INF3800/INF4800

Assignment D

WHAT TO DELIVER:
Please submit a zip containing both the implementation task (the entire project) and the pencil-and-paper tasks. Exercise 2 is optional for INF3800 students.

DEADLINE:
The assignment must be submitted, using Devilry, by 27.03.2017 at the latest.

On paper

Exercise 1:

Query term proximity

The scoring strategies we have considered so far do not take into account the distance between matches of query terms in the documents. For instance, a search for *rachmaninov 3* will yield the same score for a document where *rachmaninov* and *3* directly follow each other, and one where these two terms appear at opposite ends of the document.

A scoring method that considers the relative closeness of the query terms is generally desirable. How can it be integrated in a search engine? Provide a few alternative options for such implementation, and describe the advantages and disadvantages of each.

Exercise 2: (Optional for INF3800)

Relevance evaluation

Task 1
You want to quantify how relevant the results returned by a given search engine are, given a set of benchmark queries with relevance judgments. For the three benchmark queries below the search engine's top 10 results are:

```
ariana grande
RNRNNRNNRR

trine skei grande
NRNNRNRNNN

jono el grande
RNNRNNNNNR
```

Here R indicates a document that is relevant to the query and N indicates a nonrelevant document. You can assume that all relevant documents are among the top 10 results. What is the search engine's MAP (mean average precision) on this set of benchmark queries?

Taks 2

Wanting to go beyond binary relevancy judgments, you now produce a benchmark query "grande latte" with relevance judgments on a scale of [0, 16] as follows:

```
Document A = 4
Document B = 2
Document C = 8
Document D = 16
```

For all other documents the relevancy judgment is assumed 0 for this query. For this query the search engine's top 10 results are:

```
grande latte
ADXCXXXBXX
```

Here X indicates a document other than the ones listed above. What is the search engine's DCG (discounted cumulative gain) score at rank 10 for this query? Use a simple logarithmic discounting.

Task 3

Explain how you would normalize the DCG score you computed above to arrive at the search engine's NDCG (normalized DCG) score at rank 10.

(This exercise is taken from the V15 final exam)

In code

TF-IDF weighting

In the previous assignment, we implemented a query evaluator, and relied on a simplistic BrainDeadRanker to return the relevance score. The only operation performed by BrainDeadRanker was to calculate the overlap between the words in the query and the ones in the document.

A more realistic search engine should use a weighting scheme for the words in the document, given some words are more relevant than others. For instance, if you search for the query *rachmaninov 3*, you will probably be more interested in the documents which include *rachmaninov* than the ones which include *3*.

TF-IDF is such a weighting scheme. In this assignment, we will therefore replace the BrainDeadRanker by a new TfIdfRanker. Your task is to implement the update(…) method in the TfIdfRanker, and ensure that the result is correct using the provided JUnit test.

Document similarity

Many search engines (such as Google) offer the possibility to find similar pages after a search. This can be done by computing the cosine similarity between documents, and returning the documents with the highest score. To do this, you have to extend the system with a document vector representation. Each document gets a distinct document vector, which can then be compared with other document vectors using the cosine similarity measure.

DocumentEvaluator evaluates an input document against all other documents in the document store, and returns the documents with highest cosine score. You can keep this class as it is, and complete the implementation of DocumentVector. Again, use the JUnit test to determine whether your code is correct.