Web Information Retrieval

Assignment 3

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: May 23, 2017, 10:00 a.m.
Tutorial on: May 23 and May 26, 2017

This assignment covers *Boolean Retrieval.* As supplementary material to the lecture, I can recommend reading *Chapter 1* and *Sections 2.4* and *3.1* 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 Boolean Retrieval (18 Points)

Consider the following collection of 4 documents:

Doc 1: `"preliminary findings in cancer research"`
Doc 2: `"novel cancer research findings"`
Doc 3: `"new research in cancer healing"`
Doc 4: `"new optimism in cancer patients"`

1.1 Index Construction (10 Points)

1. List all (term, document-id) pairs for this document collection.
2. Specify the term-document matrix for this document collection in a similar style as in the lecture (sort terms alphabetically).
3. Draw the inverted index for this document collection in a similar style as in the lecture, namely containing a posting list for each vocabulary word.

1.2 Binary Search Tree (4 Points)

One way of efficiently looking up the posting list for a word is using a *binary search tree* (this was also depicted in the lecture slides on inverted indices). Research how binary search trees work and draw the tree for the vocabulary of the above corpus.

1.3 Retrieval (4 Points)

For the document collection above, what are the returned documents that match the following queries? Incude your derivation of the solution.

1. `"cancer" AND "research"`
2. `"in" AND NOT ("research" OR "healing")`

2 Preprocessing (8 Points)

Consider the following documents:

Doc 1: `"My name is Bond, James Bond."`
Doc 2: `"That was probably one of the most well known 'James Bond' quotes, right?"`

Research what the following techniques do and apply them in order to each of the given documents. Show results for the intermediary steps.

1. Tokenization
2. Lemmatization
3. Case Folding
4. Stop Word Removal

Showcase the data type using the common python notation, i.e. `"this is a string"` and `["this", "is", "a", "list"]`.

3 Positional Inverted Index (14 Points)

Your task is to extend the implementation of an simple inverted index to a *positional* inverted index (as discussed in the lecture).

A positional inverted index can not only efficiently find all documents that contain a certain keyword (or boolean operations on top of it), but it can also find all documents that contain a certain phrase (a sequence of words). In order to do this, it also has to store word positions for each document.

The implementation of an inverted index is given in file `inverted_index.py`. For your solution, you will mainly have to implement the method `find_documents_with_phrase()`. But you will probably also have to modify the existing methods, since they do not store/handle positional information.

You can test your solution using file `test_inverted_index.py`.

For this task you may not `import` any non self-written modules.

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/assignment3/` 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.