COL764
Inverted Index Construction

Manoj Kumar
2018CS50411

Introduction: This report documents the implementation details related to the Corpus Pre-processing, Inverted Index Construction, Compression Strategies, Mode of Query Retrieval etc. used in the assignment which when integrated constitutes a full Boolean Retrieval System.

List Compression Models Implemented:

c0 : normal copression with 4 bytes to each integer with gap encoding.
c1 : works with chunks of 7 bits.
c2 : encoding each integer with O(logx) bits
c3 : encoding using google's fast general-purpose compression library called snappy.
(Bonus) c4 : using unary and binary representation and a manually selected integer k.

Files in Submission:

invidx_cons.py: the construction of the Inverted Index, by first processing the
documents and then storing them on the disk in a compressed way.
query.py: preforms the query processing and retrieval tasks.
helper.py: helper functions for both main.py and query.py are implemented here.
Stemmer.py: class file for stemming

Operations in invidx_cons.py:

1. parsing of all xml files

2. tokenizing and stemming
3. mapping of docno to an integer index
3. store the data of each word and their posting list in the disk using many chunk files.
4. read the disks and compress posting lists for each word according to the given compression type.

We are storing the following information in indexfile.dict file:

1. compression type
2. next lines are of the following format:

word starting_index_in_idx_file length(in bytes)

until '!' appears in the whole line

3. next lines are the docnos where ith docno has a mapping to no. `i`
4. docno's continue until `!` appears.
5. now remaining last line contains space separated stopwords.

5. Then we are compressing the data according to the compression strategy.
6. Gap encoding is done before compression.

So basically, we are storing the following information in indexfilename.dict file:

1. compression type
2. starting byte and length for posting lists of each word.
3. mapping of docno to integer indices.
4. stopwords

Operations in query.py:

1. Reading indexfilename.dict file and getting compression type using the first read line.
2. Store the dictionary words and their starting byte in idx file with their length in the memory.
3. extract mapping of docno to index integers.
4. extract stopwords
5. read queries

6. for each single word query:
    * check if the word is in stopwords, if yes, return null
    * if word is not in stopwords, decode its posting list with the respective compression type and return the result.
7. for each multiple word query:
    * check if the word is in stopwords, if yes, move to the next word
    * if word is not in stopwords, decode its posting list with the respective compression type and if it is the first word of the query, store its posting list to final list and move to the next, else take intersection from the final posting list.
8. print the results

Observations:

The following metrices are being evaluated on the dataset provided and the sample query file:

1. **Index Size Ratio (ISR):**
    compression type 0 : (155.8MB)/514.9MB = 0.3
    compression type 1 : (50.5MB)/514.9MB = 0.09
    compression type 2 : (40.7MB)/514.9MB = 0.08
    compression type 3 : (83.5MB)/514.9MB = 0.16
    (Bonus) compression type 4 : (105.5MB)/514.9MB = 0.2 for k = 6

2. **Compression Speed:**
    compression type 0: 21 minute
    compression type 1 : 20 minute
    compression type 2 : 20 minute
    compression type 3 : 21 minute
    (Bonus) compression type 4 : 20 minute

3. **Query Speed:**
    compression type 0 : 0.15 min per query

compression type 1 : 0.106 min per query
compression type 2 : 0.12 min per query
compression type 3 : 0.14 min per query
(Bonus) compression type 4 : 0.22 min per query

* The Index Size Ratio (ISR):
  mininmum for c2 maximum for c0
* Compression Speed:
  minimum for c1 and c2 maximum for c0 and c3
* Query Speed:
  minimum for c1 maximum for c4

* all posting lists are gap encoded before compressed.

Thanks
Manoj Kumar
4th Year Dual Undergraduate
Computer Science and Engineering, IIT Delhi