COMP0084 CW2

1 Task 1

Question:

Evaluating Retrieval Quality. Implement methods to compute the average precision and NDCG metrics. Compute the performance of using BM25 as the retrieval model on the validation data (validation_data.tsv) using these metrics.

In this part, I use the BM25 algorithm to rank pairs of queries and passages. First of all, I load the files needed and process those datasets such as tokenization, after that, I create an inverted index to help improve the speed of the following code, which is used to calculate the TF-IDF scores. Finally, in order to evaluate the performance of the BM25 algorithm, I use average precision (AP) and normalized discounted cumulative gain (NDCG) as metrics.

AP is a commonly used evaluation metric, which measures how well the ranked passages retrieved from a given query match the truth. To calculate the AP for the BM25 model, I compute the precision at the top 100 and top 10 positions for each query in the ranked list, where precision is the number of relevant passages retrieved at the given positions divided by the total number of relevant passages. After that, I calculate the average AP, which is the sum of the AP for each query, and then divided it by the number of queries.

NDCG measures the relevance of the corresponding passages for a given query but also based on their ranking order. It considers both the relevance of the ranked passages and their position in the ranked list. To calculate NDCG, I need DCG first since NDCG is DCG divided by the optimal DCG for each query. First of all, I assign a weight to each passage in the ranked list based on its rank in the list, which is $\frac{1}{\log 2(i+1)}$, where i is the rank of the passages. After that, I calculate the DCG by summing up the relevance of the passages and multiplied by their weights. For the optimal DCG, I do the same calculation but only on the rank list of relevant passages. Then I calculate the average NDCG using the same step as average AP.

After calculating those metrics, I get the following results: when I only consider the pairs of queries and passages with top 100 scores, the value of the average AP is 0.2356 and the value of the average normalized discounted cumulative gain (NDCG) is 0.3548, which means the model performs well but still has room for improvement. In addition, when we only consider the pairs of queries and passages with top 10 scores, the value of the average AP is 0.2239 and the value of the average normalized discounted cumulative gain (NDCG) is 0.2860. In other words, an average precision of 0.2356 means that the model is only able to retrieve relevant documents with a possibility 23.5%. and the value of NDCG is 0.3548 indicating that the model is able to accurately predict the relevance of passages approximately 35.48% of the time.

2 Task 2

Question:

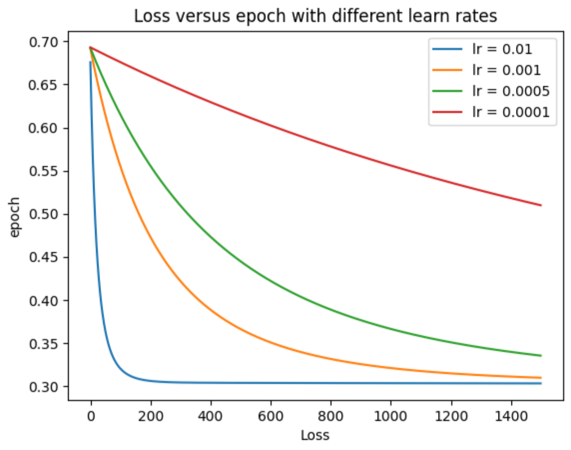
Represent passages and query based on a word embedding method, (such as Word2Vec, GloVe, FastText, or ELMo). Compute query (/passage) embeddings by averaging embeddings of all the words in that query (/passage). With these query and passage embeddings as input, implement a logistic regression model to assess relevance of a passage to a given query. Describe how you perform input processing & representation or features used. Using the metrics you have implemented in the previous part, report the performance of your model based on the validation data. Analyze the effect of the learning rate on the model training loss.

In this part, first of all, I sub-sample the training data using negative sampling, which can reduce the amount of training data required to train our word2vec model, which can also make the model more efficient and faster. There are several steps to apply negative sampling. In the beginning, I randomly select a fixed amount of negative samples for each positive sample, which can reduce the total number of samples needed for training. Furthermore, since our training dataset is quite unbalanced, where the number of relevant passages is much less than the number of irrelevant passages for each query, I set the ratio of two kinds of passages to 10:1.

Secondly, for the implementation of the model, I choose Word2Vec as the embedding method. In order to calculate the embedding of the datasets, I build corpora for both queries and passages using the same processing steps as task1, which are used to train the word2vec model. The first step is to tokenize the queries and passages into tokens, which means breaking up the text into a list of words, where each word is treated as a separate token. After that, I process those tokens to remove unnecessary information such as punctuation and stop words. Furthermore, I train the Word2Vec models using the training data (corpora) and use the default architecture and hyperparameters for the models but set sg=1 which means using the skip-gram algorithm. Finally, I use the trained model to generate embeddings for the queries and passages in both the training set and validation set, where the dimension of the xTr is 200 and yTr (label or relevancy) is 1.

After that, I build a Logistic Regression model and used these embeddings as input to train the Logistic Regression model. The training process involves feeding the model with pairs of passages and queries embeddings and adjusting the weights of the model to minimize the prediction loss. And using the sigma function to predict the labels with the adjusted weight.

Finally, I use average precision (AP) and normalized discounted cumulative gain (NDCG) as metrics to evaluate the performance of the Logistic Regression model on the validation dataset. And then analyzing the training loss with different learning rates, here is a figure displaying the relationship between them as follows.



And for the performance of the model, when we only consider the pairs of queries and passages with top100 scores, the value of the average precision (AP) is 0.01039 and the value of the normalized discounted cumulative gain (NDCG) is 0.0309 which is quite worse than BM25 model in the previous part.

3 Task 3

Question:

Use the LambdaMART learning to rank algorithm from XGBoost gradient boosting library to learn a model that can re-rank passages. You can command XGBoost to use LambdaMART algorithm for ranking by setting the appropriate value to the objective parameter as described in the documentation. You are expected to carry out hyper-parameter tuning in this task and describe the methodology used in deriving the best performing model. Using the metrics you have implemented in the first part, report the performance of your model on the validation data. Describe how you perform input processing, as well the representation/features used as input.

In this part, first of all, I used the functions from task2 like split_data to load and convert the data in the training set into query and passage embedding vectors in order to feed the model, and also process the validation set in order to evaluate the trained model using the same metrics AP and NDCG as before.

Secondly, I create two lists of parameters generate all the possible combinations of hyperparameters such as learning rate and depth. And then generating different XGBoost models with each combination of hyperparameters and training them using the training datasets. After that, for each model, storing the predicted scores for the validation data.

In order to find the model with the best performance, in other words, which has the best average AP and mNDCG, I use the function eval_model which takes the validation data and the predicted scores as inputs to calculate those metrics. And I get the following result which is when we only consider the pairs of queries and passages with top 100 scores, the learning rate and the maximum depth for the best model are 0.01 and 6. In addition, the corresponding average AP and mNDCG are 0.01368 and 0.03613, which means the performance of the trained XGBoost model has better performance than the LR model in task 2 but the model still needs improvement.

Finally, I follow the same step as task 2 to perform the model we selected to the file "candidate_passages_top1000.tsv" to predict the scores for the top 100 scores of passages for each query in test_queries.tsv file, and store the scores and all the data needed in the file LM.txt.

4 Task 4

Question:

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. Justify your choice by describing why you chose a particular architecture and how it fits to our problem. You are allowed to use different types of neural network architectures (e.g. feed forward, convolutional, recurrent and/or transformer based neural networks) for this part. Using the metrics you have implemented in the first part, report the performance of your model on the validation data. Describe how you perform input processing, as well as the representation/features used.

In this part, I implement a neural network model to rank passages based on their relevance to given queries. Firstly, I load the preprocessed training and validation dataset from the files "test2.pickle" and "val_data.pickle" which are generated in task 2.

After that, I generate the neural network model using keras from TensorFlow, where I add a LSTM layer. The reason is that LSTM layer can effectively find the dependencies (even temporal) in the training data. And I set this layer with 20 units, this is because large number of LSTM units can help the model find more complex patterns in the training dataset. And I choose "tanh" as the activation function which works better on our model. In addition, I use L2 regularization in the model which can help avoid overfitting that improve the performance of the model. Furthermore, I add a dense output layer and then compile the model using the mean squared error function and choose rmsprop as the optimizer which is used to adjust the learning rate of weights in each iteration. After that, I train the model and predict the validation data to evaluate the performance of the model. As a result, when we only consider the pairs of queries and passages with top 100 scores, the value of the average precision (AP) is 0.01177 and the value of the normalized discounted cumulative gain (NDCG) is 0.03124 which is quite similar to the model in the previous part.