Cmpe 493 Introduction to Information Retrieval, Fall 2020
Assignment 2 - A Document Retrieval System with Trie
Due: 04/12/2020 (Friday), 17:00

In this assignment you will implement a document retrieval system for single word queries and wildcard queries using the inverted indexing scheme and a trie data structure. You will use the Reuters-21578 data set, which can be download from Moodle (reuters21578.zip). Reuters-21578 contains 21578 news stories from Reuters newswire classified under one or more of 118 categories. There are 22 SGML files, each containing 1000 news articles, except the last file, which contains 578 articles.

You should perform the following steps:

1. Pre-processing the Data Set: The text of a news story is enclosed under the <TEXT> tag. You should use the <TITLE> and the <BODY> fields to extract the text of a news story. Implement your own tokenizer to get the tokens from the news texts and perform normalization operations including case-folding, stopword removal, and punctuation removal. Please use the stopword list on Moodle (stopwords.txt) for stopword removal and you can use the list in "string.punctuation" for punctuation removal for Python.

   Please use "latin-1" encoding while you read the .sgm files due to the corruption in one file.

2. Building the Trie and Inverted Index: Instead of taking each SGML file as a document unit, you should index each news article as a separate document and use the $NEWID$ field as document IDs. Then, you need to generate two files, namely the trie data structure and the inverted index. The first file should contain the trie data structure that is obtained from the news documents. You can store this data structure by pickling. The second file should contain the postings list of each word by document id ($NEWID$). When you implement the query processor, you should use **only** these two files, not the original Reuters-21578 dataset. We will delete the "reuters21578" folder in the query step. Note that the inverted index construction and query processor should be designed as two separate modules. That is, the query processor should NOT construct the inverted index each time it is run. It should just use the trie data structure and the inverted index files generated by the indexing module.

3. Implementing a query processor: You should implement the query processor for single word and wildcard queries. That is, the queries will be of the following two types:
   (i) w_1
   (ii) $prefix_1*$
   where w_1 is a single-word keyword and $prefix_1*$ is a wildcard query that requires you to search documents whether they contain a word that starts with $prefix_1$. You can differentiate these two query types by the last character, $*$.

   The query processor should take as input a query. Here are some example input queries:

   - price
   - pri*

The query processor should return the IDs of the matching documents sorted in **ascending order**.

You can use any programming language of your choice (Python, Java, C++ etc.). We should be able to run your program by following the instructions in your readme file. You have to state the exact commands to prepare the trie and the inverted index, and to query. You should also state the version of the programming language that you used in the readme file. You should not use any third party libraries.

File Structure: The directory should look like in the left figure before running any code. After running the preparation code to create the inverted index and the trie data structure, it should look like in the right figure. (Note that there is no "reuters21578" folder in the right figure, because we delete it for the query processing step. You should only use the inverted index and the trie for query processing.)

```
|
├─ stopwords.txt
├─ prep.py
├─ query.py
├─ HelperClass.py
├─ Report.pdf
├─ readme.txt
├─ ...
└─ reuters21578
    ├─ reut2-000.sgm
    ├─ reut2-001.sgm
    ├─ reut2-002.sgm
    └─ ...
```

```
|
├─ stopwords.txt
├─ prep.py
├─ query.py
├─ HelperClass.py
├─ inverted_index.json
├─ trie.pickle
├─ Report.pdf
├─ readme.txt
└─ ...
```

Submission: You should submit a *".zip"* file named as YourNameSurname.zip containing the following files using the Moodle system:

1. Report:
   (i) Describe the steps you have performed for data preprocessing.

   (ii) Describe the data structures that you used for representing the inverted index.

   (iii) Describe the trie data structure that you used in your code and provide your well-commented trie code in the report.

2. Source code and executable: Commented source code and executables of your document retrieval system.

3. Readme: Detailed readme describing how to run your program including the Python version.

Contact: For questions/comments you can contact Abdullatif Köksal (abdullatifkoksal@gmail.com).

Honor Code: You should work individually on this assignment and all the source code should be written by you. You are NOT allowed to use any available libraries or any code written by other people. Violation of the Honor Code will be strictly penalised, not only by a zero grade from the homework, but also by filing a petition to the Disciplinary Committee.

Late Submission: You are allowed 7 late days (one week) for this assignment with no late penalty. After 7 days, 10 points will be deducted for each late day (unless you have a serious excuse).