TÜ Information Retrieval

Übung 2

Heike Adel, Sascha Rothe

Center for Information and Language Processing, University of Munich

May 8, 2014

Problem 1

Assume that

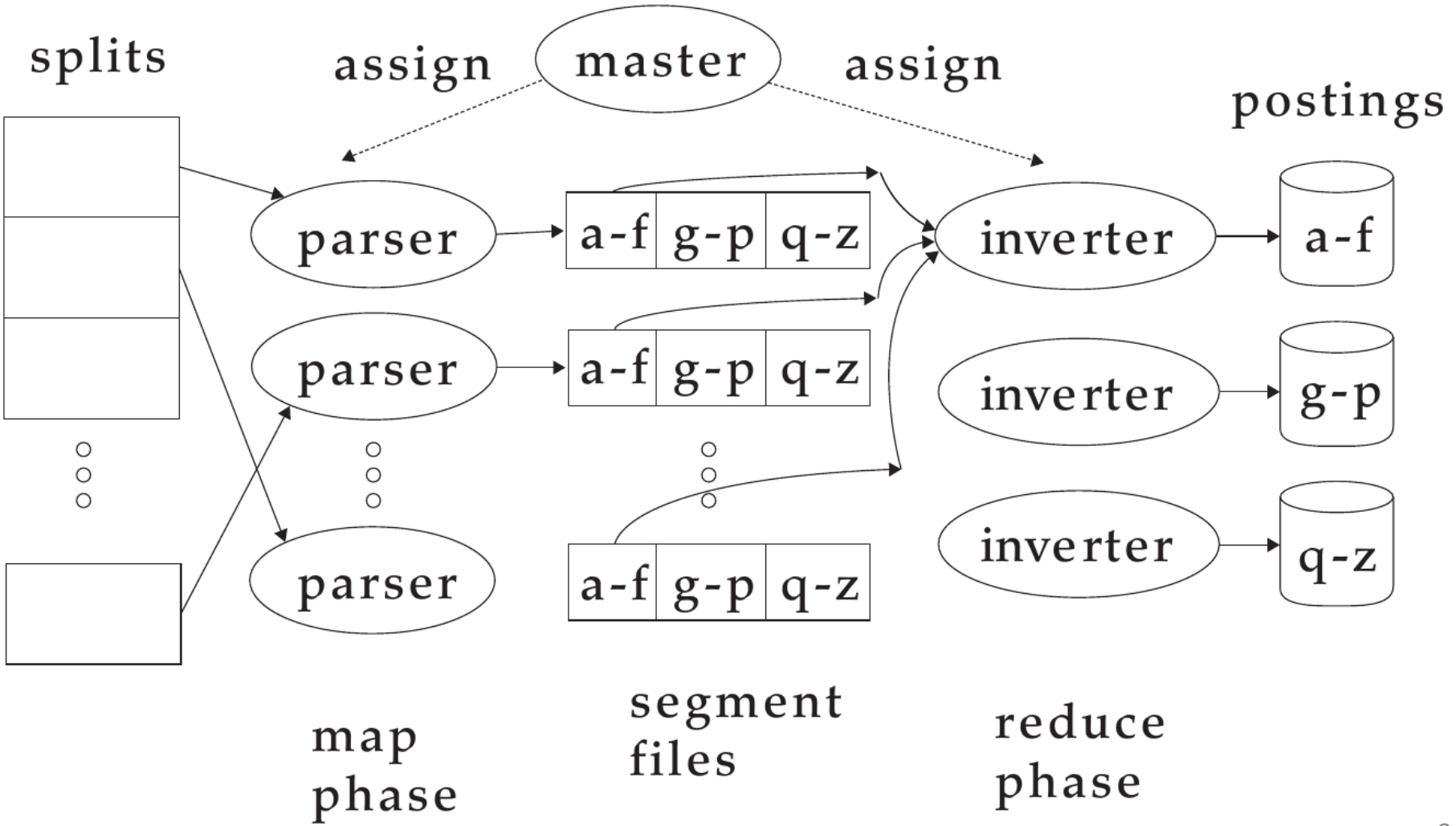
- machines in MapReduce have 100GB of disk space each
- the postings list of the term THE has a size of 180GB for a particular collection
- we do not use compression

Then the MapReduce algorithm as described in class cannot be run to construct the inverted index. Why?
How would you modify the algorithm so that it can handle this case?

Problem 1

Recap

MapReduce as described in class:



Problem 1

Assume that

- machines in MapReduce have 100GB of disk space each
- the postings list of the term THE has a size of 180GB for a particular collection
- we do not use compression

Then the MapReduce algorithm as described in class cannot be run to construct the inverted index. Why?

Problem 1

Assume that

- machines in MapReduce have 100GB of disk space each
- the postings list of the term THE has a size of 180GB for a particular collection
- we do not use compression

Then the MapReduce algorithm as described in class cannot be run to construct the inverted index. Why?

$\Rightarrow$ The algorithm assumes that each term's postings list will fit on a single inverter. This is not true for the postings list of THE.

Problem 1

Assume that

- machines in MapReduce have 100GB of disk space each
- the postings list of the term THE has a size of 180GB for a particular collection
- we do not use compression

How would you modify the algorithm so that it can handle this case?

Problem 1

Assume that

- machines in MapReduce have 100GB of disk space each
- the postings list of the term THE has a size of 180GB for a particular collection
- we do not use compression

How would you modify the algorithm so that it can handle this case?

$\Rightarrow$ Let N be the largest *docID* for THE.
The master defines two partitions for the postings list of THE:
(1) THE in documents with $docID \leq \frac{N}{2}$
(2) THE in documents with $docID > \frac{N}{2}$
These shorter postings lists can be sorted by two single inverters.
Afterwards, the master links the end of list (1) to the beginning of list (2).

Problem 2

Given a collection with exactly 4 words a, b, c, d.
The frequency order is $a > b > c > d$.
The total number of tokens in the collection is 5000.

Assume that Zipf's law holds exactly for this collection.
What are the frequencies of the four words?

Problem 2

Recap

Zipf's law:

The i^{th} most frequent term has a collection frequency cf_i proportional to $\frac{1}{i}$. Hence: $\exists c : cf_i = c \cdot \frac{1}{i}$

More general: In natural language, there are a few very frequent terms and very many very rare terms.

Problem 2

Given a collection with exactly 4 words a, b, c, d.
The frequency order is $a > b > c > d$.
The total number of tokens in the collection is 5000.

Assume that Zipf's law holds exactly for this collection.
What are the frequencies of the four words?

Problem 2

Given a collection with exactly 4 words a, b, c, d.
The frequency order is $a > b > c > d$.
The total number of tokens in the collection is 5000.

Assume that Zipf's law holds exactly for this collection.
What are the frequencies of the four words?

$\Rightarrow$Assume a appears f_a times in the collection. Then:
$f_a + \frac{1}{2} \cdot f_a + \frac{1}{3} \cdot f_a + \frac{1}{4} \cdot f_a = 5000$
$f_a = 2400$
$f_b = \frac{1}{2} \cdot f_a = 1200$
$f_c = \frac{1}{3} \cdot f_a = 800$
$f_d = \frac{1}{4} \cdot f_a = 600$

Problem 3

We define a hapax legomenon as a term that occurs exactly once in a collection. We want to estimate the number of hapax legomena using Heap's law and Zipf's law.

(i) How many unique terms does a web collection of 600,000,000 web pages containing 600 tokens on average have?
Use the Heaps parameters $k = 100$ and $b = 0.5$.

Problem 3

We define a hapax legomenon as a term that occurs exactly once in a collection. We want to estimate the number of hapax legomena using Heap's law and Zipf's law.

(i) How many unique terms does a web collection of 600,000,000 web pages containing 600 tokens on average have?
Use the Heaps parameters $k = 100$ and $b = 0.5$.

Recap

Heap's law:

$M = k \cdot T^b$
with M being the size of the vocabulary and T the number of tokens in the collection

Problem 3

We define a hapax legomenon as a term that occurs exactly once in a collection. We want to estimate the number of hapax legomena using Heap's law and Zipf's law.

(i) How many unique terms does a web collection of 600,000,000 web pages containing 600 tokens on average have?
Use the Heaps parameters $k = 100$ and $b = 0.5$.

Problem 3

We define a hapax legomenon as a term that occurs exactly once in a collection. We want to estimate the number of hapax legomena using Heap's law and Zipf's law.

(i) How many unique terms does a web collection of 600,000,000 web pages containing 600 tokens on average have?
Use the Heaps parameters $k = 100$ and $b = 0.5$.

$\Rightarrow M = 100 \cdot (600,000,000 \cdot 600)^{0.5} = 60,000,000$

Problem 3

We define a hapax legomenon as a term that occurs exactly once in a collection. We want to estimate the number of hapax legomena using Heap's law and Zipf's law.

(ii) Use Zipf's law to estimate the proportion of the term vocabulary of the collection that consists of hapax legomena.
Hint: $\sum_{i=1}^{n} \frac{1}{i} \approx ln(n)$

Problem 3

We define a hapax legomenon as a term that occurs exactly once in a collection. We want to estimate the number of hapax legomena using Heap's law and Zipf's law.

(ii) Use Zipf's law to estimate the proportion of the term vocabulary of the collection that consists of hapax legomena.
Hint: $\sum_{i=1}^{n} \frac{1}{i} \approx ln(n)$

$\Rightarrow$ Zipf's law: $cf_i \propto 1/i \Rightarrow \exists c : cf_i = c \cdot \frac{1}{i}$
Calculate c:
The sum of all collection frequencies is the total number of tokens T:
$(600,000,000 \cdot 600) = T = \sum_{i=1}^{M} c \cdot \frac{1}{i} = c \cdot \sum_{i=1}^{60,000,000} \frac{1}{i}$
$\approx c \cdot ln(60,000,000) \approx 17.9c$
$\Rightarrow c = \frac{T}{17.9} \approx 2 \cdot 10^{10}$

Problem 3

We define a hapax legomenon as a term that occurs exactly once in a collection. We want to estimate the number of hapax legomena using Heap's law and Zipf's law.

(ii) Use Zipf's law to estimate the proportion of the term vocabulary of the collection that consists of hapax legomena.
Hint: $\sum_{i=1}^{n} \frac{1}{i} \approx ln(n)$

Problem 3

We define a hapax legomenon as a term that occurs exactly once in a collection. We want to estimate the number of hapax legomena using Heap's law and Zipf's law.

(ii) Use Zipf's law to estimate the proportion of the term vocabulary of the collection that consists of hapax legomena.
Hint: $\sum_{i=1}^{n} \frac{1}{i} \approx ln(n)$

$\Rightarrow$ Zipf's law: $cf_i \propto 1/i \Rightarrow cf_i = 2 \cdot 10^{10} \cdot \frac{1}{i}$

Calculate the frequency of the least frequent term (i.e. term with rank $i = 60,000,000$):
$cf_{60,000,000} = \frac{2 \cdot 10^{10}}{60,000,000} \approx \frac{1}{3} \cdot 1000$
$\Rightarrow$ The least frequent term appears more than once! $\Rightarrow$ Based on Heap's law and Zipf's law, there are no hapax legomena in the collection!
$\Rightarrow$ The proportion of hapax legomena is 0.

Problem 3

Based on Heap's law and Zipf's law, there are no hapax legomena in the collection!

(iii) Do you think that the estimate you get is correct?

Problem 3

Based on Heap's law and Zipf's law, there are no hapax legomena in the collection!

(iii) Do you think that the estimate you get is correct?

$\Rightarrow$ This prediction is not correct. Generally, roughly 50% of the vocabulary consists of hapax legomena (but this depends on the collection!)

Problem 3

Based on Heap's law and Zipf's law, there are no hapax legomena in the collection!

(iv) Discuss what possible reasons there might be for the incorrectness of the estimate

Problem 3

Based on Heap's law and Zipf's law, there are no hapax legomena in the collection!

(iv) Discuss what possible reasons there might be for the incorrectness of the estimate

$\Rightarrow$ One of the laws has to be the reason for the incorrect prediction

- Heap's law: fairly accurate (see: class) $\Rightarrow$ Heap's law is not the reason
- Zipf's law: bad fit, especially at the low-frequent end $\Rightarrow$ This is the reason for the incorrect prediction!

Problem 4

γ-codes are inefficient for large numbers (e.g. > 1000) because they encode the length of the offset in binary code. δ-codes, on the other hand, use γ-codes for encoding this length.

Definitions

- γ-code of G: unary-code(length(offset(G))), offset(G)
- δ-code of G: γ-code(length(offset($G+1$))), offset($G+1$)

Compute the δ-codes for $1, 2, 3, 4, 31, 63, 127, 1023$

Problem 4

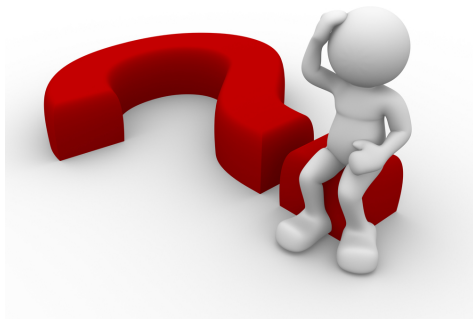
Compute the δ-codes for $1, 2, 3, 4, 31, 63, 127, 1023$

$\Rightarrow$

number	δ-code
1	0,0
2	0,1
3	10,0,00
4	10,0,01
31	110,01,00000
63	110,10,000000
127	110,11,0000000
1023	1110,010,0000000000

The end

Thank you for your attention!



Do you have any questions?