Assignment 1: Document Collection Processing

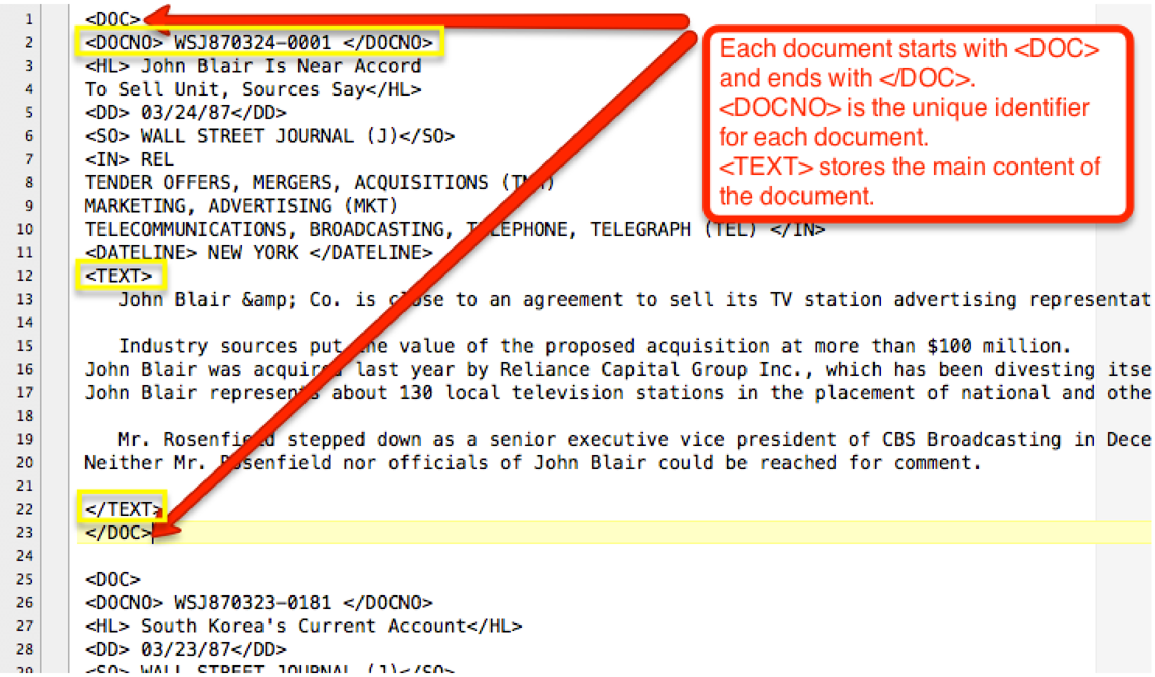
Instruction

The goal of the assignment is to develop Java classes that can process TREC standard format document collections. You need to follow the instructions to implement the required classes for processing the document collection files.

In information retrieval (IR) experiments, a document is usually the basic unit to be indexed and retrieved, such as a webpage or a pure text file. To simply the processing of documents, we usually compile small document files into large collection files. Each single collection file contains the contents of many small documents. trectext and trecweb are two popular document collection formats. In this assignment, we need to write programs to process documents stored in these two types of collections.

The trectext format is widely used to store textual documents such as news articles. Figure 1 shows a sample snippet of a trectext format collection file: each document starts with <DOC> and ends with </DOC>; <DOCNO> stores the unique identifier of each document; <TEXT> stores the main content of the document.

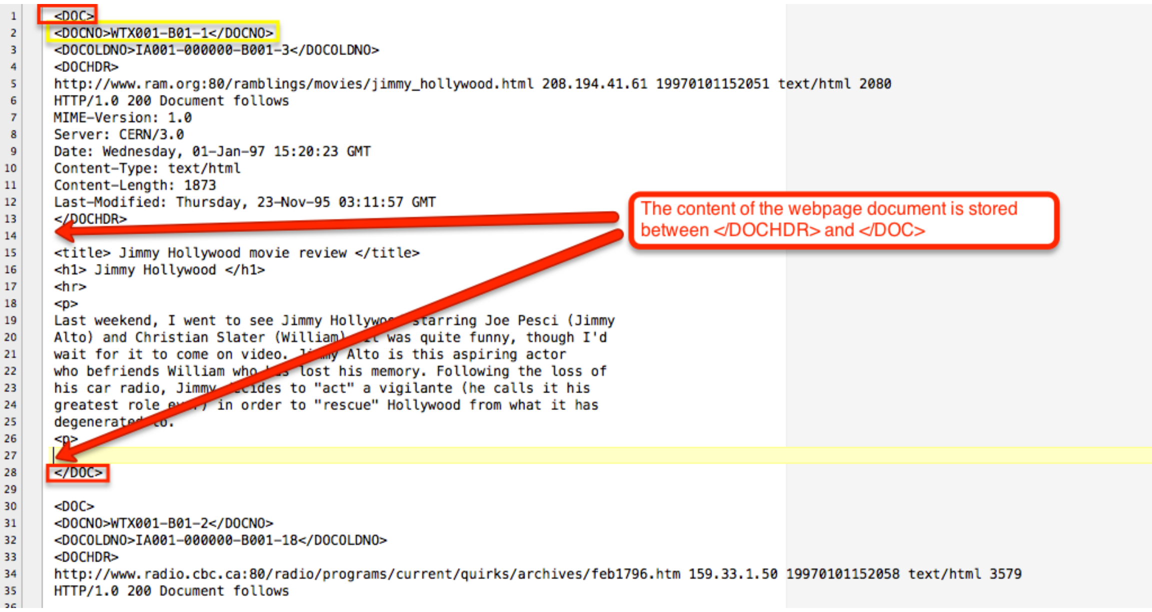
The trecweb format is used to store web documents. Figure 2 shows a sample snippet of a trecweb format collection file: each document starts with <DOC> and ends with </DOC>; <DOCNO> stores the unique identifier of each document; <DOCHDR> usually stores the http response header information of accessing the web document; the content of the web document is stored between </DOCHDR> and </DOC>.



```
<DOC>
<DOCNO> WSJ870324-0001 </DOCNO>
<HL> John Blair Is Near Accord
To Sell Unit, Sources Say</HL>
<DD> 03/24/87</DD>
<SO> WALL STREET JOURNAL (J)</SO>
<IN> REL
TENDER OFFERS, MERGERS, ACQUISITIONS (TNM)
MARKETING, ADVERTISING (MKT)
TELECOMMUNICATIONS, BROADCASTING, TELEPHONE, TELEGRAPH (TEL) </IN>
<DATELINE> NEW YORK </DATELINE>
<TEXT>
   John Blair &amp; Co. is close to an agreement to sell its TV station advertising representat

   Industry sources put the value of the proposed acquisition at more than $100 million.
John Blair was acquired last year by Reliance Capital Group Inc., which has been divesting itse
John Blair represents about 130 local television stations in the placement of national and othe

   Mr. Rosenfield stepped down as a senior executive vice president of CBS Broadcasting in Dece
Neither Mr. Rosenfield nor officials of John Blair could be reached for comment.

</TEXT>
</DOC>

<DOC>
<DOCNO> WSJ870323-0181 </DOCNO>
<HL> South Korea's Current Account</HL>
<DD> 03/23/87</DD>
<SO> WALL STREET JOURNAL (J)</SO>
```

Each document starts with <DOC> and ends with </DOC>.
<DOCNO> is the unique identifier for each document.
<TEXT> stores the main content of the document.

Figure 1: A sample snippet of a trectext format collection file.



```
<DOC>
<DOCNO>WTX001-B01-1</DOCNO>
<DOCOLDNO>IA001-000000-B001-3</DOCOLDNO>
<DOCHDR>
http://www.ram.org:80/ramblings/movies/jimmy_hollywood.html 208.194.41.61 19970101152051 text/html 2080
HTTP/1.0 200 Document follows
MIME-Version: 1.0
Server: CERN/3.0
Date: Wednesday, 01-Jan-97 15:20:23 GMT
Content-Type: text/html
Content-Length: 1873
Last-Modified: Thursday, 23-Nov-95 03:11:57 GMT
</DOCHDR>

<title> Jimmy Hollywood movie review </title>
<h1> Jimmy Hollywood </h1>
<hr>
<p>
Last weekend, I went to see Jimmy Hollywood, starring Joe Pesci (Jimmy
Alto) and Christian Slater (William). It was quite funny, though I'd
wait for it to come on video. Jimmy Alto is this aspiring actor
who befriends William who has lost his memory. Following the loss of
his car radio, Jimmy decides to "act" a vigilante (he calls it his
greatest role ever) in order to "rescue" Hollywood from what it has
degenerated to.
<p>

</DOC>

<DOC>
<DOCNO>WTX001-B01-2</DOCNO>
<DOCOLDNO>IA001-000000-B001-18</DOCOLDNO>
<DOCHDR>
http://www.radio.cbc.ca:80/radio/programs/current/quirks/archives/feb1796.htm 159.33.1.50 19970101152058 text/html 3579
HTTP/1.0 200 Document follows
```

The content of the webpage document is stored between </DOCHDR> and </DOC>

Figure 2: A sample snippet of a trecweb format collection file.

Tasks

Task 1: Reading Documents from Collection Files

In this task, you should implement two classes that can read individual documents from trectext and trecweb format collection files (you can find the classes and detailed descriptions in src.zip):

- **PreProcessData.DocumentCollection** is a general interface for sequentially reading documents from collection files
- **PreProcessData.TrectextCollection** is the class for trectext format
- **PreProcessData.TrecwebCollection** is the class for trecweb format

Task 2: Normalize Document Texts

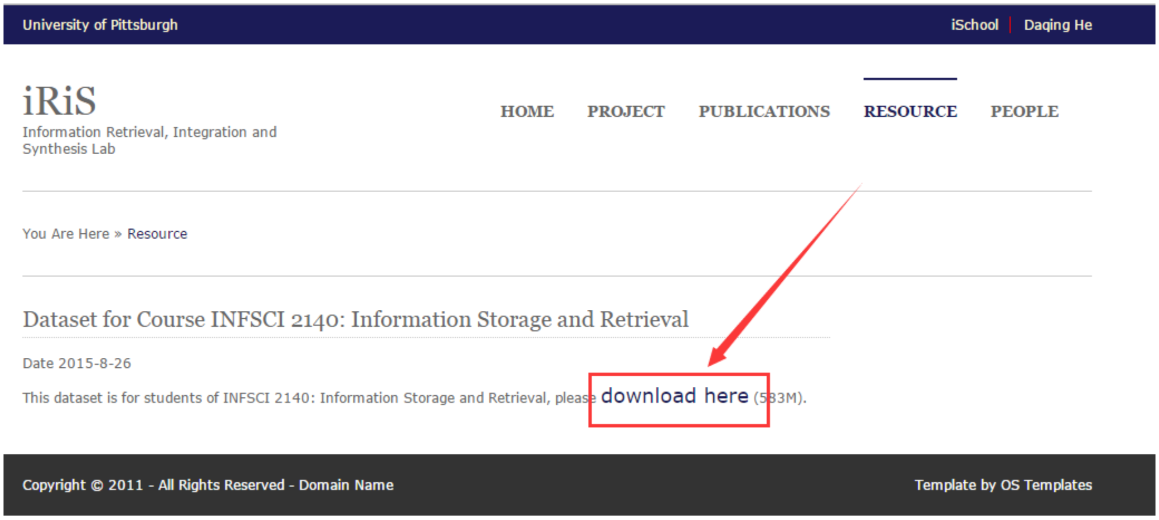
In this task, you should first implement classes to tokenize document texts into individual words, normalize all the words into their lowercase characters, and finally filter stop words.

- **PreProcessData.TextTokenizer** is a class for sequentially reading words from a sequence of characters
- **PreProcessData.TextNormalizer** is the class that transform each word to its lowercase version, and conduct stemming on each word.
- **PreProcessData.StopwordsRemover** is the class that can recognize whether a word is a stop word or not. A stop word list file will be provided, so that the class should take the stop word list file as input.

The 6 classes to be implemented in task1 & 2 can be found in src.zip. You **CANNOT** change the classes' names or the required methods' names. However, you can add new variables, constants, and methods in these classes and create new classes if necessary.

HW1Main is the main class for running your assignment 1. You can find the class in src.zip, and you are **NOT** allowed to change anything in this file. If you have successfully implemented the 6 classes in task 1&2, you should be able to directly run HW1Main, which can read an input collection file and output the normalized version of each document in the collection as another file.

Classes.Path contains addresses of all input and output files, so you should put all files in the corresponding directory. **Classes.Stemmer** is the stemmer that you will use to normalize each word, and you can learn how to use this stemmer through its main method. These two classes are **NOT** allowed to change, too.



University of Pittsburgh | iSchool | Daqing He

iRiS
Information Retrieval, Integration and Synthesis Lab

You Are Here » Resource

Dataset for Course INFSCI 2140: Information Storage and Retrieval

Date 2015-8-26

This dataset is for students of INFSCI 2140: Information Storage and Retrieval, please download here (583M).

Copyright © 2011 - All Rights Reserved - Domain Name

Template by OS Templates

Figure 3: Download the collection files.

Two collection files and stopword.txt are provided in http://crystal.exp.sis.pitt.edu:8080/iris/resource.jsp , as shown in Figure 3. docset.trectext is a trectext format file and docset.trecweb is a trecweb format one. Your programs should at least be able to successfully process these two documents.

Requirements and Reminders

- You CAN ONLY use Java to finish this assignment.
- 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 1.6 or 1.7. Please only use Oracle JDK or openjdk.

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 processing two collections within 10 minutes (20%)
- Necessary program annotation and commentaries (10%)

Submission Requirements

A zipped file package with the naming convention as “pittids_2140a1”. For example, suppose the Pitt id is jud1, then the submission package should be jud1_2140a1.zip.
The file package should contain:

1. All the scripts/programs you used for this assignment (**src folder**)
2. A short instruction on how to read your scripts and how to run your scripts, including environment configuration. (This should be a **txt file**.)
3. A short instruction on how long it takes to finish running your code, and how many documents of two corpus are processed. You will **NOT** upload your result file, so make sure your code finish running within 10 minutes; if cannot, you must make it clear whether the code will generate the result, and how much time it costs to finish running. (This should be a **txt file**.)