University of Texas at Dallas
CS 6322 : Information Retrieval
Fall 2011
Instructor: Dr. Sanda Harabagiu
Grader: Bryan Rink
Issued: October 3rd 2011
Due October 26th 2011 before midnight

Problem (200 points)
Indexing

In this assignment you build the index for a simple statistical retrieval system. In the next assignment, you will build the retrieval system itself. For this assignment, index the Cranfield documents used in the last assignment:
A copy of the publicly available Cranfield collection is located on the UTD Apache machine at:
/people/cs/s/sanda/cs6322/Cranfield/

Do not store stop-words in your index. You may use your own stop-word list or the list from the directory on the UTD Apache machine at:
/people/cs/s/sanda/cs6322/Cranfield/resourcesIR.

The terms in your index should be stemmed with the Porter stemmer. The code for the Porter stemmer is available in the same directory on the UTD Apache machine at:
/people/cs/s/sanda/cs6322/Cranfield /resourcesIR.

For every term that is indexed, store:
- Document frequency (df): The number of documents that the term occurs in.
- Term frequency (tf): The number of times that the term occurs in each document, and
- The list of documents containing the term.

For each document, store the frequency of the most frequent stem in that document (max_tf), and the total number of word occurrences in the document, including stop-words (doclen).

Store the inverted lists in your own storage manager. Also, in a second version of your index, compress the inverted lists before storing them, using delta encoding for the document-id and gamma code for the frequency information. A penalty of -100 points will be applied if you do not have also a version of the compressed index, obtaining only max 100 points in this homework.

Delta codes are similar to the gamma codes: they represent a gap by a pair: (length, offset). First the number is represented in binary code. The length of the binary representation is encoded in gamma code, prior to removing the leading 1-bit. After generating the code of the length only, the leading 1-bit is removed and represented in gamma code.

Example 1: To write 5 in gamma and delta codes we perform the following operations:

1. write 5 in binary as 101
2. For the gamma code remove the leading 1-bit to obtain the offset: 01
3. The length of the offset is 2:
    - In unary the length is 110
4. The code of 5 in gamma is 11001 (or 110,01 – to represent [length, offset])
5. For the delta code, the length of the offset is 3, because the leading 1-bit is removed afterwards. When writing the length=3 in gamma code it becomes:10,1
6. for delta code, the leading 1-bit of the offset is removed now, generating an offset of 10
7. The code of 5 in delta is 10101

Example 2: To write 9 in gamma and delta codes we perform the following operations:

1. write 9 in binary as 1001
2. for gamma code, remove the leading 1-bit to obtain the offset: 001
3. The length of the offset is 3:
    - In unary the length is 1110
4. The code of 9 in gamma is 1110001 (or 1110,001 – to represent [length, offset])
5. for the delta code, the length of the binary representation is 4
6. The length is represented in gamma code: 11000
7. The leading 1-bit of the binary representation is removed, generating the offset 001
8. The code of 9 in delta is 11000001

Example 3: To write 1 in gamma and delta , we perform the following operations:

1. write 1 in binary: 1
2. For the gamma code, we remove 1, generating a length=0, which is still 0 in unary code
3. in Gamma code, 1 becomes 0
4. for the Delta code, the length of the binary representation for 1 is 1. Gamma code for 1 is 0
5. The code for 1 in delta is 0

More values for gamma and delta codes are given in the following table:

N	Binary	Gamma code	Delta code
1	1	Len(-)=0; unary(0)=0; Gamma(1)=0	Len(1)=1; Gamma(1)=0; Delta(1)=0
2	10	Len(0)=1; unary(1)=10; Gamma(2)=100	Len(10)=2; Gamma(2)=100; Delta(2)=1000
3	11	Len(1)=1; unary(1)=10; Gamma(3)=101	Len(11)=2; Gamma(2)=100; Delta(3)=1001
4	100	Len(00)=2; unary(2)=110; Gamma(4)=11000	Len(100)=3; Gamma(3)=101; Delta(4)=10100
5	101	Len(01)=2; unary(2)=110; Gamma(5)=11001	Len(101)=3; Gamma(3)=101; Delta(5)=10101
6	110	Len(10)=2; unary(2)=110; Gamma(6)=11010	Len(110)=3; Gamma(3)=101; Delta(6)=10110
7	111	Len(11)=2; unary(2)=110; Gamma(7)=11011	Len(111)=3; Gamma(3)=101; Delta(7)=10111
8	1000	Len(000)=3; unary(3)=1110; Gamma(8)=111000	Len(1000)=4; Gamma(4)=11000; Delta(8)=11000000
9	1001	Len(001)=3; unary(3)=1110; Gamma(9)=111001	Len(1001)=4; Gamma(4)=11000; Delta(9)=11000001
10	1010	Len(010)=3; unary(2)=1110; Gamma(10)=1110010	Len(1010)=4; Gamma(4)=11000; Delta(10)=11000010

Production-level IR systems build these compressed indices in about 5 minutes. If your program takes more than an hour, you are doing something wrong.

Turn in your program, a written description of your program, and the following statistics:

- the elapsed time ("wall-clock time") required to build your index,
- the size of the index uncompressed (in bytes),
- the size of the index compressed (in bytes),
- the number of inverted lists in the index, and
- the df, tf, and inverted list length (in bytes) for the terms:
  "Reynolds", "NASA", "Prandtl", "flow", "pressure", "boundary", "shock" (or stems that correspond to them).