Week 4:
Retrieval Models I

- This assignment is due on **11th November, 2015 (13:30)**
- You can discuss the problems with other groups of this course or browse the Internet to get help. However, copy and paste is cheating.
- There are 13 weekly exercises in total. In each one of them, all assignments sum up to 20 points. You need to achieve at least 80% of all assignments during the course in order to participate in the final exam. Hence, you need to achieve at least 208 points in total (*13*20*0.8=208*).
- Submission at
  `https://www.dcl.hpi.uni-potsdam.de/submit`
  - only pdf files
  - one file per group per week (week4.pdf)
  - put your names on *each* sheet in the pdf file

Assignment 1: Boolean Retrieval

a) One of the drawbacks of the Boolean retrieval model lies in the size of the returned result set. Why is the size typically difficult to control? **3 P**

b) Does Google support Boolean search? Which operators? **2 P**

c) In general, patent professionals agree that Boolean queries are a powerful retrieval tool for patent data. Why do you think that Boolean operators are necessary in professional search? Are consumer and professional search precision or recall oriented? **4 P**

Assignment 2: Boolean Retrieval in Practice

Given the following documents

$$\begin{array}{ll} D_1 = t_1, t_5, t_9 & D_4 = t_4, t_5, t_{10} \\ D_2 = t_1, t_2, t_4, t_5, t_9 & D_5 = t_3, t_5, t_6, t_7 \\ D_3 = t_3, t_6, t_7, t_8 & D_6 = t_1, t_2, t_{10} \end{array}$$

a) Evaluate the query: $q_1 = (t_1 \text{ OR } t_5) \text{ NOT } t_2$. **2 P**

b) Evaluate the query: $q_2 = (t_1 \text{ AND } t_5) \text{ OR } (t_3 \text{ AND } t_2)$. **2 P**

Assignment 3: (Programming) Boolean Queries

This week we will implement a simple Boolean retrieval system based on the index we built in the previous assignment.

- We want to implement Boolean search this week. Therefore you need to change the implementation of your `ArrayList<String> search(String query, int topK, int prf)` to support boolean queries. As in the previous assignments, you should ignore the last two parameters of the method. The return value should be a list of patent invention titles of the documents relevant to the query.
  - Load the seek list of your compressed index.
  - Pre-process the query (stemming, stopword removal).

- Identify Boolean keywords (″AND″, ″NOT″, and ″OR″) in the query and also the prefix (*) operator for enabling prefix search.
- As far as the boolean operators are concerned, you will only allow these query types: ″a AND b″, ″a NOT b″ and ″a OR b″. No complex, nested Boolean queries.
- Implement prefix search to answer queries such as ″per*″, i.e. find the patents that contain words starting with ″per″ (period, permission, etc.).
- Allow combinations of prefix and Boolean queries such as ″pro* NOT protection″. That is retrieve all the documents that contain words such as process, program etc. but not the term protection.
- Implement phrase queries to find all patents containing an exact phrase, such as ″The application is installed using″.

a) Print the invention titles of the patents that match the following queries:

- ″comprises AND consists″ **1 P**
- ″methods NOT inventions″ **1 P**
- ″data OR method″ **1 P**
- ″prov* NOT free″ **1 P**
- ″inc* OR memory″ **1 P**
- ″the presented invention″ **1 P**
- ″mobile devices″ **1 P**