Assignment 4 - TDT4117

Hermann Owren Elton, Olaf Rosendahl

September 21, 2022

Task 1: Text Indexing

Text: Intelligent behavior in people is a product of the mind. But the mind itself is more like what the human brain does.
We've used the english stopwords which were linked to in assignment 3 where stopwords have been necessary

a) The text without stop words and punctuation: **Intelligent behavior people product mind mind itself more human brain**

Inverted file/list:

Vocabulary	Occurrences
behavior	13
brain	106
human	100
intelligent	1
itself	71
mind	52, 66
more	81
people	25
product	37

b) **Blocks**: | Intelligent behavior in people | is a product of the mind. | But the mind itself is more | like what the human brain does. |

Vocabulary	Occurrences
behavior	1
brain	4
human	4
Intelligent	1
itself	3
mind	2, 3
more	3
people	1
product	2

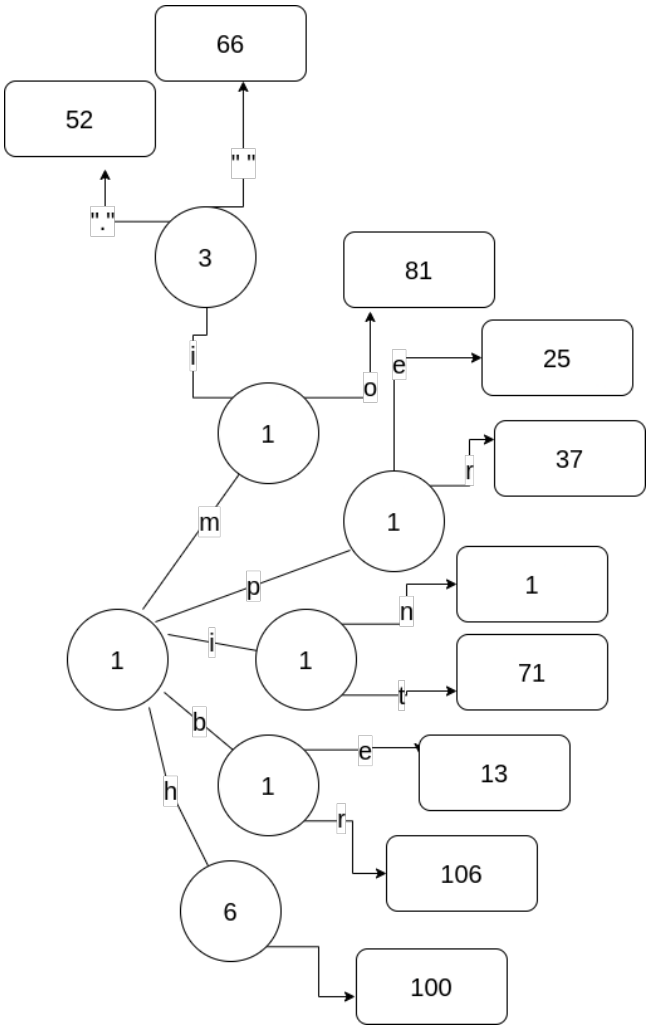

Figure 1: A partial vocabulary suffix tree

c)

d) Documents without stopwords and punctuation:

D1 Although know much more human brain even

D2 ten years ago thinking engages remains pretty much total

D3 mystery big jigsaw puzzle see many

D4 pieces cannot put together much about

D5 understand

Word	Documents	Word	Documents
about	4	mystery	3
ago	2	pieces	4
Although	1	pretty	2
big	3	put	4
brain	1	puzzle	3
cannot	4	remains	2
engages	2	see	3
even	1	ten	2
human	1	thinking	2
jigsaw	3	together	4
know	1	total	2
many	3	understand	5
more	1	years	2
much	1, 2, 4		

Task 2 : Index Analysis Using Lucene

a) The ELK stack in an acronym for 3 projects called Elasticsearch, Logstash, and Kibana. Where Elasticsearch is a document indexer that provides a search engine and an analytics tool, Logstash is a pipeline that can retrive data from multiple places and transform it so it can be put in a database/collection, Kibana is a visualization tool that can visualize the data in Elasticsearch. Lucene is a information retrivel library written in Java. Its a full-text search engine. Elasticsearch is built on top of Lucene as a server that utilizes Lucene power to preform text-search. Elasticsearch is a distributed system for Lucene. The Lucene query langauge is written sort of like text, you state what field you want to search on in the collection or the whole collection then you write what you want to find on that field or collection. An example of this can be you have a collection with the field status and you want to find all the documents with a status off 400 or lowwer. this can be done with the query status: [* to 400] or if you want to find all statuses with 200 you can write status:200. KQL works in the same way but has a bit different syntax, on somethings.

b) We believe a simple solution would be to create a set of the words in all documents with stopwords removed. Each key can contain an array of which documents it exists in. (`Set[List[int]]`)

c)

d) The resulting documents in our implementation and ELK does not always match. as can be seen in Figure 2 and 3.

```
Query: claim, hits: {'Text2.txt'}
Query: claim*, hits: set()
Query: claims of duty, hits: {'Text6.txt'}
Query: claims of duty in an alternative way, hits: {'Text4.txt', 'Text6.txt'}
```

Figure 2: Query Results - Our implementation

```
Query: "claim", hits: ['Text2.txt']
Query: "claim*", hits: ['Text2.txt']
Query: "claims of duty", hits: ['Text6.txt', 'Text1.txt', 'Text4.txt', 'Text5.txt', 'T
ext2.txt', 'Text3.txt']
Query: "claims of duty in an alternative way", hits: ['Text6.txt', 'Text4.txt', 'Text5
.txt', 'Text1.txt', 'Text2.txt', 'Text3.txt']
```

Figure 3: Query Results - ELK stack

e)

- In our implementation, we split the query into an array of words: `["claims", "of", "duty"]`. We then check if each word are in the index, and if so add the documents which are stored in the index to the result. The result is a set to avoid including the same document multiple times if a multiple words are present in the same document.

  In the ELK stack, Elasticsearch uses its inverted index to find documents that contains the words in the given query. We assume that the index have been stemmed on creation since "claims" isn't present in `Text2.txt` ("claim" is), but still part of the result.
- For example only "duty" results in the same result both from our implementation and the ELK stack.
- The result is what we expected since our search returns a document if at least one of the words are in the given document. Elasticsearch is more sophisticated with more advanced techniques for processing words both in the index and the query and will therefore retrieve more results in most cases even if the query are the same.

f) Both queries, *"Norwegian and University and Science and Technology"* and *"Norwegian University Science Technology"*, returned the exact same result when searching with a *match*-query. The following are the results:

- `enron/maildir/kean-s/deleted\_items/495.`
- `enron/maildir/presto-k/sent\_items/689.`
- `enron/maildir/derrick-j/deleted\_items/74.`
- `enron/maildir/kean-s/deleted\_items/300.`
- `enron/maildir/derrick-j/sent\_items/273.`
- `enron/maildir/lokay-m/discussion\_threads/35.`
- `enron/maildir/lokay-m/enron\_t\_s/4.`
- `enron/maildir/lokay-m/all\_documents/38.`
- `enron/maildir/kean-s/delete\_items/37.`
- `enron/maildir/giron-d/inbox/188.`