# Web Information Retrieval

**Assignment 6**

Lukas Schmelzeisen lukas@uni-koblenz.de
Dr. Chandan Kumar kumar@uni-koblenz.de

Institute of Web Science and Technologies
Department of Computer Science
University of Koblenz-Landau

Submission until: June 20, 2017, 10:00 a.m.
Tutorial on: June 20 and June 23, 2017

This assignment covers *Probabilistic Information Retrieval* and *Web Search Basics* in Information Retrieval. As supplementary material to the lecture, I can recommend reading *Chapters 11* and *19* of the book "Introduction to Information Retrieval" by Manning, C. D., Raghavan, P., and Schütze, H. from 2008. The book is available for free under http://informationretrieval.org

# 1 Binary Independence Model (10 Points)

## 1.1 Comparision to TF-IDF (6 Points)

Describe the differences in how the following two retrieval models rank documents and what practical consequences this has:

- The *Binary Independence Model*, as thought in the lecture, together with how term probabilities could be estimated in practice.
- The *TF-IDF vector space model*, as though in the lecture

## 1.2 Document Relevance (4 Points)

The binary independence model $P(R \mid d, q)$ (and also the IR model based on language models $P(d \mid q)$) ranks documents after a probability of relevance of each document computed independently of other documents (i.e., documents are scored against the query and not against each other).

Describe why this assumption could be problematic in practice.

# 2 Web Search Characteristics (30 Points)

## 2.1 Shingling (12 Points)

Given the following document corpus:

**Doc A.1**: `"a bump on the log in the hole in the bottom of the sea"`
**Doc A.2**: `"a frog on the bump on the log in the hole in the bottom of the sea"`

**Doc B.1**: `"your mother drives you in the car"`
**Doc B.2**: `"in mother russia car drives you"`

Calculate the similarity of both pairs of documents (i.e., similarity of A.1 and A.2 and similarity of B.1 and B.2) via the jaccard-coefficient of their shingle sets with $n$-shingles for $n = 1, 2, 3$.

Specify both the shingle sets aswell as the final document similarity scores.

## 2.2 Implementation (8 Points)

Implement the above task in Python. Create a new file `shingles.py` that—only given the input documents—computes and outputs shingle sets and similarity scores with $n$-shingles for $n = 1, 2, 3$.

You may use any functions in the Python standard library but no other libraries.

## 2.3 Hashing for Duplicate Detection (6 Points)

Explain why comparing the hash values of documents (i.e., computing fixed-size numbers from their contents) is not sufficient for duplicate detection on the web.

Find an example of duplicate documents on the web which hashing could probably not detect.

## 2.4 Choice of Prior (4 Points)

In the formulation of the language model-based retrieval model Bayes' theorem was used as

$$P(d \mid q) = \frac{P(q \mid d) \cdot P(d)}{P(q)}$$

and the document prior (the probability of a document $P(d)$) was treated as equal for all documents (i.e., as a uniform distribution) meaning that it could be ignored in ranking.

However, in web search the document prior could be a significant factor to model the scenario that independent of the query some documents could be more relevant than others. Give four different examples of factors that could be considered for document prior relevance.

# Important Notes

Ask questions and discussion with regard to the lecture, tutorial, or assignments in

- The WebScience newsgroup:
  https://webnews.uni-koblenz.de/newsgroups.php?group=infko.webscience
- Our Facebook group:
  https://facebook.com/groups/InformationRetrievalUniKoblenz

## Submission

- Solutions have to be checked into the SVN repository. Use the directory name `solutions/assignment6/` in your group's repository *mandatory*.
- The SVN repository is available via
  https://svn.uni-koblenz.de/westteaching/ir-ss17/<groupname>
- Solution format: all solutions for theoretical taks as *one* PDF document. Programming code has to be submitted as Python code.
- The name of the group and the names of all participating students must be listed on each submission:
  - at the top of each PDF file
  - at the top of each Python file (in comments)

  Only named students will receive credit.
- With the submission of your solution you confirm that you created the solution independently as a group, especially without using other intellectual contributions. That is, you submission should not be plagiarism!

## Programming Assignments

The programming assignments require you to have a Python 3.6+ interpreter (older versions may still work coincidentally, but are not officially supported) and the SciPy stack installed. For an installation guide, see http://west.uni-koblenz.de/en/studying/installing-python

The following rules apply for the *implementation* of your solution:

- You can create as many additional code files as you need. You can create as many additional classes or methods as you need (even in the provided code files). However, do not forget to submit *all* `.py` code files to the SVN.
- Check that your code compiles and runs without errors.

- Use `UTF-8` as the file encoding. *Other encodings will not be taken into account!*
- Test your implementation with the provided test cases.

  Passing of all tests is a *necessity* to receive full score, no *sufficiency*. In general, programming against the provided test cases should assist you in finding correct solutions.
- Make sure your code is formatted to be easy to read.
  - Make sure you code has consistent indentation.
  - Make sure you comment and document your code adequately in English.
  - Choose consistent and intuitive names for your identifiers.
- Do *not* use any accents, spaces or special characters in your filenames.