Information Retrieval and Data Mining (COMP0084) Coursework 2

ABSTRACT

Coursework 2 for Information Retrieval and Data Mining (COMP0084).

KEYWORDS

Information Retrieval, Neural Networks, LambdaMART

ACM Reference Format:

. 2023. Information Retrieval and Data Mining (COMP0084) Coursework 2. In *Proceedings of (COMP0084)*. ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Information retrieval (IR) is the process of finding relevant information from a collection of unstructured or semi-structured data. Learning to rank (LTR) is a popular sub-field of information retrieval that uses machine learning algorithms to learn how to rank search results based on their relevance to a user's query.

In this coursework, our objective is to create an advanced information retrieval model that addresses the challenge of passage retrieval. Specifically, we aim to develop a model that can efficiently and accurately provide a prioritized list of passages that are relevant to a given query. Our model should be able to effectively and efficiently return a ranked list of passages relevant to a given query.

2 TASK 1: EVALUATING RETRIEVAL QUALITY

In this task, the **average precision (AP)** matrics and **normalised discounted cumulative gain (NDCG)** metrics will be implemented to evaluate the performance of BM25 model.

As we have already explored the BM25 retrieval model in coursework 1, we will provide a concise summary of its key features. The BM25 scoring function is defined by the following equation.

$$score(p,q) = \sum_{i \in q} \log \frac{1}{(n_i + 0.5)(N - n_i + 0.5)} \cdot \frac{(k_1+1)f_i}{K + f_i} \cdot \frac{(k_2+1)qf_i}{k_2 + qf_i} \tag{1}$$

where $K = k_1((1-b) + b \cdot \frac{dl}{avdl})$. f_i and qf_i indicate the frequency of a specific word that occurs in the passage and query respectively. The values for k_1 k_2 and b are the recommended values, which are 1.2, 100, and 0.75.

COMP0084, March 28, 2023, London, UK

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

2.1 Average Precision (AP)

Average Precision (AP) [4] is a measure of the accuracy and effectiveness of a retrieval system in returning relevant results to a user's query. In other words, it measures how many relevant results are retrieved among the top k results, where k is the number of retrieved passages.

The following equation show how to obtain **Average Precision (AP)** for a given query q_i.

$$AP_{q_i} = \frac{1}{|P_r(q_i)|} \sum_{k=1}^{P_r(q_i)} Precis(k) \cdot rel(k) \tag{2}$$

where $|P_r(q_i)|$ is the total number of retrieved passages for the q_i. $Precis(k)$ is the precision of relevant passages among the top k retrieved passages, and $rel(k)$ is an indicator function that is 1 if the passage at rank k is relevant, and 0 otherwise.

Besides, in order to evaluate the performance across all queries, we use mean average precision (mAP) as the evaluation function. The formula is shown in equation (3).

$$mAP = \frac{1}{|Q|} \sum_{q_i \in Q} AP_{q_i} \tag{3}$$

where $|Q|$ is the total number of queries.

2.2 Normalised Discounted Cumulative Gain (NDCG)

Discounted Cumulative Gain (DCG) [4] is calculated by comparing the predicted rankings of the retrieval system with the *graded relevance* rankings. It assigns a score (*gain*) to the predicted rankings based on the position of the relevant passages in the ranking list. Score (*gain*) is accumulated starting at the top of the ranking and would be discounted at lower ranks.

The formula of DCG with *nonlinear gain function* is given by the equation (4), which shares the same notation as in the equation (2).

$$DCG_{q_i} = \sum_{k=1}^{P_r(q_i)} \frac{2^{rel(k)} - 1}{\log(1+k)} \tag{4}$$

The score of NDCG is normalized by dividing DCG by the optimal ranking score at rank k, which is the maximum achievable score for a given query. Under a similar consideration in section 2.1, we use the mean NDCG to evaluate the performance of a retrieval model.

$$mNDCG = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{DCG_{q_i}}{optDCG_{q_i}} \tag{5}$$

where $optDCG_k$ is the optimal DCG at rank k, which is the maximum achievable DCG_k score for the given query.

Table 1: Performance of BM25 Model

Top k ranking	mAP	mNDCG
$k = 3$	0.1685	0.1906
$k = 10$	0.2025	0.2025
$k = 100$	0.2158	0.3285

2.3 Performance of BM25

The results of BM25 model on the validation data are shown in Table 1.

3 TASK 2: LOGISTIC REGRESSION (LR)

Logistic regression (LR) in information retrieval modeling is a binary classification algorithm that predicts the probability that a passage is relevant to a given query, based on a set of features that describe the passage and the query.

3.1 Input Processing

3.1.1 Word Embedding. **Word2Vec** [5] is chosen as the word embedding technique, which specifically we select Continuous Bag-of-Words (CBOW) model instead of Skip-gram model. The reason is its fast-convergence property and the merit of learning better syntactic relationships between words. In other words, CBOW tends to find the probability of a word occurring in a context. In this task, we use **gensim** [9] to implement Word2Vec, converting each word into (1×100) vector via all unique passages in training data.

After that, the **average term embeddings** [6] [7] are used to compute the centroids of the query and passage term embeddings, respectively, using the following equations.

$$\begin{aligned} \boldsymbol{v_q} &= \frac{1}{|q|} \sum_{w \in q} \frac{\boldsymbol{v_w}}{||v_w||} \\ \boldsymbol{v_p} &= \frac{1}{|p|} \sum_{w \in p} \frac{\boldsymbol{v_w}}{||v_w||} \end{aligned} \tag{6}$$

3.1.2 Feature Representation. Building upon [8], we select a subset of features, consisting of $idf(\boldsymbol{q})$ and $bm25(p, q)$, as well as the L2 distance ($l2dist(\boldsymbol{p}, \boldsymbol{q})$) and cosine similarity ($cosine(\boldsymbol{p}, \boldsymbol{q})$) between $\boldsymbol{v_q}$ and $\boldsymbol{v_p}$. In addition, we also incorporate the query and passage vectors themselves as features. We can then define the feature mapping function $\phi(\cdot)$ from passage p to feature space as follows.

$$\phi(\boldsymbol{p}, \boldsymbol{q}) = [idf(\boldsymbol{q}), bm25(\boldsymbol{p}, \boldsymbol{q}), l2dist(\boldsymbol{p}, \boldsymbol{q}), cosine(\boldsymbol{p}, \boldsymbol{q}), \boldsymbol{v_q}, \boldsymbol{v_p}] \tag{7}$$

3.1.3 Downsampling. By the statistics of the training data, we can find the imbalanced distribution between relevant retrievals ($\leq 0.1\%$) and irrelevant retrievals ($\geq 99.9\%$). Hence, the technique of downsampling is applied so as to lower the influence of the major data and prevent the hazard of overfitting. In this task, we only use the whole relevant retrievals and partial irrelevant retrievals $(60,000$ entries) for each query in training data.

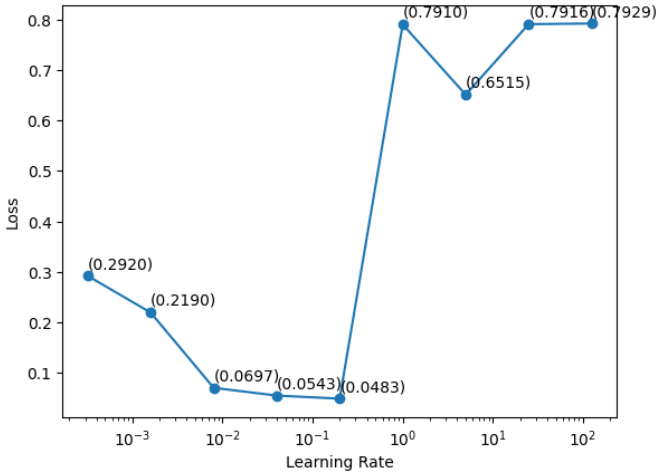

Figure 1: Learning Rate v.s. Training Loss

3.2 Model Implementation

The LR model learns to assign a score to each document in the collection, based on its similarity to the query and its relevance to the topic of interest. The model implementation is via Pytorch, and its scoring function is defined by equation (8), and loss function is a binary cross-entropy function.

$$score(p, q) = \frac{1}{1 + exp(-(\boldsymbol{w}^T \phi(\boldsymbol{p}, \boldsymbol{q}) + \boldsymbol{b}))} \tag{8}$$

where $\boldsymbol{w}$ and $\boldsymbol{b}$ are the parameters of weights and bias.

3.2.1 Model Training. In this section, we let the number of epochs be 1000 and simulate the training process with different learning rates. The result in figure 1 shows that the learning rate has a significant impact on the model. While a lower learning rate might cause the model hard to converge, a higher learning rate might also incur gradient explosion, pushing the model far from the optimal state.

3.3 Performance of LR Model

The results of LR model on the validation data are shown in Table 2.

Table 2: Performance of LR Model

Top k ranking	mAP	mNDCG
$k = 3$	0.1771	0.1991
$k = 10$	0.2110	0.2694
$k = 100$	0.2244	0.3348

4 TASK 3: LAMBDAMART MODEL (LM)

LambdaMART (LM) is a gradient-boosting framework designed for ranking problems in information retrieval, which aims to directly optimize the NDCG metric, which measures the quality of a ranked

list of passages. In this task, we use the sampling method as in task 3.

For feature space, besides the features used in 3.1.2, we include tfidf score ($tfidf(p,q)$), and the count of similar words between query and passage ($c(p,q)$).

$$\begin{aligned}\phi(\boldsymbol{p},\boldsymbol{q}) = [&idf(\boldsymbol{q}), bm25(\boldsymbol{p},\boldsymbol{q}), l2dist(\boldsymbol{p},\boldsymbol{q}),\\ &cosine(\boldsymbol{p},\boldsymbol{q}), tfidf(\boldsymbol{p},\boldsymbol{q}), c(\boldsymbol{p},\boldsymbol{q}), \boldsymbol{v_q}, \boldsymbol{v_p}]\end{aligned} \tag{9}$$

In this task, XGBoost[3] library is utilized to implement LM. by setting parameter *objective* to `rank:ndcg`, we can start training with listwise objective and directly optimise and improve LM with gradient λ_{ij} [2], which is defined by equation (10).

$$\lambda_{ij} = \frac{-\sigma|\Delta NDCG_{ij}|}{1 + exp(\sigma(s_i - s_j))} \tag{10}$$

where $\Delta NDCG_{ij}$ means the difference in terms of NDCG value of the retrieved passages, p_i and p_j, for a given query. s_i and s_j indicate the score obtained by the LM model for p_i and p_j.

4.1 Hyperparameter Tuning

In this section, we leverage **RandomizedSearchCV** in sklearn to find the best parameter setting in the following grid.

$$\begin{aligned}&learning\_rate = [0.0001, 0.001, 0.01, 0.1]\\ &subsample = [0.5, 0.75, 1.0]\\ &colsample\_bytree = [0.6, 0.8, 1.0]\\ &max\_depth = [2, 4, 6, 8]\end{aligned} \tag{11}$$

After the simulation with setting the number of epochs to 1000, the best parameter setting is $learning\_rate = 0.1$, $subsample = 0.75$, $colsample\_bytree = 0.8$, and $max\_depth = 8$.

4.2 Performance of LM Model

The results of LM model on the validation data are shown in Table 3.

Table 3: Performance of LM Model

Top k ranking	mAP	mNDCG
$k = 3$	0.1799	0.2022
$k = 10$	0.2182	0.2811
$k = 100$	0.2320	0.3511

5 TASK 4: NEURAL NETWORK MODEL (NN)

In this section, we will leverage the same sampling methods and features in section 4 task 3, and classify the features into the following 3 categories, $\phi_{ltr}(\cdot)$ and $\phi_{vec}(\cdot)$.

$$\begin{aligned}&\phi_{ltr}(\boldsymbol{p},\boldsymbol{q}) =\\ &[idf(\boldsymbol{q}), bm25(\boldsymbol{p},\boldsymbol{q}), l2dist(\boldsymbol{p},\boldsymbol{q}), cosine(\boldsymbol{p},\boldsymbol{q}), tfidf(\boldsymbol{p},\boldsymbol{q}), c(\boldsymbol{p},\boldsymbol{q})]\\ &\phi_{vec}(\boldsymbol{q}) = \boldsymbol{v_q}\\ &\phi_{vec}(\boldsymbol{p}) = \boldsymbol{v_p}\end{aligned} \tag{12}$$

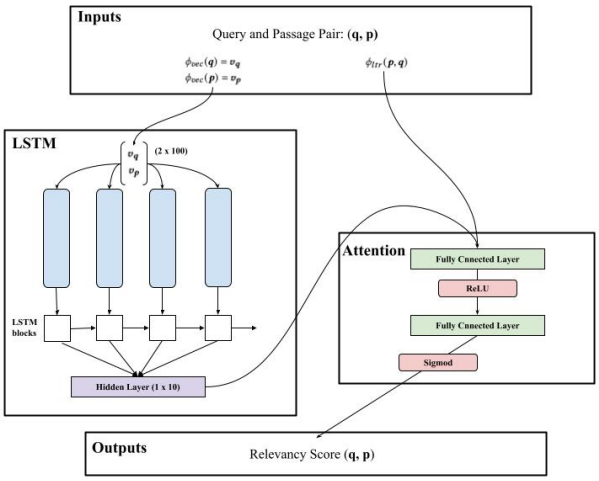

Figure 2: The architecture of the NN Model.

where $\phi_{ltr}(\cdot)$ can generate traditional features for learning-to-rank while $\phi_{vec}(\cdot)$ can output average word embedding vector with $size = (1 \times 100)$ for queries and passages.

The structure is shown in Figure 2, which is inspired by [1]. The purpose of using LSTM model is to capture the sequential dependencies between terms, which could potentially improve the understanding of relevancy between queries and passages.

In the first block of Figure 2, we transform query and passage pairs into feature vectors with $\phi_{ltr}(\cdot)$ and $\phi_{vec}(\cdot)$. Later, in the LSTM block, $\boldsymbol{v_q}$ and $\boldsymbol{v_p}$ are concatenated vertically into (2×100) matrix, and forward pass through LSTM blocks so as to generate hidden vectors with size = (1×10).

In the attention block, we combine outputs from LSTM block and $\phi_{ltr}(\cdot)$ to generate new vectors to feed in 2-layer NN model, and finally output the relevancy score for a given $(\boldsymbol{p},\boldsymbol{q})$. With optimizer **Adam**, we leverage the similar hyperparameter tuning method in section 4 to complete the training process.

5.1 Performance of NN Model

The results of NN model on the validation data are shown in Table 4.

Table 4: Performance of NN Model

Top k ranking	mAP	mNDCG
$k = 3$	0.1989	0.2201
$k = 10$	0.2365	0.2977
$k = 100$	0.2505	0.3668

REFERENCES

[1] Rayan Abri, Sara Abri, and Salih Cetin. 2022. Providing A Topic-Based LSTM Model to Re-Rank Search Results. In *2022 7th International Conference on Machine Learning Technologies (ICMLT)*. 249–254.
[2] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[4] Christopher D Manning. 2009. *An introduction to information retrieval*. Cambridge university press.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26 (2013).

[6] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. 2016. A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137* (2016).

[7] Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*. 83–84.

[8] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13 (2010), 346–374.

[9] Radim Rehurek and Petr Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3, 2 (2011).