Exercise Set 2

Enrico Buratto

28/01/2022

Question 1

Elias codes

- **100**: the binary representation for 100 is 1100100; the offset is then 100100 (the binary number without the leading bit). The selector size is 6; in unary code, this is 0000001. The Elias-γ code for 100 is then **0000001100100**.
- **200**: The offset is 1001000, the selector is 7 so 00000001. The Elias-γ code for 200 is then **000000011001000**.
- **400**: The offset is 10010000, the selector is 8 so 000000001. The Elias-γ code for 200 is then **00000000110010000**.

VByte codes

- **100**: the binary representation for 100 is 1100100. This fits 7 bit, so we can put the continuation bit $c = 0$. The VByte code for 100 is then **01100100**.
- **200**: the binary representation for 200 is 11001000. This does not fit 7 bit, so we put the lower 7 bits in the first byte, we put the continuation bit $c = 1$ and we encode the remaining bit in the second byte (with $c = 0$). The VByte code for 200 is then **11001000 00000001**.
- **400**: the binary representation for 400 is 110010000. This does not fit 7 bit, so we put the lower 7 bits in the first byte, we put the continuation bit $c = 1$ and we encode the remaining two bits in the second byte (with $c = 0$). The VByte code for 400 is then **10010000 00000011**.

Question 2

a. In a Boolean retrieval system, stemming never lowers precision - **False**: the retrieved results are more and more general or, at least, the same. The precision is then at most equal, if not lower.
b. In a Boolean retrieval system, stemming never lowers recall - **True**: for the same reason, the recall is at least equal, if not higher.
c. Stemming increases the size of the vocabulary - **False**: more original words can become the same but not vice versa, so the size of the vocabulary is at most equal if not lower.
d. Stemming should be invoked at indexing time, but not while processing a query - **False**: it should be applied also while processing a query in order to match the searched terms with the indexed ones.

Question 3

I would say that **marketing/market** and **university/universe** should not be the same. While abandon/abandonment, absorbency/absorbent and volume/volumes all belong to the same field of interest,

`marketing` has a totally different meaning than `markets` (and both are stemmed to `market`); the same applies for `university` and `universe` (that are stemmed to `univers`).

Question 4

The URL-seen test is a test that the web crawler should perform in order to check if it already seen a specific document. It is performed checking if the URLs extracted by the parser are already present in the seen URLs set. An efficient data structure for this task a **sieve**; the main idea is to have a "queue with memory" which provides functions to enqueue and dequeue (like every other queue structure), but it adds the guarantee that if an element is enqueued multiple times it will only be dequeued once.

Question 5

The reason why it is better to partition hosts rather than individual URLs between the nodes of a distributed crawler is that same hosts most probably (and usually) are located on the same local machine and/or on the same network. URLs, on the contrary, are not assured to be on the same network. Obviously, being on the same network reduces the time that the crawler needs to complete its task, *i.e.* index the websites following the hyperlinks.

Question 6

Normalization

Before drawing the inverted index, I applied these normalization techniques (in order):

- Case folding
- Punctuation removal
- Stemming with Porter algorithm

Case folding and punctuation removal

Page 1: the smallest planet is mercury

Page 2: jupyter the largest planet

Page 3: neptune is smaller than uranus but is heavier

Page 4: uranus is smaller than saturn

Page 5: neptune is heavier than uranus

Page 6: jupyter saturn uranus neptune

Stemming

Page 1: the smallest planet is mercuri

Page 2: jupyt the largest planet

Page 3: neptun is smaller than uranu but is heavier

Page 4: uranu is smaller than saturn

Page 5: neptun is heavier than uranu

Page 6: jupyt saturn uranu neptun

Inverted index

The inverted indices I drew have the following notation:

```
lexicon[doc frequency] -> [document ids] , [position counts] , [term positions]
```

```
but[1] -> [3] , [1] , [6]
heavier[2] -> [3][5] , [1][1] , [8][3]
is[4] -> [1][3][4][5] , [1][2][1][1] , [4][2][7][2][2]
jupyt[2] -> [2][6] , [1][1] , [1][1]
largest[2] -> [2] , [1] , [3]
mercuri[1] -> [1] , [1] , [5]
neptun[3] -> [3][5][6] , [1][1][1] , [1][1][4]
planet[2] -> [1][2] , [1][1] , [3][4]
saturn[2] -> [4][6] , [1][1] , [5][2]
smaller[2] -> [3][4] , [1][1] , [3][3]
smallest[1] -> [1] , [1] , [2]
than[3] -> [3][4][5] , [1][1][1] , [4][4][4]
the[2] -> [1][2] , [1][1] , [1][2]
uranu[4] -> [3][4][5][6] , [1][1][1][1] , [5][1][5][3]
```

Question 7

In Elias-Fano encoding we have an array of n non-negative integers in increasing order and an upperbound on them; we can call x_i the numbers and z the upperbound. So we have:

$$0 \leqslant x_0 \leqslant x_1 \leqslant \cdots \leqslant x_{n-1} \leqslant z$$

. We then represent the sequence of number in two bits arrays, which are composed as follows:

- The lower $max\{0, floor(log_2(z/n))\}$ bits of each integer are stored explicitly and contiguously in the lower-bits array
- The upper bits are stored in the upper-bits array as a sequence of unary-coded gaps

For the example set {4,8,10,12,33} we assume an upperbound equal to 36, so $l = 2$ bits. We have that:

- 4 = 100
- 8 = 1000
- 10 = 1010
- 12 = 1111
- 33 = 100001

We then have the lower-bits array **L=0000101101** (the last $l = 2$ digits of each number). We then have that:

- 1 = 1
- 10 = 2
- 10 = 2
- 11 = 3
- 1000 = 8

We then have the upper-bits array **U=0101101000001** (the differences between the remaining bits in unary code).