Assignment 3: Retrieval Models

**Instruction**

In this assignment, you will have two tasks: **automatically translate topic statements to queries**, and **implementing the statistical language model** discussed in the classes.

**Tasks**

### Task 1: Automatically translate topic statements to queries

For a given set of topic statements, you should have a module to translate the topic statement information into a set of queries that can be recognized by your retrieval module, in which each query corresponds to a search topic and consists of a query content and a query id. Tokenization, normalization and stop-word removing should be conducted on each query.

*Note: Remember that TREC like topic statements have title, description, and narrative sections. Your module should have the ability to pick up any combination of them (i.e., title only, description only, title and description, all of them). But for the assignment, you can simply pick up title only for query. Of course, you can use description and narrative too.* ***Remember that queries in general have multiple query terms****.*

In this task, you should implement:

* **Search.ExtractQuery**

Queries are extracted and preprocessed from TREC style topic file topics.txt.

* **Classes.Query**

This class stores query information, including the topic id and a representation of the query. If you need to store other information about the query, you CAN MODIFY Classes.Query.java.

The topic file “topics.txt” contains four TREC style topics. Note that the forth topic contains a word “Dysphagia” that did not appear in the collection. You should detect and process such cases.

### Task2: Implementing the Statistical Language Model

Your implementation should be able to read the index you built in assignment 2 using document collection file “docset.trectext” (you should use our provided implementation of index builder), and return documents based on the ranking of the documents generated by your retrieval model. The retrieval model is the query likelihood model with Dirichlet Prior Smoothing.

**Note**: The Dirichlet smoothing method can be found in the slide of lecture *Unit 5:Matching models- probabilistic and language model.* You can find more details from Professor Chengxiang Zhai’s work in SIGIR 2001: *A Study of Smoothing Methods for Language Models*. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.65.8943&rep=rep1&type=pdf>. In the statistical language model, *miu* (µ) is an important parameter. ***Provide a short paragraph, telling us how to tune the parameter µ and its effect to the query result.***

In this task, you should implement:

* **Search.QueryRetrievalModel**

In this class, you should implement the method **retrieveQuery(Query aQuery, int TopN)**, which retrieves the input query and returns the top N retrieved documents as a list of Classes.Document objects.

* **IndexingLucene.MyIndexReader**

If more indexing information is needed, you can modify this class with STANDARD JAVA API.

Returned documents for each query should be ranked in decreasing order of their scores, and they should be in the following format.

queryid Q0 documentid rank score runid

For example, in the run with an id TEST, for query 40004, document APW20010120.0310.0146 is ranked as number 1 with score 0.65, then the content format for this document is as following. Here Q0 is just a dummy filler, you put it all for all the lines. All the results for every query are saved in one single file.

40004 Q0 APW20010120.0310.0146 1 0.65 TEST

**Requirements and Reminders**

* You CAN ONLY use Java to finish this assignment.
* In your own code, you CANNOT use any Java API library other than the standard JDK (for example, you cannot use apache commons, apache Lucene, indri etc. in this assignment).
* Feel free to use IDE tools such as Eclipse and Netbeans.
* Do tell us the Java version you used for writing your assignment, e.g. JDK 1.8 or 1.7. Please only use Oracle JDK or openjdk.
* You CANNOT modify anything in class HW3Main, package Classes except Query.java, and package IndexingLucene except MyIndexReader.java. Path.java has been updated with one more address, so please replace the old one with this new one.

**Grading**

Your submission will be graded based on:

* Correctness of the implementation on the required functions (70%)
* Efficiency of your implementation, make sure your code finish 4 query search within 2 minutes (20%)
* Necessary program annotation and commentaries (10%)
* A good exploration to the parameter µ (10%)

## Submission Requirements

A zipped file package with the naming convention as “pittids\_2140a3”. For example, suppose the CSSD id is jud1, then the submission package should be jud1\_2140a3.zip.

You can submit your assignment in **CourseWeb>>Assginments>>Assginment 3**.

The file package should contain:

1. src folder, which contain all your codes.
2. A txt file describing:
   1. how long it takes to finish 4 queries
   2. retrieval result of 4 queries
   3. a paragraph on how to tune µ, and its effect on the query results.
   4. Java version you used for writing your assignment.