Assignment 3

TDT4117 Information Retrieval

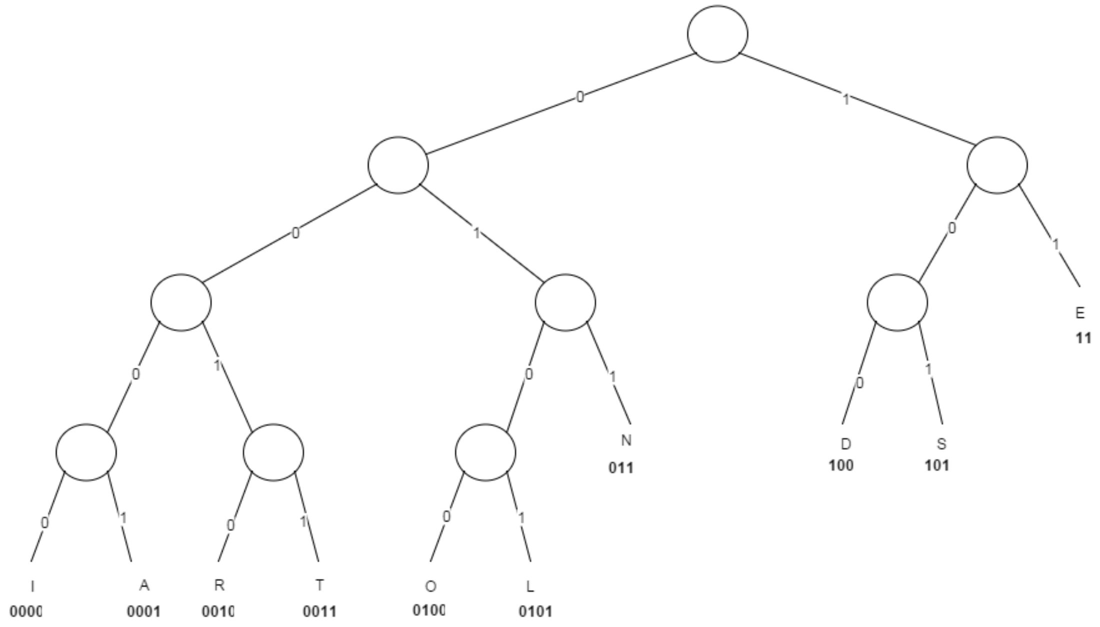
Task 1: Text Compression 1.

1. *Given the following list of most common characters in Norwegian language with frequencies, calculate the Huffman codes by using the canonical tree. Show even the canonical Huffman tree.*

$$E = 16 \quad N = 9 \quad T = 8 \quad R = 7 \quad A = 6 \quad I = 5 \quad S = 5 \quad D = 4 \quad L = 4 \quad O = 4$$

Symbol	Frequency	Canonical code	Code lenght
E	16	11	2
N	9	011	3
T	8	0011	4
R	7	0010	4
A	6	0001	4
I	5	0000	4
S	5	101	3
D	4	100	3
L	4	0101	4
O	4	0100	4

Left is 0, right is 1. *Huffman tree.*



What is the average length of the code?

If X represents all the characters, the average length will be f(x) = (frequency(x) * codelenght(x)) / (totalfrequency)

Totalfrequency = 68

Average length = (freq(E)*len(E) + freq(N)*len(N) + freq(T)*len(T) + freq(R)*len(R) + freq(A)*len(A) + freq(I)*len(I) + freq(S)*len(S) + freq(D)*len(D) + freq(L)*len(L) + freq(O)*len(O)) / (totalfrequency)

(32 + 27 + 32 + 28 + 24 + 20 + 15 + 12 + 16 + 16) / 68 = 222 / 68 = **3,265.**

What would be the average length of the code if the frequency of the letters was equal?

The average length of code when the frequency is equal will be
(len(E) + len(N) + len(T) + len(R) + len(A) + len(I) + len(S) + len(D) + len(L) + len(O)) / numberOfChars

Average length = (2 + 3 + 4 + 4 + 4 + 4 + 3 + 3 + 4 + 4)/10 = 35/10 = **3,5.**

2. *Given the following Huffman codes*

Letter	***Code***
U	*01*
O	*111*
S	*000*
T	*001*
A	*1000*
C	*1001*
L	*1010*
M	*1011*
N	*1100*
R	*1101*

decode the string 100111110101111101000:

1001 - C
111 - O
1010 - L
111 - O
1101 - R
000 - S

So the string *100111110101111101000* gives the word COLORS when decoded with the given Huffman code.

Task 2: Index Analysis Using Lucene

1. *Give a short explanation of Lucene.*

   Lucene is a free, open-source information retrieval software library. Lucene is suitable with all applications that needs text indexing and searching capability, at which it is most used. The logical architecture of Lucene is based on the idea of a document containing text fields. This allows Lucene's API to be independent of the file format, being able to index text from PDF, HTML, OpenDocuments, Microsoft Word etc., as long as the textual information can be extracted.

2. *Index the files found in the documents folder available on its learning. This is done by running the IndexFiles class. This class creates a Lucene index in the index directory from the directory passed as an argument.*

   Copy the console output after doing the indexing.

```
Indexing to directory 'index'...
adding C\Users\Magnus\documents\doc1.txt
adding C\Users\Magnus\documents\doc10.txt
adding C\Users\Magnus\documents\doc2.txt
adding C\Users\Magnus\documents\doc3.txt
adding C\Users\Magnus\documents\doc4.txt
adding C\Users\Magnus\documents\doc5.txt
adding C\Users\Magnus\documents\doc6.txt
adding C\Users\Magnus\documents\doc7.txt
adding C\Users\Magnus\documents\doc8.txt
adding C\Users\Magnus\documents\doc9.txt
477 total milliseconds
```

   Explain the steps involved in the indexing process according to the source code. Consider principally the main method and do not explain the first part regarding the input args reading.

   The main method creates from the arguments an indexpath and a doc-path. It so checks if the directory was not found and if the doc-path is not equal, and they are good, the method initializes the analyser and indexWriterConfig. The indexWriterConfig creates new indexes into the directory, and initializes indexWriter as well as running indexDocs, and after indexDocs is done the indexWriter closes.

   IndexDocs uses indexWriter to take in and check if a file can be read. If the file can be read and is a directory, it will be put in a list. As long as the file in the list isn't empty, it will run the indexDocs with the file for each file in the directory.

   If the file is not a directory FileInputStream will be initialized and runs on the file. FileInputStream creates a document and a Field, and adds the Field into the document, and adds the last value and the contents of the file. After this it check if the document needs to be updated before lastly closing FileInputStream.

3. *After indexing the documents files, when the file index has been created, a search can be performed. This is done by running the SearchFiles class. Do the following search:*

HTC

```
Enter query:
HTC
Searching for: htc
6 total matching documents
1. C\Users\Magnus\documents\doc1.txt
2. C\Users\Magnus\documents\doc9.txt
3. C\Users\Magnus\documents\doc6.txt
4. C\Users\Magnus\documents\doc7.txt
5. C\Users\Magnus\documents\doc3.txt
6. C\Users\Magnus\documents\doc10.txt
Press (q)uit or enter number to jump to a page.
```

Apple HTC

```
Enter query:
Apple HTC
Searching for: apple htc
8 total matching documents
1. C\Users\Magnus\documents\doc7.txt
2. C\Users\Magnus\documents\doc1.txt
3. C\Users\Magnus\documents\doc9.txt
4. C\Users\Magnus\documents\doc10.txt
5. C\Users\Magnus\documents\doc6.txt
6. C\Users\Magnus\documents\doc2.txt
7. C\Users\Magnus\documents\doc4.txt
8. C\Users\Magnus\documents\doc3.txt
Press (q)uit or enter number to jump to a page.
```

“Apple HTC”

```
Enter query:
"Apple HTC"
Searching for: "apple htc"
2 total matching documents
1. C\Users\Magnus\documents\doc1.txt
2. C\Users\Magnus\documents\doc9.txt
Press (q)uit or enter number to jump to a page.
```

Apple HTC university

```
Enter query:
Apple HTC university
Searching for: apple htc university
8 total matching documents
1. C\Users\Magnus\documents\doc7.txt
2. C\Users\Magnus\documents\doc1.txt
3. C\Users\Magnus\documents\doc9.txt
4. C\Users\Magnus\documents\doc10.txt
5. C\Users\Magnus\documents\doc6.txt
6. C\Users\Magnus\documents\doc2.txt
```

```
7. C\Users\Magnus\documents\doc4.txt
8. C\Users\Magnus\documents\doc3.txt
Press (q)uit or enter number to jump to a page.
```

It collects the documents where both words is represented and therefore the AND-operator is used at first. Then all the documents that have the first quarry listed are retrieved and finally all the documents containing the last of the quarry is listed. For this, the OR-operator is listed.

4.
```
Indexing to directory 'index'...
:
:
adding C:\enron\maildir\zufferli-j\sent_items\97
adding C:\enron\maildir\zufferli-j\sent_items\98
adding C:\enron\maildir\zufferli-j\sent_items\99
1626753 total milliseconds
```

NTNU

```
Enter query:
NTNU
Searching for: ntnu
1 total matching documents
1. C:\enron\maildir\lay-k\inbox\1126
Press (q)uit or enter number to jump to a page.
```

Meeting

```
Enter query:
meeting
Searching for: meeting
57635 total matching documents
1. C:\enron\maildir\dasovich-j\notes_inbox\3389
2. C:\enron\maildir\dasovich-j\all_documents\12436
3. C:\enron\maildir\haedicke-m\sent_items\9
4. C:\enron\maildir\lay-k\all_documents\1107
5. C:\enron\maildir\lay-k\sent\201
6. C:\enron\maildir\lay-k\_sent\241
7. C:\enron\maildir\campbell-l\all_documents\1083
8. C:\enron\maildir\campbell-l\discussion_threads\958
9. C:\enron\maildir\campbell-l\notes_inbox\216
10. C:\enron\maildir\blair-l\sent_items\626
Press (n)ext page, (q)uit or enter number to jump to a page.
```

Hi

```
Enter query:
hi
Searching for: hi
23884 total matching documents
1. C:\enron\maildir\schoolcraft-d\deleted_items\128
2. C:\enron\maildir\schoolcraft-d\deleted_items\108
3. C:\enron\maildir\jones-t\sent\6053
4. C:\enron\maildir\jones-t\all_documents\10833
5. C:\enron\maildir\griffith-j\all_documents\1
6. C:\enron\maildir\griffith-j\discussion_threads\1
7. C:\enron\maildir\griffith-j\private_folders\personal\8
8. C:\enron\maildir\kean-s\sent\1350
9. C:\enron\maildir\kean-s\all_documents\5002
10. C:\enron\maildir\kean-s\archiving\untitled\5001
Press (n)ext page, (q)uit or enter number to jump to a page.
```

The time needed when searching the documents where next to nothing and we got the results right away.