Index construction

Paolo Boldi
DSI
LAW (Laboratory for Web Algorithmics)
Università degli Studi di Milan

The problem

Given a collection of documents, we have to parse them and build the inverted index.

- The *inverted index* is made up by:
  - A data structure (the *dictionary*) that maps each term to a termID
  - For each termID, an *inverted list* that contains the docID's of the documents that contain that term (in increasing order of docID)
  - Possibly, the inverted list may also contain: the term frequency or other information (positions where the term appears in the document...).

Block Sorted Base Indexing (BSBI)

$T \leftarrow$ empty hashtable
$B \leftarrow$ empty block
for all documents **do**
  $dID \leftarrow$ docID
  for every term t in the document **do**
    add t to T, if necessary
    $tID \leftarrow T(t)$
    add $\langle tID, dID \rangle$ to B
    if B is full **then**
      sort B by tID
      write pairs $\langle tID, dID \rangle \in B$ to disk
    end if
  end for
end for
merge all the blocks (they are sorted by tID)
write T on disk (dictionary)

Block Sorted Base Indexing (BSBI), cont'd

The map T can be built:

- With a single pass (keeping a hash-table like structure)
- With two passes on the documents (collect all terms, build some structure for the map, and then make a second pass)

Single-Pass In-Memory Indexing (SPIMI)

consider documents in batches
for each batch **do**
  $L \leftarrow$ empty hashtable
  for each document in the batch **do**
    $dID \leftarrow$ docID
    for every term t in the document **do**
      if $t \notin D$ **then**
        add an empty inverted list for t to L
      end if
      append dID at the end of t
    end for
  end for
  sort the keys in L lexicographically
  write out $\langle t, \langle dID_1, \ldots, dID_k \rangle \rangle \in L$ to disk
end for
merge all output files (they are sorted by t)
constructing (simultaneously) the inverted index and the dictionary

Distributed indexing

- Large collections are usually distributed, and call for a distributed indexing
- A *distributed index* can be seen as a set of distributed indices, and can be partitioned
  - *lexicographically* (each machine contains *some of* the inverted lists)
  - *documentally* (each machine contains *a piece of each* inverted list)
- Distributed indices can be built either with an *ad hoc* distributed algorithm or using some general-purpose distributed framework.