CW1: Document indexing

The coursework aims to help you develop skills in document indexing. You are expected to submit a jupyter notebook to address the two tasks as specified below, along with a report that describes what you have done and to critically evaluate the resulting implementation and the outcome.

We suggest using nltk to support your development. Note that this coursework is not intended to have a single correct answer in terms of what is developed or how. Thus, you can choose (and then motivate and explain in your report) different functionalities. You could even support the same functionalities in different ways and then compare them.

Context of the task:
Your boss is keen on The Simpsons series and is asking your team to develop a specialised search engine. Someone has already downloaded a corpus with a number of episodes from Wikipedia (in a text format), where each document corresponds to a single episode (folder *Simpsons*). You are specifically asked to build *an inverted index* for that collection, which will support retrieving episodes that match the user's query. In addition to a random keyword query, your boss would specifically like to be able to find episodes in which specific characters (e.g. *Bart Simpson*, *Nice Family*) or locations (e.g. *Bart's Treehouse*) are featured. Luckily, your colleagues have already collected lists of these entities.

Task 1: Corpus analysis [5 marks]
To warm up for the main task, you will first perform a simple corpus analysis of the dataset. Specifically, you are asked to provide a list of top 100 most frequent tokens in the corpus. You may want to consider if you need to do any document cleaning, what useful stop words for the task might be and whether you would like to do any case folding. You should report and justify your decisions in the report.

Task 2: Inverted index
Your main task is to construct an inverted index from a corpus that can support efficient and effective search. The functionalities to be supported are divided into two parts.

Part 2.1 [15 marks]
You will implement an inverted index with term positional information. You should consider carefully and justify what types of index terms are appropriate for the task, e.g., single words, multi-word terms, etc. You should also consider what kind of pre-processing you would do – e.g. whether you should apply stemming, stop-word removal and case folding. You will need to justify the decisions in the report (you may refer to Task 1 above), including how they affected the index both in quality and size. You can refer to specific examples that you can include in your report (see below).

Part 2.2 [5 marks]
Write a demo function that will retrieve documents that contain either a given term, or two given terms that are within a predefined window size, and discuss whether this could be used to support the specification of the task as above.

Deliverables

You should submit the following in an archive (i.e., a zip file) via Blackboard containing:

(1) a jupyter notebook for the tasks, which will implement at least the functions that are specified in the template available at the end of this document (and in BB).

-

(2) a single PDF file that includes:

1. A report of not more than 4 pages that explains what you have done and why in each of the tasks. Specifically, for Task 2, you should describe
   i. What features were implemented, what they did to improve the index, and what problems may they create?
   ii. The limitations of the design and its implementation.

   Note that quite a few marks require you to go beyond bookwork. Your report should not be generic (e.g. "stop-words were removed"), but rather focused on what the specific features of *your* implementation are, why you decided to do that, how does it link to The Simpsons data set, what problems might have been created (if any), etc.

2. Up to 2 pages with fragments of the output of your program: the top tokens in Task 1 **and** an excerpt of the index for Task 2. The excerpt should include complete index entries for terms that illustrate well the properties of your index. You can refer to this sample from your report to provide specific examples or points.

Submission Deadline: Friday October 29th 6pm (end of Week 5)

This is a hard deadline; extensions will only be granted as a result of formally processed Mitigating Circumstances (http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=4271). Marks for late submissions will be reduced in line with the university's rather punitive policy (http://documents.manchester.ac.uk/display.aspx?DocID=24561).

Support

This is an individual piece of coursework. The coursework will be supported by a lab session in week 4 (Friday October 22nd 2-4pm) for any questions you may have. Make the most of this session as it might be difficult or indeed late to make arrangements to discuss your coursework-related questions at other times.

Assessment

This exercise is worth 25% of a student's overall mark for this course unit. Credit will be given taking into account: (i) the amount of functionality supported; (ii) the elegance and effectiveness of the design and implementation; and (iii) the clarity and depth of insight in the report.

A first-class level submission will provide comprehensive functionality, with few mistakes, implemented in an elegant and efficient way, with an insightful discussion of the strengths and weaknesses of decisions made.

An upper-second level submission will provide significant functionality, with few mistakes, implemented in a generally appropriate way, with a reasonable discussion of the strengths and weaknesses of the features supported.

A lower-second level submission will provide reasonable functionality, but may well include some mistakes, implemented in a way that provides significant opportunities for improvement, with a rather superficial discussion of the features supported.

Below lower second-class honours, there is likely to be limited functionality, which likely contains some errors, with significant weaknesses in the implementation, and a discussion that rarely goes beyond bookwork, and/or might be wrong.

Appendix - CW1 template

Please make sure that your implementation uses the function names as specified below.