IRDM Course Project Part II

1 Task One: evaluating retrieval quality

To evaluate the retrieval quality, Mean average precision (MAP) and normalised Discounted Cumulative Gain (NDCG) were implemented. MAP is the mean average precision scores across the queries, where precision is the proportion of relevant document in the retrieved documents and rewards models that retrieve relevant results at higher ranks. NDCG considers relevance and the position of the retrieved relevant documents in the ranked list. A logarithmic discount was applied to penalise relevant documents appearing lower in the ranks. These evaluation metrics were applied on the validation data using a BM25 retrieval model set as a baseline model.

Metric Implementation: These metrics will be used for evaluating different models later, the average precision for a ranked list was calculated by adding the total precision at each position for relevant document and dividing by the number of documents considered, the function iterates through the ranked list, accumulates the precision values for relevant items to compute the final average. To implement the NDCG, a standard formula for Discounted Cumulative Gain was used, this was normalised by dividing by the ideal DCG which was computed from the ground truth relevance scores sorted in descending order. For Evaluation, K=10 was used (I.e. NDCG@10). For the ranked list, BM25 ranking function was used. The queries and passages were tokenised, and each 1000 candidate passages were scored.

Results: The MAP result was 0.1183, this means only 11.8% of the retrieved documents in the top ranks were relevant across queries, suggesting that the relevant documents are not consistently at the top of the list for each query and so further improvement is needed. NDCG@10 result was 0.1365. This helps to compare how similar the model's ranking order is compared to the ideal order of relevance, the low value suggests that BM25 often ranks highly relevant documents lower than not relevant documents. These is values will be the benchmark to compare any relevant improvements from implementing different retrieval models.

2 Task 2: Logistic Regression

In this task, a logistic regression model was implemented from scratch. The input features were constructed using GloVe, this is a pre-trained word embedding model that maps words into dense vector representations. I have used a 300-dimension word embeddings file of GloVe for this task. (For the following tasks, I have used "glove.6B.300d.txt" for vector embeddings) All queries and passages were initially lowercased and had punctuation removed using regular expressions, along with tokenization. Each word was then embedded with the corresponding 300-dimensional GloVe embedding. Then, For Each query and passage, the mean vector across all the word embeddings was computed. The 300-dimensional query and passage embedding for each sample was concatenated, giving a 600-dimentional vector for each sample. This final vector captures the semantics of both the query and the passage while maintaining simplicity and interpretability. The logistic regression model was implemented with a sigmoid activation. For each query-passage pair, the model outputs a probability score in the range of 0 and 1 which is the predicted relevance. A Binary Cross-Entropy loss was used for training along with Adam optimiser which adapts the learning rates for each parameter.

The data is seen to have a large imbalance between the relevant and non-relevant passages. In training data, there are total of 4,797 positive sample (I.e. relevance >0) and 4,359,542 negative samples (I.e. passages with relevance =0). This gives a positive-to-negative ratio of 0.0011, making this an extremely imbalanced dataset. This could affect the performance because the model may be biased towards predicting a passage to be non-relevant (i.e. the majority class), this leads to a poor recall for relevant passages, which lowers MAP and NDCG scores. Hence, negative sampling was used to balance the training data, while keeping the original dataset unchanged for unbiased evaluation.

After experimenting with different negative sampling ratios, a ratio of 4:1 was selected. This means that four non-relevant passages were included at random for every relevant passage. This resulted in the training set size of 23975 samples. Although this is a significant reduction in size compared to the full training dataset, this is acceptable since all the relevant passages are included. Also, increasing the sampling ratio would lead to more non-relevant passages being included, potentially overwhelming the model with non-relevant passages and hindering its ability to learn meaningful relevance patterns. Furthermore, the sampled training data was also shuffled after carrying out negative sampling.

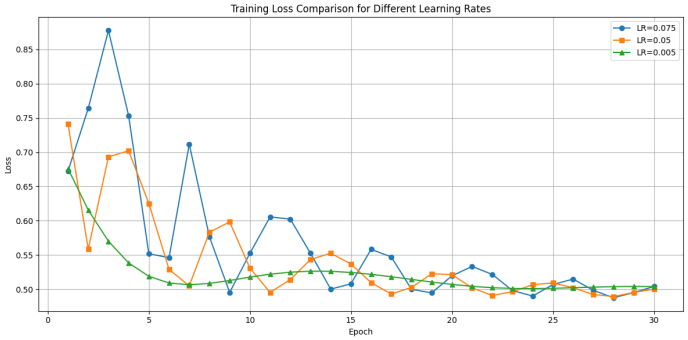

Figure 1: Graph training loss over epoch=30 for different learning rates.

The learning rate (LR) value and the number of epochs used were tuned to get the optimal performance. Figure 1 shows training loss recorded over epochs for various learning rates. A low learning rate results in a slow convergence and higher final loss, a high learning rate results in instability and noisy losses. LR=0.075 appears to be very unstable, even though it becomes more stable over epochs, it has the biggest spikes to the very end of training. LR=0.05 exhibited smoother convergence with a lower final loss compared to higher LR values. This shows that learning rate tuning plays an important role for simple models like logistic regression, especially with imbalanced data and noisy inputs.

Results: The Evaluation of this model was performed using the custom MAP and NDCG@10 metrics as described in task 1, with the original validation dataset without sampling. The optimal LR is 0.05 for this model, giving a performance of: MAP=0.0200 and NCGD@10=0.0161. Following this, 'LR.txt' file is generated which contains the ranked passages for each query along with its relevance score.

3 Task 3: LambdaMART

This task implements a passage re-ranking model using the LambdaMART algorithm from the XGBoost library. The model has been trained on query-passage pairs to predict the ranking scores used to reorder the candidate passages.

Model Implementation: The query and passages are tokenised, punctuation removed to compute the average embedding for all tokens in the text. This gives a final input feature for a query-passage pair of 600-dimensional vector, created by concatenating the query and passage embeddings.

To balance the data, negative sampling was applied during training with negative-to-positive ratio of 4:1, The LambdaMART model was trained using the rank:ndcg objective in XGBoost, this is optimised especially for ranking tasks. LambdaMART combines gradient boosted decision trees with the LambdaRank loss function to optimise ranking metrics like NDCG. For this model, the same 300-dimensional pre-trained embeddings (GloVe) were used like task 3 and 4 to maintain consistency in embeddings across models.

During training, each query's associated passages were grouped together using query IDs, this ensures the model learns to rank the documents for each query group, this is required by LambdaMART's pairwise ranking objective. The hyperparameter tuning was performed using random search, where several combinations of learning rate, tree depth, regularisation parameters (min_child_weight, gamma) were evaluated. Each combination was trained for 100 rounds and were then evaluated using the custom MAP and NDCG metrics on the validation dataset. The best model was then used to predict the relevance score for each query-passage pair.

Results: The best hyperparameter values are {'eta': 0.05, 'max_depth': 4, 'min_child_weight': 0.5, 'gamma': 0.5} with Mean Average Precision (MAP): 0.0307 and Mean NDCG@10: 0.0317 This demonstrates that the LambdaMART model improved on the logistic regression model and is slightly better at re-ranking candidate passages based on relevance to each query.

4 Task Four: Neural Networks

This task involved building a neural network model using PyTorch to re-rank passages for given set of queries, using the same input features and data representations as in Task 3. A network was trained to predict the relevance scores which helps to reorder the candidate passages effectively. Also, the model is trained with the same sampled training data and the full validation data to ensure a consistent comparison with the LamdaMART model from task 3.

Each query and passage are pre-processed by lowercasing and removing punctuation, pre-trained GloVe embeddings (300 dimensions) were used to convert the tokens to vectors. The average for all token embeddings is calculated. For each query-passage pair, the two embeddings were concatenated to form a 600-dimensional input feature vector. Negative sampling was also applied to address the data imbalance with a 4:1 negative to positive ratio.

For this task, a fully connected feed forward neural network (I.e. multilayer perceptron) is chosen. This is because the input features are fixed-size dense vectors, in our case, 600-dimensions, this makes feedforward networks ideal for this task. Furthermore, the mode is trained to regress relevance scores, suitable for pairwise or pointwise ranking. Also, the architecture balances expressiveness and training efficiency without overfitting.

For training, mean squared error was used along with Adam optimiser with learning rate of 0.001, epochs=30 and batch size=32. The model is evaluated using the custom MAP and NDCG@10 metrics. These are computed per query ground truth on the validation set. These hyperparameter values were chosen based on systematic experimentation and empirical evaluation to obtain the best model performance.

Results: On the validation data the neural network achieved Mean Average Precision (MAP): 0.0356 and Mean NDCG@10: 0.0367, this is slightly above LamdaMART's performance, suggesting that the neural networks are slightly better at ranking.

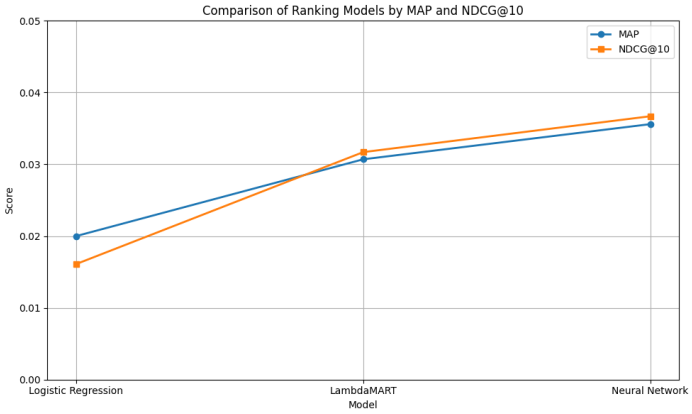

Figure 2: Shows the metrics of MAP and NDCG@10 for the three models.

Figure 2. Shows the performance on the validation data for the three models, Both MAP and NDCG@10 increase consistently for the three models. This shows clear improvement as the model complexity and representational power increase. Logistic regression is useful as a baseline but lacks power for learning ranking. LambdaMART is seen to provide a strong and interpretable performance. Neural networks are shown to offer the best ranking performance for this task when properly tuned and are also more scalable with model size and data.