Block Sort Based Indexing

1st StutiChaturvedi
202161006

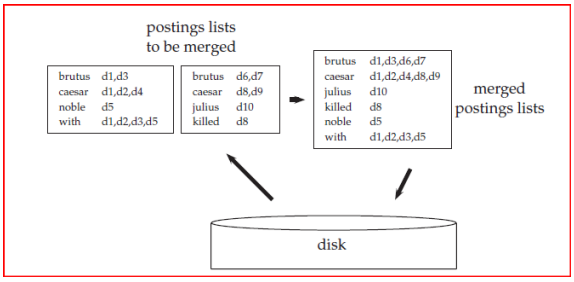
2nd HaripriyaGoswami
202161005

3rd YashitaVajpayee
202161012

***Abstract*—The main aim of this experiment is to perform block sort based indexing and make term document matrix. primarily in block sort we divide collection into blocks of same size and store them on secondary storage and perform sorting on pairs.Then we merge into single indexed file and then we will use that file to make term document matrix which comprises of frequency of terms in document**

I. Introduction

In block based indexing We partition the corpus into equal-sized blocks.The block size is chosen such that the strain on main memory is tolerable.The main goal of such an algorithm is to reduce the amount of random disk seeks when sorting—sequential disc reads are much quicker than seeks.The block indexing algorithm is one of the solution.
In BSBI ,There are following steps
i) divides the collection into equal-sized blocks,
(ii) sorts the term ID–doc ID pairings in memory,
(iii) saves intermediate sorted results to disk, and
(iv) combines all intermediate results into the final index.



Like in figure lets say we have n posting list p1,p2..pn.They are loaded from disk to main memory.
brutus:d1,d3
which means that document number 1 and 3 contains brutus term. After going through entire corpus ,merge them into single sorted file pmerged file
pseudocode:

```
BSBINDEXCONSTRUCTION()
1  n ← 0
2  while (all documents have not been processed)
3  do n ← n + 1
4     block ← PARSENEXTBLOCK()
5     BSBI-INVERT(block)
6     WRITEBLOCKTODISK(block, f_n)
7  MERGEBLOCKS(f_1, ..., f_n; f_merged)
```

II. Approach and Implementation

A. Importing modules and corpus

We used python for implementing the experiment.Modules that are required are imported. nltk, nltk.stem.porter, nltk.corpus, stopwords from nltk, defaultdict from collections, regular expression, headq, json, and gc were used. We got the list of stopwords from nltk. For this experiment, we'll use the Cran Corpus.

B. Applying BSBI algorithm

The file is subjected to the BSBI algorithm. We read lines from our chosen file, break them into words, and then delete symbols and stopwords with this method. We make a frequency dictionary of the terms, as well as a count of them. We read lines until one entire block is filled with the created data, then we build a new block and write the new data into it. Each block's data is ordered alphabetically by words. The document id is used to sort each posting list.

C. combine the blocks and store the final result

We combine blocks into a single output file with indexing created from the entire corpus when we've finished reading it.

III. Result

We received a number of temporary files with posting lists of words from the corpus. We obtain final posting lists of terms when the merging procedure is done. We have efficiently indexed a big corpus of over 15,000 entries using the BSBI algorithm (memory wise).

IV. Output

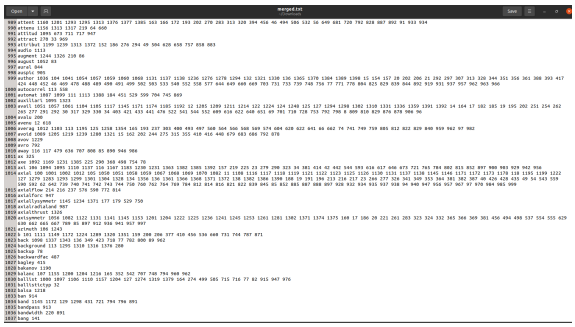


Figure shows the merged file(Final output)

V. Conclusion

For Large collection of file that usually needs secondary storage , the major requirement of external sorting is because , it reduces the number of disk seeks while sorting. But sequential disk read is faster than seeks. Hence the solution for this is BSBI . Time Complexity : O (TlogT). Further reference codes can be found at https://github.com/ir4h/LAB-CS653-2021.

References

[1] b1 Introduction to information retrieval by Christopher D Manning Prabhakar Raghavan Hinrich Schutze