Team 5 : The Crawlers
BELAID Mohamed Karim (87892)
Faizuddin Nasaruddin (87682)

Assignment 4

Since we are only 2 in the team (Semper fi) we will work on the second task: compression

Task: Compression

1. Non Compressed inv. Index:

before the presentation day our index had this structure :

token:[[articleid$_1$,[pos1,pos2,..]],

[articleid$_2$,[pos1,...]],

[...]

]

•Example:

winner:[[1744,[7,9,16]],

[274,[100]],

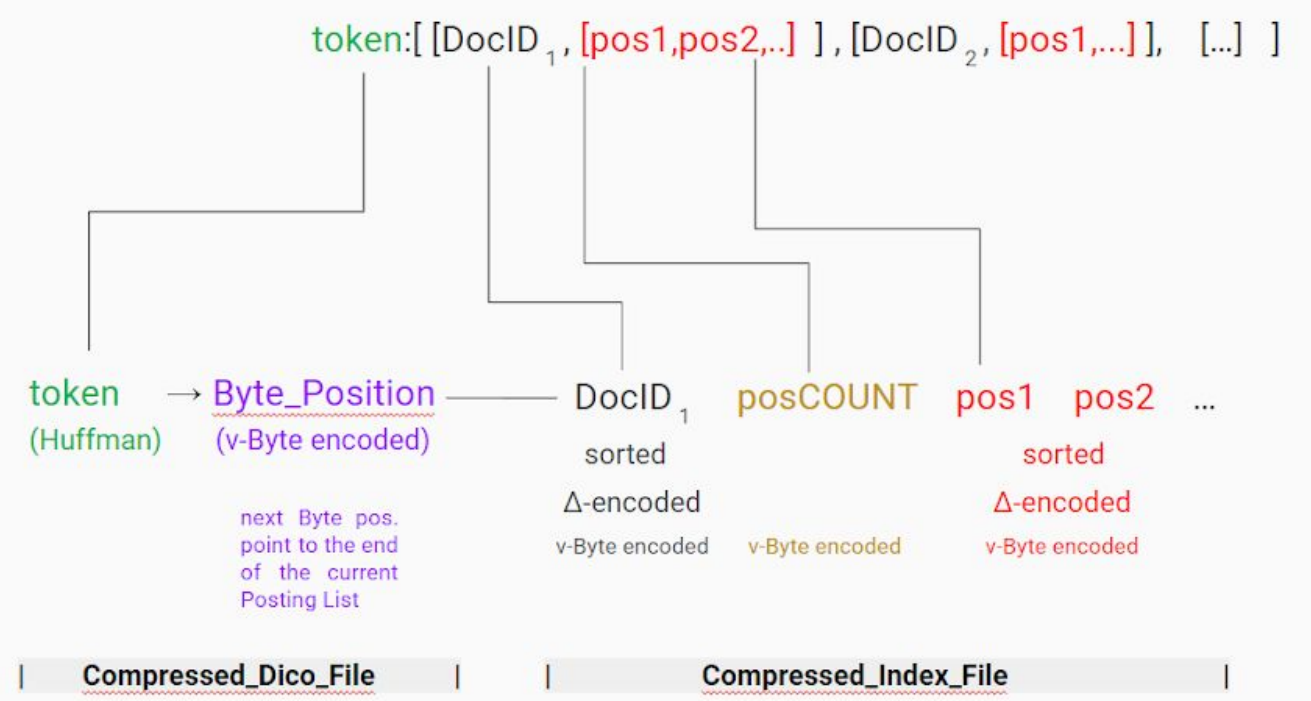
[741,[75]]

]

Team 5 : The Crawlers
BELAID Mohamed Karim (87892)
Faizuddin Nasaruddin (87682)

2. Compression Schema

We use 3 methods: Δ-encoding , v-byte and Huffman



Since the dico is using 2 different compression method (huffman for tokens and v-byte for Byte_position) we decided to store each in a separate file to have a faster decompression (parallelizable)

Team 5 : The Crawlers
BELAID Mohamed Karim (87892)
Faizuddin Nasaruddin (87682)

3. Compression Results

7-zip	WinRar	**Our Code**
27.8 %	32.7 %	**28.5 %**

Table: Comparing different compression method

7-zip and Winrar are just used as a reference and to see which percentage of compression we are able to reach.

We first compressed the index only (the dico was not compressed) we reached a percentage of 55 % after that we added Huffman encoding to compress the dico.

General Data:

- Nom compressed index's size = 13 Mo
- Unique tokens = 37'000
- articles = 2'800

Team 5 : The Crawlers
BELAID Mohamed Karim (87892)
Faizuddin Nasaruddin (87682)

4. Performance

Method	7-zip	WinRar	**Our Code**
Compression	**7.1**	**2.3**	**0.8**
Decompression	**0.5**	**1**	**0.25**

Our compression method is faster becode we compress byte by byte unlike 7-zip.

Our decompression method is faster than the others since we only decompress the dico and decompress the few Posting List we use during query time (thanks to the rand access on the compressed file)

3. Optimisations

• To speed up compression and decompression we use:
- a Trie for storing the tree of huffman
- Byte operation for v-byte, because it's faster than bit read write
- BufferedOutputStream for write operations
- StringBuilder Class

• We build index using HashMap. Then, we sort over docID to apply Δ-encoding
HashMap.insert is O(1) n times → sort is O(n log n) by it is done only one time