University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	Information Retrieval M			
Coursework Number	2			
Deadline	Time:	04:30 pm	Date:	15th March 2023
% Contribution to final course mark	10%			
Solo or Group ✓	Solo	✓	Group	
Anticipated Hours	~10 hours			
Submission Instructions	See details in assignment.			
Please Note: This Coursework cannot be Re-Done				

Information Retrieval M
Exercise 2
February 2023

Introduction

Learning-to-rank (LTR) is a recent machine learning paradigm used by commercial search engines to improve retrieval effectiveness by combining different sources of evidence (aka features). The key point of learning-to-rank is that it is easy to incorporate new features and to leverage the amount of potential training data available to Web search engines. In this exercise, you will be experimenting with learning-to-rank using a number of provided features, and implementing two additional features of your own. In particular, you will be implementing, testing and evaluating a query independent feature (URL length, taking into account the length of the URL of a given document) and a query dependent feature (a proximity feature, which aims to boost the scores of documents where the query terms appear in close proximity). Similar to Exercise 1, you will then be reporting the obtained results and analysing the effectiveness of the IR system, including the effectiveness of your new features, but *only* using the homepage finding topic set ("hp"). Effectiveness will be measured using two metrics: Mean Average Precision (MAP) and Precision at Rank 5 (P@5). This exercise assumes that you are familiar with both the warm-up exercise and Exercise 1.

Exercise Specification

For retrieval, for all of Exercise 2, you will use PyTerrier's support for learning-to-rank (LTR) and the state-of-the-art LightGBM LambdaMART LTR technique. In particular, the online documentation about LTR available at https://pyterrier.readthedocs.io/en/latest/ltr.html will be an **essential** guide on how to deploy and adapt learning-to-rank.

For your experiments, you should use the new provided index – this has more evidence in the index, namely it provides the document URLs as metadata, and contains fields and "block" positions, so as to have sufficient information to support the implementation of your two new features.

In conducting your experiments, you must **only** use the homepage finding topic set ("hp") for evaluation. The test set is the same set of queries you used in Exercise 1. However, to ensure a fair experimental setting, you will need to use the provided *separate* "training" and "validation" topic sets to deploy learning-to-rank.

For applying LTR, you first need to identify the candidate set of documents to re-rank. For this necessary step, you need to use the PL2 weighting model. Then you need to deploy and evaluate a *baseline LTR approach*, which uses a total of 3 features, namely the PL2 scores (candidate set to re-rank), plus the BM25 scores on titles, and the PageRank scores, following the instructions in the Colab/Jupyter Exercise 2 notebook. Your sample is then re-ranked using these three features combined using a LambdaMART learned model.

Q1. In the Quiz, report the effectiveness performances of the two system configurations: LTR (using PL2 to generate the initial candidate set) vs. PL2 (baseline). **Report the retrieval performances rounded to 4 decimals.**

Use the PyTerrier's t-test statistical significance testing support to conclude if the performances of LTR are significantly better than that of PL2 using either (i) MAP or (ii) P@5 *(i.e. you will conduct 2 statistical tests in Q1)*. In particular, use the p-values obtained in your experiment, to draw your conclusions as to whether the deployed LTR system (which re-ranks the PL2 sample) is better by a statistically significant margin than PL2 on either or both used effectiveness metrics. Now answer the questions in the Quiz instance.

[6]

Q2. You now need to implement an additional *query independent* feature, namely a URL length feature that will score the documents according to the length of their associated URL string. There are different ways to compute such scores. You will implement two instantiations of the URL length feature, and determine which instantiation leads to the best improvements in retrieval effectiveness in comparison to the LTR *baseline* system (i.e. your deployed LTR model with 3 features from Q1):

(a) Count the number of slash characters ("/") in the URL.

(b) Categorise the URL of the document into one of these 4 categories – use *ascending* integers to represent these categories:

- root: a domain name, optionally followed by "index.html" (e.g. *http://trec.nist.gov*)
- subroot: a domain name, followed by a single directory, optionally followed by "index.html" (e.g. *http://trec.nist.gov/pubs/)*
- path: a domain name, followed by an arbitrary deep path, but not ending in a file name other than "index.html" (e.g. *http://trec.nist.gov/pubs/trec9/papers/*)
- file: anything ending in a filename other than "index.html" (e.g. *http://trec.nist.giv/pubs/trec9/t9_proceedings.html*)

NB: Subroot and path end with a trailing slash or an /index.html - your solution for *these categories* should **NOT** consider any other variations of filenames (e.g. /default.htm is NOT an alternative for /index.html)

First, following good software engineering practice, you are advised to test your implementation using appropriate assertions on mock URLs/dataframes. Next, in evaluating your new URL length feature instantiations, you should evaluate their effectiveness in two manners: (i) as re-rankers of PL2, and (ii) as features within the learned model. In particular, for **each** of the two URL length feature instantiations:

(i) Use your URL length feature as a re-ranker of the PL2 candidate set (without using LTR, and making use of the PyTerrier >> operator), and evaluate its effectiveness. Now complete the questions in the Quiz. **(Report all your performances rounded to 4 decimal places).**

(ii) Integrate your added feature instantiation into the LTR pipeline and train a new LTR model of 4 features (*NB:* Do **not** refit an older model). Examine the feature importance of the added URL feature. Using the 3-features LTR system as a baseline, evaluate the effectiveness of the new LTR model. Insert your *source code* for both your URL length feature instantiations when prompted (note that a 2-bands penalty will be applied if you do not upload the code you used to answer the Q2 Quiz questions), and complete the questions in the Quiz. **(Report all your performances rounded to 4 decimal places).**

[22]

Q3. You will now implement a new *query dependent* feature, namely a proximity search feature that boosts documents where the query terms occur closely together. You will use the following function to compute the feature:

MinDist(a, b, D): given two query terms *a* and *b*, this function is defined as the minimum distance, in terms of number of tokens, between any occurrences of *a* and *b* in document D. The distance can be calculated from the positions of *a* and *b* in the text of document, which is available in the "body" column of the results dataframe. The intuition of this function is that if any occurrence of *a* is closer to any occurrence of *b*, then this indicates that the two terms are related.

Given that the query often contains more than 2 terms, you should calculate your proximity feature by *aggregating* the MinDist function scores for each document. You should apply AvgMinDist, which takes the mean over the *pairs* of sequential query terms that *match* in a given document (if no pairs match in a document, set the aggregated value to be equal to the maximum possible value for that document, which is the number of tokens in the document). Define a Python function to calculate an AvgMinDist that adheres to the following specification:

```
def avgmindist(query : str, document : str) -> float
```

Your implementation should be case-insensitive and should remove punctuation from documents. However, it should *not* apply stemming or remove stopwords.

As always, following good practices, you should test your implementation using appropriate assertions on mock documents/dataframes. Furthermore, we have identified 9 test cases for the .GOV corpus to test your implementation. Please apply your implemented AvgMinDist function to these test cases and submit the answers on the corresponding Quiz questions.

[11]

Now, integrate your added feature to a new LTR model (PL2 sample). Using the 3-features LTR system as a baseline, report the performance of your LTR model once you add the AvgMinDist proximity feature. Insert the *source code* and an informative description for your AvgMinDist feature when prompted (note that a 2-bands penalty will be applied if you do not upload the code you used to answer the Q3 quiz questions and/or you do not describe your solution in sufficient details) and complete the corresponding Quiz questions. **(Report all performances rounded to 4 decimal places).**

[3]

Q4. You should now experiment with LTR, where you include both your added features (URL type and AvgMinDist) along the 3 initial features (5 features in total). Evaluate the performance of the resulting system in comparison to the LTR Baseline (3 features). **(Report all performances rounded to 4 decimal places).**

Reflect on your conducted experiments thus far, for example if your two added features add up (i.e. the 5-features LTR model is better than either of the 4-features models with LTR Type or AvgMinDist), as well as on the performance of your various implemented features and answer the remaining questions of the Quiz.

[8]

Hand-in Instructions:
All your answers to Exercise 2 must be submitted on the Exercise 2 Quiz instance on Moodle with your completed notebook **(showing both your solutions and the results of the executions of your solutions)**. Please note that a **2-bands penalty** will be applied, if you do not upload your completed notebook or if you do not show all the results (inc. plots) obtained from the execution of your solutions. The submitted notebook will be used to *check* your answers in the Quiz. **Marks can be lost** if the notebook does not show evidence for the reported experimental answers submitted in the Quiz. This exercise is worth 50 marks and 10% of the final course grade.

NB: You can (and should) naturally complete the answers to the quiz over several iterations. However, **please ensure that you save your intermediary work on the Quiz instance by clicking the "Next Page" button every time you make any change in a given page of the quiz and you want it to be saved.**