RAWDATA Portfolio-project 3 requirements

A major part of the data we are dealing with in the SOVA application under development is textual and the key function of the application is retrieval of documents, that is, posts. Thus, an obvious extension is to introduce Information Retrieval (IR) functionality and to prepare the data in order to improve this functionality. The main goal of this portfolio subproject 3 is exactly that.

This subproject 3 does not bring any changes to the overall architecture of the application to be developed. Thus, the sketch in the general *RAWDATA Portfolio project* description and the details in *Portfolio-subproject 2 requirements* should still hold. However, some of the additions of Portfolio 3 may call for changes in what was developed in Portfolio 1 and 2.

A. Application design

Describe the changes you have done to the design of your SOVA application. You may refer to the A-sections of portfolio 1 and 2 or repeat fractions from these as you prefer. The important thing here is that it is possible for the reader to get an overview of your design and to understand your motivation and arguments for your recent decisions. The level of detail in the design description in portfolio 2 is not required for portfolio 3.

B. Extended IR functionality

To simplify the indexing for Information Retrieval on the textual columns in the stackoverflow data, some data preparation is already done and provided by means of the table **words**. It can be used as source to build various kinds of inverted indexes and, as it is, it also comprises a combined inverted index for **posts** and **comments**. The column **id** is the id of the **comment** or **post** where the **word** appears. The columns **tablename** and **what** are used to distinguish what an entry refers to (either the **title** or **body** in **posts** or the **text** column in **comments**). The columns **sen** and **idx** together specifies the position of the word, by sentence and word number. In addition, the table provides a lemmatization[1] of words with the column **lemma** and a specification of the word category (part-of-speech) with the **pos** column. The specifications in the latter are given using the Penn Tree Bank tags[2].

id	tablename	what	sen	idx	word	pos	lemma
60496	posts	body	8	12	when	WRB	when
60496	posts	body	8	13	something	NN	something
60496	posts	body	8	14	goes	VBZ	go
60496	posts	body	8	15	wrong	JJ	wrong
60496	posts	body	8	16	.	.	.

Figure 1: Small excerpt from the table **words**

From figure 1 we see that sentence 8, in the **body** of the **post** with **id** 60496, includes the phrase "when something goes wrong" as the last 4 of 15 words in that sentence (must be the last 4 words due to the punctuation mark as word 16).

The requirements to extended IR functionality are the following. Observe that these are extensions and earlier requirements still hold – especially you are supposed to deal with history as in Portfolio 1 (relevant for B.1, B.2, B.4, B.5) and uniform interface and paging as in Portfolio 2.

B.1. **Exact-match querying and Boolean indexing:** Introduce an exact-match querying function that takes one or more keywords as arguments and returns posts that match all of these. Build and use an inverted index (rather than simply using the like-operator). Thus, your querying should be processed (conceptually) by merging postings lists. You are supposed to build your own inverted index but you can take advantage of the data preparation provided with table **words**.

[1] https://en.wikipedia.org/wiki/Lemmatisation

[2] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

B.2. **Best-match querying:** Develop a refined function similar to B.1, but now with a "best-match" ranking and ordering of objects in the answer. A best-match ranking simply means: the more keywords that match, the higher rank, and the objects in the answer should be ordered by decreasing rank.

B.3. **Weighted indexing:** Build a new inverted index for weighted indexing similar to your index from B.1, but now with added weights to the postings. Discuss and decide on a weighting strategy. A good choice would probably be a variant of TFIDF, but apart from the details of this you should also consider other related issues for instance what exactly to index, how to normalize terms and whether or not to take stopwords into consideration. Be aware that some of the entries in **words** stem from an imperfect tokenization, so filtering to omit some of these could also be considered.

B.4. **Ranked weighted querying:** Develop a refined function similar to B.2, but now with a ranking based on a TFIDF weighting provided by the indexing derived in B.3.

B.5. **Word-to-words querying:** An alternative, to providing search results as ranked lists of posts, is to provide answers in the form of ranked lists of words. These would then be weighted keyword lists with weights indicating relevance to the query. Develop functionality to provide such lists as answer. One option to do this is the following: 1) Evaluate the keyword query and derive the set of all matching posts, 2) count word frequencies over all matching posts, 3) provide the most frequent words (in decreasing order) as an answer (the frequency is thus the weight here). Notice that this approach ignores weights in postings. Consider an alternative that take TFIDF weightings into account. Discuss and compare.

B.6. **Word cloud visualization:** Provide one or a few examples of visualizations of weighted keyword list answers (results of B.5 and/or B.6), by copying the answer(s) and pasting into a word cloud visualization tool[3]. Notice that this is intended to be a subject for further elaboration in Portfolio 4.

B.7. **Co-occurrence term network:** Extend your data model and database by introducing a co-occurrence term network, that is, a table of two-word combinations that includes weights corresponding to how often the two words co-occur. The weight can be based on the count of co-occurrences in all the posts in the database. To allow investigation of the database as a whole (from this perspective), develop a function that can take a word as input and return the closest terms as "context".

B.8. **Force network visualization:** Provide one or a few examples of visualizations of force networks based on the result of a word query as in B.7. Download **generate_force_graph_input.sql** and follow the instructions in the note on Moodle: *Visualization of a term network (Force Graph)*. Use the retrieved words as nodes and include all edges connecting nodes in this set. Edges are the weighted co-occurrence in the term network. Notice that this is intended to be a subject for automation in Portfolio 4.

Consider, *if time allows*, one or two of the following issues.

B.9. [OPTIONAL] Introduce a similarity search, such that evaluation of keyword-match is independent of the word form (ending) used in the posts.

B.10. [OPTIONAL] Introduce a word proximity based search, such that a query for instance may be in the form "A B; C; D E" when looking for: A followed by B, C, and D followed by E, where the first and third criteria are better satisfied, the closer B (respectively E) comes after A (respectively D). You can alternatively introduce your own syntax for proximity specification and/or get inspiration considering the proximity connectors defined for the WestLaw system[4].

---

[3] One option is to use the word cloud visualization tool available on Moodle.

[4] https://www.cbs.dk/files/cbs.dk/wln_getting_started_ug1_0.pdf

B.11. [OPTIONAL] Consider and suggest an approach to relevance feedback that allows the user to select most interesting documents in the first result set and the "system" to generate a second result set taking the users preference into consideration.

B.12. [OPTIONAL] Consider and suggest an approach to query expansion, where the query is modified before the answer is derived. The general principle is to add words to the query that are similar to words already there – e.g. by being synonyms, by being frequently co-occurring or by being inflections of the same word.

B.13. [OPTIONAL] If you have an idea of your own, you can plug it in here …

C. Improving performance by indexing

One of the advantages of using a relational database in the data layer is that query performance can be improved and tailored to the actual needs (most frequent and/or most important queries/query types) at any state of the development process (even after the development has finished). Without need to reconsider other parts of the system, performance can be improved by adding or modifying the **indexing** of tables in the database.

C.1. Review the indexing of your database as modeled in portfolio subproject 1. Consider improvements.

C.2. Consider the extension developed under **B** and discuss/explain what is needed to obtain an acceptable performance when dealing with inverted index, weighting and term networks.

D. Data Access Layer and Web Service Layer

As in the A section it is satisfactory just to focus on the changes compared to Portfolio 2. Describe, document and implement the changes that your additions in **B** calls for.
Follow the structure from Portfolio 2 and cover your model, access layer and interfaces. Be brief and assume that the reader has read your previous reports.

E. Testing

The changes you make to your implementation in this subproject should of course be tested. However, the detailed unit level and integration tests you did in Portfolio 2 are not required here. Rather you should perform a black box testing of your system in line with what you did in assignment 3. Write a test suite and in that connection, discuss and decide on a reasonable level of detail.

The project report

You are supposed to continue in the groups from portfolio 1 and 2 and submit your portfolio 3 result by implementing the revised database (including new functionality) on the course server rawdata.ruc.dk and committing the Visual Studio solution, including projects for each part(layer) to GitHub (or similar resource) with a Section3 tag. The URL of this resource together with the Portfolio project 3 report (in pdf format) must be uploaded to the course website on Moodle.

The report should in size be around 6-12 normal-pages[5] excluding appendices. The submission deadline for the report as well as the product is 26/11-2018.

Notice that the report you hand in for Portfolio project 3 is not supposed to be revised later. However, if you find good reasons for this, your design and implementation can be subject to revision later. Documentation for such changes can be included in the final combined Portfolio project 4 / Reflexive synopsis report.

[5] A normal-page corresponds to 2400 characters (including spaces). Images and figures are not counted.