to read tsv data with pandas library, group the data by query IDs, and extract the values for each feature and append them to the feature dictionary. I also reshaped the labels array for each query. The vector for each feature is reshaped to [query number, list size, 1]. The labels array is reshaped to [query number, list size]. Then I write several functions to return a function that will convert the feature dictionary and the label list to tensors. In addition, placeholders are created for the inputs so that the documents are grouped by queries and the data will be feed to the scoring function query by query.

2. Transform the dataset to dense tensors.

I borrowed the codes from the libsvm example to write a transformation function to return a function that converts features to dense tensors.

3. Build the scoring function

For this part I borrowed some codes from the demo tutorial for Tensorflow-Ranking, but I incorporated the hyperparameters as variables in the function so that we can tune them later. The scoring function defined here is a one-layer feedforward neuro networks.

4. Build the metrics function

I selected NDCG10, NDCG20, and NDCG100 as the evaluating metrics.

5. Combine the functions

I borrowed some codes from the libsvm example but made some adaption.

6. Put the dataset into the model for training and evaluation

For this part, I made some adaption. I build a simple hyperparameter tuning and cross validation function by myself. Tensorflow does not seem to have a built-in tuning and cross validation function.

7. make estimation on the test data.

Train the model with the optimized hyperparameters and estimate the test dataset.

LIBRARIES USED

Tensorflow==2.6.0
Tensorflow_ranking==0.4.2
Pandas==1.3.3
Numpy==1.19.5
Scikit-learn==1.0

REFERENCES (if any)

Ai, Qingyao et al. “Learning Groupwise Multivariate Scoring Functions Using Deep Neural Networks.” Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval. ACM, 2019. 85–92. Web.

Tensorflow Ranking. (2021). Tutorial: TF-Ranking for sparse features. https://github.com/tensorflow/ranking/blob/master/tensorflow_ranking/examples/handling_sparse_features.ipynb

Tensorflow Ranking. (2021). tf_ranking_libsvm. https://github.com/tensorflow/ranking/blob/master/tensorflow_ranking/examples/tf_ranking_libsvm.py