Homework 1

Data Mining Technology for Business and Society

Deadline: 19 April 2017

Exactly Two students for each group

Data and software are available at:

http://www.diag.uniroma1.it/~fazzone/Teaching/Data_Mining_Technology_for_Business_and_Society_2016_2017/DMT4BaS_2016_2017.html

Software Configuration:

To perform the homework you have to download and extract **Homework_1_software.zip** file on your hard drive. Moreover, you have to set another time the classpath in the same way (but not identically) you already did in the exercise. This step is mandatory because you will use additional software for this homework.

Description:

In this homework you will configure a search engine on a particular collection of documents: *'Cranfield'* collection.
This collection consist of:

1. a set of html documents.
2. a set of queries.
3. a set of relevant documents ids for each query in the query set: the Ground-Truth.

One of the objective of this homework is to find the best configuration (in terms of stemming method and scorer function) for the search engine, using the available Ground-Truth data.
To evaluate the search engine performance, you will use the two metrics: **Average R-Precision** and **nMDCG.**
You can find the definition of Average R-Precision on slides and on book.
The evaluation metric nMDCG (normalized Modified Discounted Cumulative Gain) is a modified version of the nDCG (normalized Discounted Cumulative Gain).
The nMDCG of a particular query and value of k is defined as follow:

$$nMDCG(query, k) = \frac{MDCG(query, k)}{MaximumMDCG(query, k)}$$

The MDCG of a particular query and value of k is defined as follow:

$$MDCG(query, k) = relevance(doc_1, query) + \sum_{rank=2}^{k} \frac{relevance(doc_{rank}, query)}{\log_2(rank)}$$

The function "relevance" is defined as follow:

$$relevance(doc, query) = \begin{cases} 1, & \text{if } doc \in GroundTruth(query) \\ 0, & \text{otherwise} \end{cases}$$

You have to find out by yourself how to compute the MaximumMDCG(query, k), for each query in the GroundTruth and for each k in {1, 3, 5, 10}.

Let’s clarify the calculation of nMDCG with an example.
Let’s assume that for the input query ‘q’ we have obtained the following rank (sorted list of doc_id):
[doc_3, doc_5, doc_4, doc_1, doc_2, doc_10, doc_7, doc_9, doc_8, doc_6].
Let’s assume also that the GroundTruth associated to the input query ‘q’ contains the following documents’ identifiers (set of doc_id):
GroundTruth(‘q’)= {doc_2, doc_9, doc_4, doc_8, doc_3}.

We have:
MDCG(q, 1) = relevance(doc_3, q) = **1**
MDCG(q, 3) = relevance(doc_3, q) + relevance(doc_5, q)/log$_2$(2) + relevance(doc_4, q)/log$_2$(3) =
= 1 + 0/log$_2$(2) + 1/log$_2$(3) = **1.63**
MDCG(q, 5) = relevance(doc_3, q) + relevance(doc_5, q)/log$_2$(2) + relevance(doc_4, q)/log$_2$(3) +
+ relevance(doc_1, q)/log$_2$(4) + relevance(doc_2, q)/log$_2$(5) =
= 1 + 0/log$_2$(2) + 1/log$_2$(3) + 0/log$_2$(4) + 1/log$_2$(5) = **2.06**
MDCG(q,10) = relevance(doc_3, q) + relevance(doc_5, q)/log$_2$(2) + relevance(doc_4, q)/log$_2$(3) +
+ relevance(doc_1, q)/log$_2$(4) + relevance(doc_2, q)/log$_2$(5) +
+ relevance(doc_10, q)/log$_2$(6) + relevance(doc_7, q)/log$_2$(7) +
+ relevance(doc_9, q)/log$_2$(8) + relevance(doc_8, q)/log$_2$(9) +
+ relevance(doc_6, q)/log$_2$(10) =
= 1 + 0/log$_2$(2) + 1/log$_2$(3) + 0/log$_2$(4) + 1/log$_2$(5) +
+ 0/log$_2$(6) + 0/log$_2$(7) + 1/log$_2$(8) + 1/log$_2$(9) + 0/log$_2$(10) = **2.71**

Let’s assume the following values for the MaximumMDCG:
MaximumMDCG(q, 1) = 1
MaximumMDCG(q, 3) = 2
MaximumMDCG(q, 5) = 2.5
MaximumMDCG(q, 10) = 3

Finally, we have the following values for the four nMDCG of interest:
nMDCG(q, 1) = MDCG(q, 1)/MaximumMDCG(q, 1) = 1/1 = 1
nMDCG(q, 3) = MDCG(q, 3)/MaximumMDCG(q, 3) = 1.63/2 = 0.815
nMDCG(q, 5) = MDCG(q, 5)/MaximumMDCG(q, 5) = 2.06/2.5 = 0.824
nMDCG(q, 10) = MDCG(q, 10)/MaximumMDCG(q, 10) = 2.71/3 = 0.903

To complete the homework, you have to complete these five steps:

1. Create a collection on the set of html documents with MG4J.
2. Create an inverted index (with MG4J) on the collection trying different stemming methods: *default stemmer*,
   English stemmer and
   English stemmer able to filter stopwords.
3. Obtain results for each query using the software "homework.RunAllQueries_HW" trying different scorer functions: *CountScorer*, *TfIdfScorer* and *BM25Scorer*.
4. Use the Ground-Truth to evaluate the performance of each stemmer-scorer configuration: 3X3=9 total configurations.
5. Implement “Fagin’s Algorithm”.
6. Create a final report.
7. Put report, sw and data inside a single zip file.
8. Send the .zip file to fazzone@diag.uniroma1.it.

Zoom on Stemmers:

Here there is a list of stemmers that you have to use:

1. Default stemmer.
2. Stemmer for the English language: it.unimi.di.big.mg4j.index.snowball.EnglishStemmer.
3. Stemmer for the English language able to filter stopwords: homework.EnglishStemmerStopwords.

You can create an index with the default stemmer using the command shown in the exercise. For creating an index with a particular stemmer, you have to specify the stemmer using the '-t' option:

```
 java it.unimi.di.big.mg4j.tool.IndexBuilder -t PARTICULAR_STEMMER_COMPLETE_NAME -S
COLLECTION_NAME.collection COLLECTION_NAME
```

Zoom on Scorer Functions:

Here there is a list of scorer functions that you have to use:

1) NumOccurrences scorer function: a trivial scorer function that computes the score by adding the number of occurrences within the current document of each query term.

2) TF/IDF: a well known scorer function.

3) BM25: beyond TF/IDF, more information at http://mg4j.di.unimi.it/docs/it/unimi/di/mg4j/search/score/BM25Scorer.html.

For obtaining results for each queries in the query set, you can call this software from command line in the following way:

```
 java homework.RunAllQueries_HW "COLLECTION_NAME" ./COLLECTION_NAME_all_queries.tsv
"SCORER_FUNCTION" "FIELD" OUTPUT_FILE_NAME.tsv
```

Typing java homework.RunAllQueries_HW you will obtain a guide on how to use this software.
Important note: the software homework.RunAllQueries_HW requires that both Inverted-Index and Collection have the same name.

Zoom on Aggregation:

For this step you have to implement, in Python or Java, the “Fagin’s Algorithm”. This software have to aggregate the scores obtained on field “Title” and on field “Text” with a simple “sum” aggregation function, but considering the score on “Title” two times more important than the score on “Text”.

For this step you have to consider **only** the results from the [*English_stemmer_able_to_filter_stopwords*, *BM25Scorer*] configuration.

This software must take in input two files that are the output files of homework.RunAllQueries_HW software (one for “Title” and one for “Text”) and must give in output a file of the same format: [Query_ID \t Doc_ID \t Rank \t Score].

You must use this output file to evaluate the performance of the aggregation strategy.

Zoom on Performance Evaluation:

To evaluate the performance of a particular stemmer-scorer combination, you have to write an ad-hoc Python or Java software.
More precisely, you must to write a software for each requested evaluation metric: a software for the evaluation of the **Average R-Precision** and another software for evaluating **nMDCG.**
These softwares must be able to evaluate the **average nMDCG** and the **Average R-Precision** over all queries in the query set, by taking in input the search engine results for each query in the query set (the output file of `homework.RunAllQueries_HW` software) and the Ground-Truth file (`COLLECTION_NAME_Ground-Truth.tsv`).
For what concerns **average nMDCG**, we have interest only for k \in {1, 3, 5, 10}.
For evaluating the output of the two aggregation algorithms, only the computation of the **Average R-Precision** is requested.

What To Put in the Report:

The final report must contain:

1. A simple statistic on the used dataset: #documents and #queries.
2. A list of used stemmers.
3. A list of used scorer functions.
4. The script to create the collection.
5. The scripts to create the inverted indexes (for all stemmers).
6. The scripts to obtain the results from the search engine (for all scorer function).
7. The Average R-Precision for each stemmer-scorer_function configuration end for the aggregation algorithm: (3X3 + 1)= 10 Average R-Precision values.
8. The plot of the average nMDCG that have:
   1) on the 'x' axis the value of k. Consider only the following values: 1, 3, 5, 10.
   2) on the 'y' axis the average value of nMDCG: average value over all queries.
   3) one curve for each stemmer-scorer_function configuration: 3X3=9 curves in total.
9. An answer to each of the following questions:
   Which is the best combination stemmer-scorer_function?
   Which is the best stemmer?
   Which is the best scorer function?

Where To Send What:

At the end of the process, you have to create a zip file with **ONLY** the following data:

1. The software to evaluate the Average R-Precision.
2. The software to evaluate the average nMDCG.
3. The software implementing “Fagin’s Algorithm”.
4. All tsv files containing the results for each query in the query set.
   You will have 3X3+1=10 tsv files of this type: three different stemmers, three different scorer functions and two different aggregation algorithms.
5. The final report in **PDF**.

The file name must have this format:
WEBIR__STUDENTID_NAME_SURNAME__STUDENTID_NAME_SURNAME.zip
Finally you must to send the .zip file to fazzone@diag.uniroma1.it with this email subject:
WebIR_StudentID_StudentName_StudentSurname_StudentID_StudentName_StudentSurname.

For any problem/doubt, please use Piazza. Don't be shy :)