CS 6322: Information Retrieval

Ashika Prakash Acharya
axa190084

HomeWork- 2

Program Description

This program creates two versions of the index, each version of the index shall be in uncompressed form and compressed form.

1. **Index_Version1.uncompress**
2. **Index_Version2.uncompress**

PART 1: Index Construction

Index_Version1.uncompress considers the terms in the dictionary to be lemmas of words and the lemmas were obtained using **WordNetLemmatizer** from **nltk.stem** module.
Index_Version2.uncompress considers that the dictionary consists of stems of the tokens that was processed in Homework 1. **PorterStemmer** from **nltk.stem** module was used to obtain the stems of the tokens.

Before building the dictionaries of any of the two versions of index, we have removed the stop words. The stop word list was obtained at:

/people/cs/s/sanda/cs6322/Cranfield/resourcesIR

Design and Algorithm

The program starts reading all the files and does the following task in the mentioned order for each file that is read.

- Performs the preprocessing of the file.
  - Remove tags, stop words, special characters, symbols. [**file_preprocess()**]
  - Keep track of the number of stop words removed.
- Stemming of the raw tokens. [**get_list_of_stems()**]
  - For each stem, we either create a posting list or update the posting list of a dictionary named **dict_of_stems < stem, (posting_list)>**
    posting_list = (df, [(doc_id, tf, max_tf, doc_len), (), () …. ()])
  - Also, time is measured for stemming of each file.
- Lemmatizing of the raw tokens.

- For each lemma, we either create a posting list or update the posting list of a dictionary named **dict_of_lemmas < lemma, (posting_list)>**
  posting_list = (df, [(doc_id, tf, max_tf, doc_len), (), () …. ()])
- Also, time is measured for stemming of each file.

- The dictionary **dict_of_stems and dict_of_lemmas** are sorted and saved as Index_Version2.uncompress.txt and Index_Version1.uncompress.txt respectively.

- This approach of index construction follows '**Single-pass in-memory indexing (SPIMI)**' algorithm. The algorithm generated separate dictionary for each block(file) and then accumulates them and sorts.

PART 2: Index Compression

The compression is achieved by compressing terms and posting files of the index separately. When compressing the inverted lists, we have considered only the document ID, discarding tf, max_tf and doclen.

Design and Algorithm

Index_Version1.compress

In Index_Version1.compressed, we have compressed the dictionary using blocked storage compression technique with k = 4 and for the posting files, we have compressed using gamma encoding for the gaps between document-ids.

- For the dictionary, each word is kept in a buffer and a count is incremented. When the count is k=4, the buffer is flushed out to a file **Index_Version1.compress** appended with its length and the count is set to 0.
- For the posting files of each term, gaps are calculated and passed as an argument to function **get_gamma_codes().**
- The gamma codes are then encoded to utf8 and written as a byte array to the same compressed file **Index_Version1.compress.**

Index_Version2.compress

In Index_Version2.compressed, we have compressed the dictionary with front-coding with the block size k=8 and for the posting files, we have used delta codes to encode the gaps between document-ids.

- For every block k = 8, we have found out the longest common prefix and encoded the rest according to the algorithm. They are then written to file **Index_Version2.compress.**

- For the posting files of each term, gaps are calculated and passed as an argument to function **get_delta_codes().**
- The delta codes are then encoded to utf8 and written as a byte array to the same compressed file **Index_Version2.compress.**

Statistics

- **The elapsed time ("wall-clock time") required to build any version of your index,**
  Time taken to build Index_Version1 is 3.7747721672058105 seconds
  Time taken to build Index_Version2 is 4.868532657623291 seconds.

- **The size of the index Version 1 uncompressed (in bytes),**
  Size of Index_Version1_uncompress is 2480798 bytes

- **The size of the index Version 2 uncompressed (in bytes),**
  Size of Index_Version2_uncompress is 2380794 bytes

- **The size of the index Version 1 compressed (in bytes),**
  Size of Index_Version1_compress is 937935 bytes

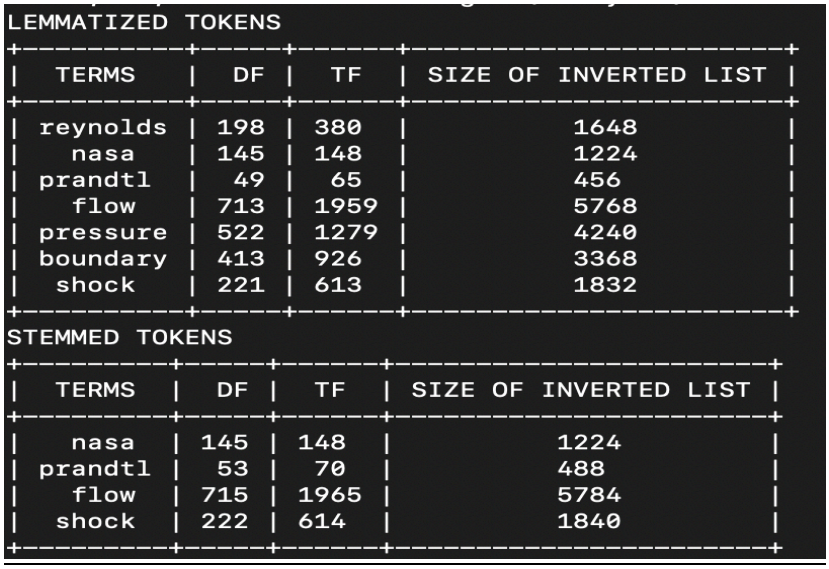
- **The size of the index Version 2 compressed (in bytes),**
  Size of Index_Version2_compress is 822339 bytes

- **The number of postings in each version of the index,**
  Number of postings in Index_Version1 is 11787
  Number of postings in Index_Version2 is 9874

- **The df for "NASA" as well as tf, the doclen and the max_tf, for the first 3 entries in the posting list**

```
9. DF of 'nasa' : 145 First 3 entries in posting list for lemmatized token 'nasa'
+------+--------+----+--------+------------+
| Term | Doc_id | TF | Max_TF | Doc length |
+------+--------+----+--------+------------+
| nasa |  0053  | 1  |   11   |    212     |
| nasa |  0058  | 1  |   6    |    179     |
| nasa |  0068  | 1  |   3    |    114     |
+------+--------+----+--------+------------+
DF of 'nasa' : 145 First 3 entries in posting list for stemmed token 'nasa'
+------+--------+----+--------+------------+
| Term | Doc_id | TF | Max_TF | Doc length |
+------+--------+----+--------+------------+
| nasa |  0053  | 1  |   11   |    212     |
| nasa |  0058  | 1  |   6    |    179     |
| nasa |  0068  | 1  |   4    |    114     |
+------+--------+----+--------+------------+
```

- **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)**



LEMMATIZED TOKENS

TERMS	DF	TF	SIZE OF INVERTED LIST
reynolds	198	380	1648
nasa	145	148	1224
prandtl	49	65	456
flow	713	1959	5768
pressure	522	1279	4240
boundary	413	926	3368
shock	221	613	1832

STEMMED TOKENS

TERMS	DF	TF	SIZE OF INVERTED LIST
nasa	145	148	1224
prandtl	53	70	488
flow	715	1965	5784
shock	222	614	1840

- **The dictionary term from index 1 with the largest df and the dictionary term with the lowest df – _if more than one list them all!!!**

  Term from Index_Version1 with largest DF is ['to']
  Number of terms from Index_Version1 with smallest DF is 6421 and is available in file Index1_smallest_df

- **The stem from index 2 with the largest df and the dictionary term with the lowest df –_if more than one list them all!!!**

  Term from Index_Version2 with largest DF is ['to']
  Number of terms from Index_Version2 with smallest DF is 5522 and is available in file Index2_smallest_df

- **The document with the largest max_tf in collection and the document with the largest doclen in the collection**

  Document with largest MAX_TF for Index_Version1 is 0673
  Document with largest Doc_len for Index_Version1 is 1313
  Document with largest MAX_TF for Index_Version2 is 0673
  Document with largest Doc_len for Index_Version2 is 1313