Implementation Assignment

You are required to implement the Vector Space Model of Information Retrieval. You must create a program (using Java) that, using the files in the archive Assignment_Files.zip , can do the following:

● Parse the documents contained in the Cranfield document collection : each document is contained in the format:

.I # The Index number of the Document

.T

Title of the document

.A

Author / Authors names

.B

Bibliography, reference of where the paper is published

.W

Text of the document goes here

There are no breaks or empty lines between the end of one document and the beginning of the next.

● Perform the common pre-processing steps: stopword removal and stemming. A list of stopwords to use is available on Moodle. For stemming, you must use any of the official implementations of Porter's Stemming Algorithm available from

http://tartarus.org/~martin/PorterStemmer/

The Java version is included in the assignment files archive.

● Create an appropriate index so that IR using the Vector Space Model may be performed. This will require you to calculate the relevant TF-IDF weights. This may be stored in memory or in an external file. You may NOT use database systems such as MySQL, SQL Server, Oracle or similar.

● Accept a query on the command line and return a list of the 100 most relevant documents,

according to the Vector Space IR Model sorted beginning with the most relevant. The output should have three columns: the rank, the document's ID and the similarity score. Sample output is:

1 109 0.675

2 76 0.663

3 3 0.634

4 934 0.603

5 32 0.583

...

100 1390 0.135

● It is ESSENTIAL that this can be run as a standalone program, without requiring IDEs such as Visual Studio, Eclipse, NetBeans or similar.

The application should be made in two parts:

part 1: preparation. At the end of preparation your application should produce a text file that contains the length of each document in the corpus.

part 2: Querying the dataset where each query outputs a file that contains the ranked output. Each query output should be in a separate file with no requirement to rename the file produced before another query is run.

Part 2 should be callable without having to run part one each time.

Grading:

The grade will be based on

1. the "correctness" of your output
2. the speed of execution of part 1 in preparing the corpus for query
3. the speed at which results are returned for three queries

4. the "correctness" of your code : it MUST compile from command line

5. The usefulness of your Readme describing compilation procedure and instructions on running the compiled application.

Each submission will be tested in a virtual machine running

Java version 1.8.0_131 (later versions can be installed if needed)

The test system is Linux running on 2GB RAM (about 1.5GB Free) with NO INTERNET CONNECTION and NO GRAPHIC USER INTERFACE!

YOUR APPLICATION MUST RUN FROM THE COMMAND LINE / SHELL