Information Retrieval
Topic- Index Construction (SPIMI)
Lecture-18

Prepared By

Dr. Rasmita Rautray & Dr. Rasmita Dash
Associate Professor
Dept. of CSE

Content

- Limitations of BSBI
- Single-pass in-memory indexing (SPIMI)

Problem with BSBI algorithm

- Our assumption was: we can keep the dictionary in memory.
- We need the dictionary (which grows dynamically) in order to implement a term to termID mapping.
- Actually, we could work with term,docID postings instead of termID,docID postings . . .
- . . . but then intermediate files become very large. (We would end up with a scalable, but very slow index construction method.)

Single-pass in-memory indexing (SPIMI)

- Key idea 1: Generate separate dictionaries for each block – no need to maintain term-termID mapping across blocks.
- Key idea 2: Don’t sort. Accumulate postings in postings lists as they occur.
- With these two ideas we can generate a complete inverted index for each block.
- These separate indexes can then be merged into one big index.

SPIMI Algorithm

```
SPIMI-INVERT(token_stream)
 1  output_file = NEWFILE()
 2  dictionary = NEWHASH()
 3  while (free memory available)
 4  do token ← next(token_stream)
 5     if term(token) ∉ dictionary
 6        then postings_list = ADDTODICTIONARY(dictionary, term(token))
 7        else postings_list = GETPOSTINGSLIST(dictionary, term(token))
 8     if full(postings_list)
 9        then postings_list = DOUBLEPOSTINGSLIST(dictionary, term(token))
10     ADDTOPOSTINGSLIST(postings_list, docID(token))
11  sorted_terms ← SORTTERMS(dictionary)
12  WRITEBLOCKTODISK(sorted_terms, dictionary, output_file)
13  return output_file
```

- As we build index, docs are parsed one at a time and turns them into a stream of term–docID pairs, which we call *tokens here, has been omitted.*
- *SPIMI-INVERT is called repeatedly on* the token stream until the entire collection has been processed.
- In line 4, Tokens are processed one by one during each successive call of SPIMI-INVERT.
- When a term occurs for the first time, it is added to the dictionary (best implemented as a hash), and a new postings list is created in line 6.

- In line 7, The call returns this postings list for subsequent occurrences of the term.
- SPIMI adds a posting directly to its postings list in line 10.
- Instead of first collecting all termID–docID pairs and then sorting them(as we did in BSBI), each postings list is dynamic (i.e., its size is adjusted as it grows) and it is immediately available to collect postings.

Advantage of SPIMI

- Generate separate dictionaries for each block – no need to maintain term-termID mapping
- Don’t sort

Thank You