CAB431 Week 8 Workshop
User Information Needs
SOLUTIONS

An information need can be defined as the underlying cause of the query that a person submits to an IR system. However, in many real applications, a query can be a poor representation of the information need as a query is normally very short.

To solve this problem, people use all the words in the query to find related words, rather than expanding each word individually. An interesting approach is called pseudo-relevance feedback, which performs query expansion based on an IR model and uses the initial query to retrieve top ranked documents (e.g., top-*k* documents or documents' relevance scores > a threshold). Pseudo-relevance feedback just assumes that the top ranked documents are relevant. It requires no other user input.

For example, a user is searching for information pertaining to crimes committed by people who have been previously convicted and later released or paroled from prison. The user may use a short query *Q* = "Convicts, repeat offenders".

Task 1. Ranking documents using the initial query *Q*.

Design a BM25 based IR model which uses query *Q* to rank documents in the "Training_set" folder and save the result into a text file; e.g., BaselineModel_R102.dat; where each row includes the document number and the corresponding BM25 score or ranking (in descendent order).

Formally describe your idea in an algorithm.

PS: We believe everyone in the class should know how to do text pre-processing and calculate BM25 scores; therefore, we don't need to discuss the details of these parts when you describe your ideas in an algorithm.

Solution: Task 1

Algorithm *baseline_bm25*(*Q*, *U*)

Inputs: a query *Q*, a set of documents *U* in the "Traning_set" folder.
Output: a text file BaselineModel_R102.dat; where each row includes the document number and the corresponding ranking score (in descendent order).
Method:

```
Step 1. # Documents' representation
        Let docs be an empty dictionary
        For each xml document d in U:  # text pre-processing
                Get the document_ID
                Find the contents in <text> … </text>:
                        get tokens (or terms) and their frequencies, and
                        append (document_ID, {term:freq, …}) into docs;
Step 2.
```

Calculate each term’s document frequency and save the result in a dictionary *df_*

Step 3.

Use query *Q*, *docs* and *df_* to work out BM25 score for each document in *U*

Step 4

Sort the documents based on their BM25 score and save the result into a file BaselineModel_R102.dat

Task 2. Training set generation

Pseudo-Relevance Hypothesis: “top-ranked documents (e.g., documents’ BM25 scores are great than 1.00) are possible relevant”.

Design a python program to find a training set $\underline{D}$ which includes both $\underline{D}^+$ (positive – likely relevant documents) and $\underline{D}^-$ (negative – likely irrelevant documents) in the given un-labelled document set *U*.

The input file is “BaselineModel_R102.dat” or BaselineModel_R102_v2.dat, and

The output file name is “PTraining_benchmark.txt”, which has the following sample output:

```
R102 73038 1
R102 26061 1
R102 65414 1
R102 57914 1
R102 58476 1          D+
R102 76635 1
R102 12769 1
R102 12767 1
R102 25096 1
R102 78836 1
…
R102 86912 0
R102 86929 0
R102 11922 0          D-
R102 14713 0
R102 11979 0
…
```

Task 3. Discuss the precision and recall of the pseudo-relevance assumption.

Compare the relevance judgements (the true benchmark – “Training_benchmark.txt”) and the pseudo one (“PTraining_benchmark.txt”) and display the number of relevant documents, the number of retrieved documents, the recall, precision and F1 measure by using pseudo-relevance hypothesis, where we view documents in $\underline{D}^+$ as retrieved documents). You may use the function you designed in week 7). The following are possible outputs:

```
the number of relevant docs: 106
the number of retrieved docs: 25
the number of retrieved docs that are relevant: 20
recall = 0.18867924528301888
```

```
precision = 0.8
F-Measure = 0.30534351145038174
```

Please analyse the above outputs to understand possible uncertainties (or errors) in the generated training set “PTraining_benchmark.txt”. You can also discuss any possible ways to generate a high-quality training set.

Solution: Task 3

It looks the precision is very high (80%), but the recall is very low (19%), which means there are many relevant documents labelled as “0” in “PTraining_benchmark.txt”.

Typically, there is a trade-off between precision and recall. For example, more documents can be selected, i.e., if you want to increase the recall value, you can change the condition "document's BM25 score is greater than 1.00" to "document's BM25 score is greater than 0.80"; however, precision may be reduced.

Also, you may ignore some indeterminate documents ranked in middle, and label top-ranked documents as “positive” and bottom-ranked documents as “negative”.

Please try it out and discuss with your tutor if you have any new ideas.