Cmpe 493 Introduction to Information Retrieval, Spring 2019
Assignment 1 - A Simple Document Retrieval System for Boolean and Wildcard Queries, Due: 18/03/2019 (Monday), 23:59

In this assignment you will implement a document retrieval system for simple boolean and wildcard queries using the bigram indexing scheme, i.e., k-gram index where k=2 (See IR-Lec3.pdf: Tolerant Retrieval). You will use the Reuters-21578 data set, which is available in the workspace from Moodle (workspace.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 (if available) under the <TEXT> tag to extract the text of a news story. You should read the sgml files with latin-1 encoding to avoid encountering any encoding problems. Implement your own tokenizer to get the tokens from the news texts and perform the following normalization operations: case-folding, stop-word removal. You **must** use the stopword and punctuation lists on Moodle in workspace.zip. Note that you should perform the same preprocessing steps for the queries as well. Please follow these steps **exactly**:

    - Tokenizer: Replace punctuation marks with space and split text from space to get the tokens.
    - Case-Folding: Lowercase all tokens.
    - Stopword Removal: Remove all stopwords included in the list provided.

2. Building the Bigram 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 create (write) two files. The first file should contain the inverted indexes of tokens. It **must** be a json file called index.json which looks like this:

```
{
"aardvark": [3, 21, 44],
"aardwolf": [1, 22, 55],
.
.
.
"zebra": [4, 17, 21]
}
```

The second file should contain the bigram mappings for the unique tokens (i.e., terms). It **must** be a json file called bigrams.json which looks like this:

```
{
"$b": ["bike", "broke", "bull"],
"et": ["beetroot", "metric", "petrify"],
.
```

.
.
”zz”: [”buzzer”, ”fizz”, ”jazz”]
}

When you implement the Boolean query processor, you need to use **only** these two files, not the Reuters-21578 dataset. Your code should generate these files to the Output folder in workspace.

3. Implementing a Boolean query processor: You should implement a query processor which returns the IDs of documents for conjunctive, disjunctive, and wildcard queries. You do NOT need to handle proximity queries, phrase queries, or queries containing the NOT operator or parenthesis. That is, the queries will be of the following three types:
   (i) w_1 AND w_2 AND w_3... AND w_n
   (ii) w_1 OR w_2 OR w_3... OR w_n
   (iii) w_*
   where each w_i is a single-word keyword without wildcard character and w_* is a single-word keyword with one wildcard star character for example, ”bo*”, ”*ar”, or ”cl*ss”. Don’t forget the **postfilter operation** for wildcard queries to eliminate false positive tokens. Also, none of the w_is is the ”AND” or ”OR” words.

   Your program should take a query type (1, 2, or 3) and a query string as arguments and should assume that the index.json and bigrams.json files are in the Outputs folder. For example,

   ```
   python process.py 1 "people AND car"
   ```

   You need to return the **sorted list of document IDs** as a result to **the command line**.

You are required to use **Python** to implement your program. Note that you are **NOT** allowed to use any third party libraries.

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

-Assignment 1
  -Code
    Your source code and executable files should be in a folder named as **Code**
  Report.pdf
  README.txt

You should delete the punctuations.txt, stopwords.txt, bigrams.json, index.json files, as well as reuters21578 folder prior to submitting your work through Moodle.

1. Report:
   (i) Describe the steps you have performed for data preprocessing and provide answers for the following questions.

(a) How many tokens does the corpus contain before stopword removal?
(b) How many tokens does the corpus contain after stopword removal?
(c) How many terms (unique tokens) are there before stopword removal and case-folding?
(d) How many terms (unique tokens) are there after stopword removal and case-folding?
(e) List the top 20 most frequent terms before stopword removal and case-folding?
(f) List the top 20 most frequent terms after stopword removal and case-folding?

(ii) Provide a screenshot of running your system for a conjunctive query.

(iii) Provide a screenshot of running your system for a disjunctive query.

(iv) Provide a screenshot of running your system for a wildcard query.

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 version of the programming language).

Grading:

1. Report+README: 10 points
2. bigrams.json: 10 points
3. index.json: 20 points
4. Queries: 60 points

You must use the specified steps for tokenization and normalization to ensure your work is graded correctly.

Late Submission: You are allowed a total of 5 late days on homeworks with no late penalties applied. You can use these 5 days as you wish. For example, you can submit the first homework 2 days late, then the second homework 3 days late. In that case, you will have to submit the remaining homeworks on time. After using these 5 extra days, 10 points will be deducted for each late day. Note than, if submission deadline is 23:59 o'clock, a submission one minute later (at 00:00 o'clock) is considered one day late submission.