Recuperação de Informação / Information Retrieval

2018/2019 MIECT/MEI, DETI, UA

Assignment 1

For this assignment you will create a simple document indexer. This should consist of a corpus reader, document processor, tokenizer, and indexer.

The corpus for this assignment is the Amazon Customer Reviews Dataset, available here: https://s3.amazonaws.com/amazon-reviews-pds/readme.html

1. Create a corpus reader class that returns, in turn, the contents of each document in a collection (corpus). The document should be processed to ignore any irrelevant sections, and a document identifier should also be returned.

2. Create different tokenizer implementations.

   2.1. A simple tokenizer that splits on whitespace, lowercases tokens, removes all non-alphabetic characters, and keeps only terms with 3 or more characters.

   2.2. An improved tokenizer that incorporates your own tokenization decisions (e.g. how to deal with characters such as ‘, -, @, etc).
   Integrate the Porter stemmer (http://snowball.tartarus.org/download.html) and a stopword filter. Use this list as default: http://snowball.tartarus.org/algorithms/english/stop.txt

   2.3. Answer the following questions:
   a) What is your vocabulary size using each tokenizer?
   b) For each tokenizer, list the ten first terms (in alphabetic order) that appear in only one document (document frequency = 1).
   c) For each tokenizer, list the ten terms with higher document frequency.

3. Create an indexer and index the corpus using a suitable data structure defined by you. Apply stemming and stopword filtering and save the resulting index to a file using the following format (one term per line):

   aaaaa,doc id:term freq,doc id:term freq …
   aaaab,doc id:term freq,doc id:term freq …
   aaaac,doc id:term freq,doc id:term freq …
   (…)

Note:

Your implementation should be modular and easily extended/adapted to other corpora structures.

Your assignment will be evaluated in terms of: modelling, class diagram, code structure, organization and readability, correct use of data structures, submitted results, and report. See suggestions and submission instructions below.

Suggestions:

- Write **modular** code
- Favour **efficient** data structures
- Add **comments** to your code
- Follow the **submission instructions**

Submission instructions:

- To manage your project please use **Maven**
- At each submission, include a small **Report** including:
  - Your project's **class diagram**
  - A description of each class and main methods, identifying where these are called
  - A data flow diagram and description of the overall processing steps
  - Instructions on how to run your code, including any parameters that need to be changed
  - A list of any external libraries that are needed to run the code
  - Efficiency measures: total indexing time; maximum amount of memory used during indexing; total index size on disk
  - A short commentary/assessment of your own work, describing features or implementation decisions that you consider the most relevant/positive (or otherwise)
- Make sure you **include your name and student number** in the code and in the report.
- Make sure all your programs compile and run correctly.
- Submit your assignment by the due date using Moodle.