Information Retrieval and Data Mining SS 2017
Prof. Dr.-Ing. Sebastian Michel
MSc. Koninika Pal
TU Kaiserslautern, FB Informatik – Lehrgebiet Informationssysteme
Exercise Sheet 3: Hand out 15.05.2017, Due 30.05.2017 - 16:00

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN
http://dbis.informatik.uni-kl.de

Note: You need to upload a single zip file to OLAT, containing all files— the code of your solution to assignments that involve coding and a pdf document with the solutions of the non-coding parts.

Assignment 1: Language Model with different Smoothings (1 P.)

Suppose we want to search in the following collection of Christmas cookie recipes. The numbers in the table below indicate raw term frequencies.

	milk	pepper	raisins	sugar	cinnamon	apples	flour	eggs	clove	jelly
d_1	4	0	0	4	0	1	1	0	0	0
d_2	1	1	0	2	0	0	0	0	1	0
d_3	3	1	0	2	0	0	0	2	0	0
d_4	1	2	1	1	2	0	2	1	0	0
d_5	2	0	2	0	1	0	5	2	1	2
d_6	1	0	0	0	0	0	1	1	0	2
d_7	2	1	0	0	1	0	0	0	0	1
d_8	0	0	3	2	0	1	0	4	0	0

(a) Determine the top-3 documents including their query likelihoods for the query

$$q_1 = \langle \text{sugar}, \text{raisins}, \text{cinnamon} \rangle$$

using the model $P(q|d) = \prod_{t \in q} P(t|d)$, with MLE probabilities for $P(t|d)$.

(b) Determine the top-3 documents when using Jelinek-Mercer smoothing ($\lambda = 0.5$).

(c) Determine the top-3 documents when using Dirichlet smoothing (for a suitable α).

Assignment 2: Latent Semantic Indexing (1 P.)

We suggest to use R (as briefly mentioned in the lecture) to solve this assignment. Alternatively, you can use Python or your favorite language/tool, but the submitted solution should show not only the outcome but the way it was achieved.

Consider the following term-document matrix.

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12}
human	2	1	1	0	0	0	0	0	0	1	0	0
genome	1	2	0	1	0	0	0	0	0	0	0	0
genetic	1	2	1	2	1	0	0	0	0	0	1	0
molecular	0	1	2	1	0	0	0	1	0	0	0	1
host	0	0	0	0	1	1	2	0	0	0	0	0
bacteria	0	0	0	0	1	2	1	1	0	0	0	0
resistance	0	1	0	1	0	1	3	2	0	0	0	0
disease	0	0	1	1	1	2	2	3	0	0	0	0
computer	1	0	0	0	0	0	0	0	2	2	1	0
information	0	0	1	0	0	0	2	2	3	0	1	0
data	1	0	0	0	0	0	1	0	1	1	1	2

(a) Determine how many dimension of the topic space you want to reduce to remove noise without loosing valuable information, justify your choice. Then, compute the top-3 similar documents for the following query using LSI on the reduced topic space:

$$q = \langle 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0 \rangle$$

(b) Determine the most related word to *gene* which appears in document d_1, d_2, d_4, d_5, and d_{11}, i.e.,

$$gene = \langle 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0 \rangle$$

Assignment 3: Benchmarking Scoring Functions (1 P.)

Download the `benchmark.tar.gz` file from the course website and extract its contents. You will find the following three files[1]:

- `documents.csv` contains, separated by semicolon and one line per document, the document id, the stemmed document content, and the non-stemmed document content.
- `queries.csv` contains one query per line, separated by semicolon, in the form, query id, stemmed full query description, non-stemmed full query description, stemmed keywords, non-stemmed keywords.
- `assessments.csv` contains pairs of query id; document id where the document has been found relevant to the query by human assessors.

Your task is the following:

(a) Start with a simple tf*idf scoring model and implement in Java (alternatively C++, Scala, Python, Ruby, or R) an evaluation framework that executes the benchmark queries over the document corpus and computes average precision and recall for the scoring function using the relevance assessments. The use of any library that assists with scoring or indexing is prohibited, also for part (b).

(b) Implement two more scoring models besides the plain tf*idf model and compare the retrieval performance of all three. For instance Okapi BM25 or the mentioned (simple) alternate forms of tf and idf (no need to compute LMs in this task). Try to play a bit around with these scoring models, e.g., regarding stemming or maxtf vs. sumtf normalization, in order to determine the best performing model you can find. We are already curious to see which group(s) will submit the highest recall / precision values and the underlying scoring model.

Next to the code, include in your .pdf file the equations underlying your evaluated scoring models. For the code, make sure it can be executed directly in the folder where the three files are put. Include a small readme file with the commands needed to execute the code to give the results of (a) and (b).

[1] This benchmark data is originally from `http://ir.dcs.gla.ac.uk/resources/test_collections/cran/`. A big thank you goes to the authors for making this benchmark data available. We have just parsed the, originally a bit complicated, syntax and kept only the document content in form of terms (originally, also document author, title, etc. were given). We added also the stemming and the short(er) keyword queries. We have picked the first 6 queries of the benchmark (and renumbered them for convenience, also of course adapted the references in the relevance assessment file for which we treat all graded documents as relevant, ignoring the level of grading that was also originally given).