Information Retrieval and Data Mining Coursework 2

Jiachen Cai
22204524
ucapjc2@ucl.ac.uk

ABSTRACT

This report is to develop different information retrieval models that solves the problem of passage retrieval for COMP0084 assignment 2. In this assignment, three different information retrieval models (logistic regression, LambdaMart, and Neural Network) are developed.

1 INTRODUCTION

1.1 Data

In this assignment, two datasets, train_data, and validation_data are used. These two datasets are both performed as <qid, pid, query, passage, relevancy>. The training data should be selected to train the information retrieval models and the validation data should be selected to evaluate the performance of the models.

1.2 Average Precision

Average Precision is a metric that evaluates the performance of the information retrieval model. The average precision equation could be displayed as follow:

$$AP = \frac{\sum_{k=1}^{n} P(K) * rel(k)}{N}$$

Where $rel(k)$ is the relevancy of the kth document. When the document is relevant the value equals 1 and zero otherwise. $P(K)$ is the precision at cut-off k for the retrieval list and N is the total number of relevant documents. What's more, the value is based on relevant documents at cut-off k retrieved documents. In this assignment, the mean Average Precision is used as a metric for evaluating retrieval models, which get the average of the Average Precision for all queries.

1.3 Normalized Discounted Cumulative Gain (NDCG)

The NDCG is another metric that evaluates the performance of the information retrieval model. The NDCG equation could be displayed as follow:

$$NDCG = \frac{DCG}{optDCG}$$

Where the DCG equation could be displayed as follow:

$$DCG = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log(1+i)}$$

Where the rel is the relevancy of the ith document to the query. For the NDCG, the denominator of the equation indicates the best possible DCG result (the perfect ranking). Compared to the DCG, the value of NDCG would not become large and the value of NDCG is between 0 and 1.

1.4 Text processing

In this assignment, the method of text processing is the same as the method of the first assignment. In this section, the stop words in the dataset would not be removed, the string in documents would be converted to lowercase letters. After that, the punctuation in the documents would be removed. The preprocessed data would be stored in list format. The lemmatization and stemming steps would not be selected in this task, which would be hidden in the source code.

2 TASKS

2.1 Task1

Implement methods to compute the average precision and NDCG metrics. Compute the performance of using BM25 as the retrieval model on the validation data using these metrics.

BM25

In this section, the BM25 retrieval model is implemented with $k_1 = 1.2$, $k_2 = 100$, $b = 0.75$, which is the same as the implementation in assignment 1. To evaluate the model, the functions of getting NDCG and AP are implemented. Firstly, two dictionaries, *rel_ dic* and *non_rel_dic* are generated to record the relevant <qid pid >combination. The dictionaries are represented as <qid: pid: index>, where<index >is the index location in the raw dataset. For each qid, the BM25 model retrieval the first 100 relevant pid then return the rank as a dictionary.

As a result, the mAP and mNDCG of the BM25 model are 0.233 and 0.362

2.2 Task2

Represent passages and query based on a word embedding method. Compute query/passage embeddings by averaging embeddings of all the words in that query/passage.

token embedding and subsample

In this section, the Word2Vec method is used to generate word embedding for the information models. Firstly, the train dataset is subsampled due to the imbalance between the positive and negative rows. The length of the train dataset is decreased to about 100,000 rows. Then the new train dataset is split into passages and queries are stored as a list for processing. During the processing part, the new train dataset and validation dataset are divided into passages and queries. Before getting the embeddings, the processed passages and queries are stored in the text file as the input of the Word2Vec function.

After generating the word embeddings, the word embeddings of the passages and queries are averaged to generate the sentence embedding.

Logistic Regression
In this section, the logistic Regression is implemented. First of all, the weight is initialized according to the shape of the sentence embeddings. What's more, an additional column is added to the weights as the intersection of the sigmoid function. The conbination of the query and passage would be the input of the model.

Training of Logistic Regression
In the training section, the binary cross-entropy function is implemented and the stochastic gradient descent is used to optimized the loss function. For the loss function, five different learning rates are used to discuss the effect of the learning rate, which would be displayed in next section.

Results of Logistic Regression
The loss while training with reasonable learning rates for 3000 epochs could be displayed by figure 1. According to the results, when the learning rate equals to 0.01 and 0.001, the loss function would converge in 3000 epochs. However, the loss function for learning rate euqal to 0.0001 and it would converge with a larger epochs

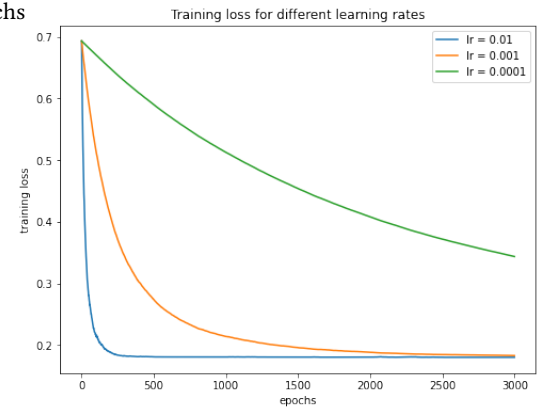

Figure 1: loss of Logistic Regression of reasonable lr

The loss while training with a learning rate of 0.5 could be displayed by figure 2.

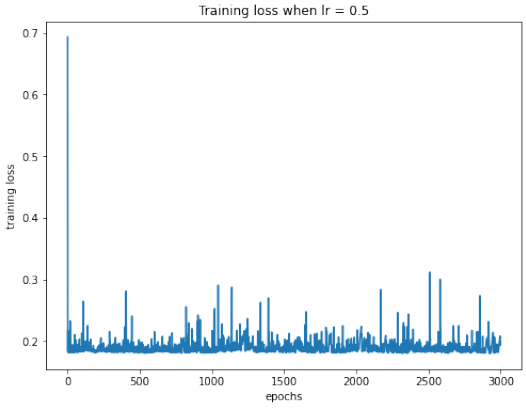

Figure 2: loss of Logistic Regression with lr = 0.5

According to the result, it is obvious that the loss function would not converge due to a large learning rate. After about 100 epochs, the loss starts to fluctuate.

The loss while training with a learning rate of 1 could be displayed by figure 3.

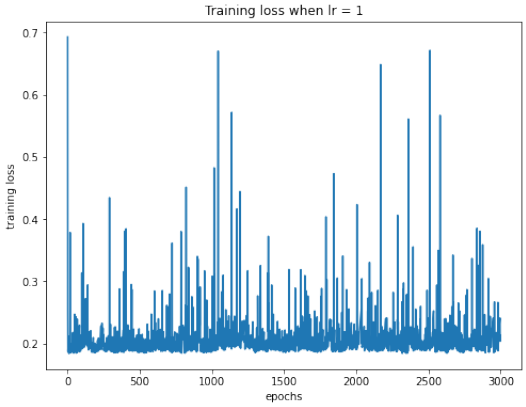

Figure 3: loss of Logistic Regression with lr = 1

According to the result, the fluctuation of the loss function get worse when the learning rate increases.
To evaluate the model, a learning rate of 0.001 is selected and the mAP is 0.0094 and mNDCG is 0.0281

2.3 Task3

Use the LambdaMART learning to rank alogrithm from XGBoost gradient boosting library to learn a model that can re-rank passages.

LambdaMART
In this section, the LambdaMART would be implemented with XGBoost library. It is the combination of the MART and LambdaRank methods. For the data processing section, the method is same as the method in task 2. The XGBRanker function is selected as the model.

Hyper-parameter tuning
In this section, three static lists which contains different parameters would be stored. There are in total 36 models are generated and evaluated these models with mAP and mNDCG. As a result, the 12th model in this experiment is selected with a mAP of 0.0131 and a mNDCG of 0.0326.

2.4 Task4

Using the same training data representation from the previous question, build a neural network based model that can re-rank passages. You may use existing packages, namely Tensorflow or PyTorch in this subtask.

data processing
In this section, the data processing is different from the previous

sections. To generate word embedding for neural network, the Standford's GloVe 100d word embeddings is used to create a index dictionary according to the words in the train dataset and validation dataset. Then the index of the sentence works as the input of the CNN model. What's more, to make sure the shape of the input, a padding parameter of 200 is selected, if the sentence is less than 200, the rest of the index list would be filled with 0.

Text Convolution Neural Network

The neural network I selected for this section is the Convolution Neural Network(CNN)because the CNN model could extract the information of the sentence by convolution and pooling. Compared with the typical CNN, the hidden layers remains the same. However, the input layer and the output layer are quite different. For the input layer, the input index of the sentence should connect to a layer which euqal to the size of the total vocabulary table. In this section, the total number of the words is N and the selected embedding length is M, then the size of embedding should be $(N, 2 * M)$ because of the concatenatenation of the query and passage.
For the output layer, I use the single linear layer with sigmoid function to map the output to a reasonable range.
As a result, the mAP of the text CNN is 0.011 and the mNDCG of the text CNN is 0.032.