INFO221v12
IR III

Truls Pedersen
Institutt for informasjons- og medievitenskap
Universitetet i Bergen

Oversikt

- (my)SQL
- Indeksering (tekst)
  - Zipf og Heap lover
  - Term-frekvens og -plassering
  - Relativ frekvens ($tf - idf$)
- Indeksering (bilde)
- Indeksering:
  - BSBI
  - distribuert

SQL - eksempel

Vi har en tabell `'multimedia'` i en database. Hvert dokument i tabellen har (bl.a.)

- unik heltall `'ID'`, og
- en tekst `'title'`.

Vi vil at det skal gå raskt å slå opp på tittel med *exact match* søk.

SQL

```
mysql> show index from multimedia;
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| Table      | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| multimedia |          0 | PRIMARY  |            1 | ID          | A         |          15 |     NULL | NULL   |      | BTREE      |         |
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
```

SQL

```
mysql> show index from multimedia;
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| Table      | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| multimedia |          0 | PRIMARY  |            1 | ID          | A         |          15 |     NULL | NULL   |      | BTREE      |         |
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
```

Vi kan legge til en ny indeks:

```
mysql> show index from multimedia;
+------------+------------+------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| Table      | Non_unique | Key_name   | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+------------+------------+------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| multimedia |          0 | PRIMARY    |            1 | ID          | A         |           3 |     NULL | NULL   |      | BTREE      |         |
| multimedia |          1 | title-index |            1 | Title       | A         |           3 |     NULL | NULL   |      | BTREE      |         |
+------------+------------+------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
```

(`create index 'title-index' on multimedia('title');`)

SQL

```
mysql> show index from multimedia;
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| Table      | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| multimedia |          0 | PRIMARY  |            1 | ID          | A         |          15 |     NULL | NULL   |      | BTREE      |         |
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
```

Vi kan legge til en ny indeks:

```
mysql> show index from multimedia;
+------------+------------+------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| Table      | Non_unique | Key_name   | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+------------+------------+------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| multimedia |          0 | PRIMARY    |            1 | ID          | A         |           3 |     NULL | NULL   |      | BTREE      |         |
| multimedia |          1 | title-index|            1 | Title       | A         |           3 |     NULL | NULL   |      | BTREE      |         |
+------------+------------+------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
```

(`create index 'title-index' on multimedia('title');`)

Vi kan nå gjøre et “eksakt match”-søk i `Title`-kolonnen *mye* raskere.

Neste gang vi legger til en rekke i `multimedia`-tabellen, må mySQL også oppdatere denne indeksen.

2-gram og jokertegn - eksempel

Vi har en termindeks `'terms'` i en database. Tabellen har (bl.a.)

- et heltall `'docid'`, og
- en tekst `'term'`.

Vi har også en tabell `'twograms'` med

- en tekst `'term'`, og
- en tekst `'twogram'`.

Hvordan søker vi etter alle dokumenter som inneholder et ord på formen

`'p*ha*'`?

2-gram og jokertegn

Søker etter: `p*ha*`:

2-gram og jokertegn

Søker etter: `p*ha*`:

Alle termer som inneholder 2-grammene “$p”

2-gram og jokertegn

Søker etter: `p*ha*`:

Alle termer som inneholder 2-grammene “$p”

```
(SELECT term FROM twograms WHERE
 twogram = '$p');
```

2-gram og jokertegn

Søker etter: `p*ha*`:

Alle termer som inneholder 2-grammene “$p” *og* “ha”

```
SELECT term FROM twograms WHERE
  twogram = 'ha'
  AND
  term IN
       (SELECT term FROM twograms WHERE
        twogram = '$p');
```

2-gram og jokertegn

```
mysql> select term, docid from terms where term in (select
term from twograms where twogram = 'ha' AND term in (select
 term from twograms where twogram = '$p'));
+-----------+-------+
| term      | docid |
+-----------+-------+
| pasha     |    10 |
| phantom   |     6 |
| pharaoh   |    11 |
| phases    |    11 |
| purchased |     2 |
| purchased |     5 |
+-----------+-------+
6 rows in set (0.00 sec)
```

(Mine lister: 4260 termer, 25201 bigrammer.)

Tekst og struktur

Vi er etterhvert ganske godt kjent med tekststruktur.

Tekst regnes som *semi-strukturert* data.

Term	Ord, forkortelser, tall, ...
Skilletegn	Mellomrom, (, ., ,, ?,), ...
Fraser	vær så god, ...
Setninger	Dette er et eksempel på en setning.
Avsnitt	Sammensetning av setninger.
Kapittel	Sammensetning av avsnitt.

Nyttige termer

Når vi søker etter et tekstdokument antar vi at ordene som forekommer i dokumentet kan brukes som en indikator for dokumentets (semantiske) innhold.

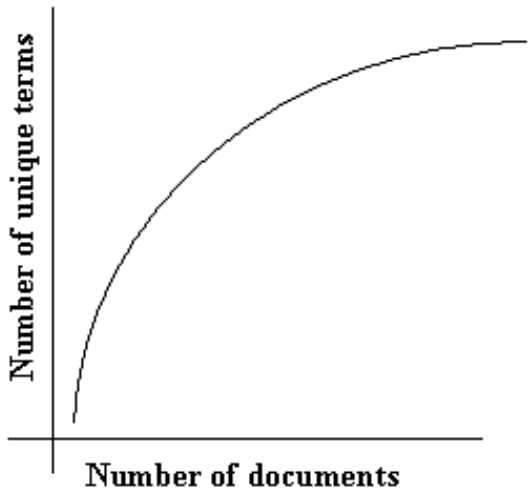
Dette kan gi et godt grunnlag for søk, men hvor mange termer skal vi ta med i indeksen vår?

Hvor mange termer trenger vi?

Når vi legger inn et dokument i samlingen vår oppdaterer vi termindeksen vår også. Hvis vi allerede har en “stor” samling, får vi sansynligvis “få” nye termer.

Hvor mange termer trenger vi?

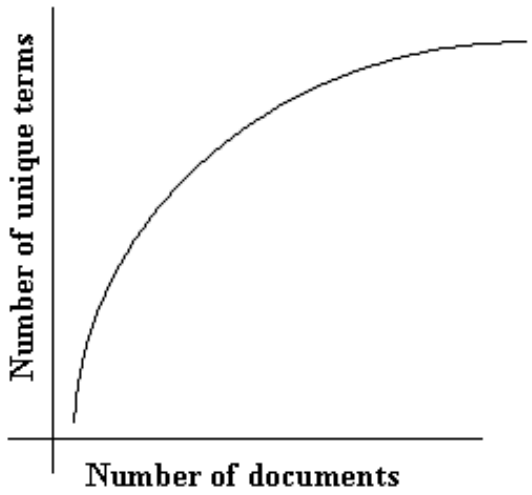
Når vi legger inn et dokument i samlingen vår oppdaterer vi termindeksen vår også. Hvis vi allerede har en “stor” samling, får vi sansynligvis “få” nye termer.



Heaps lov

Hvor mange termer trenger vi?

Når vi legger inn et dokument i samlingen vår oppdaterer vi termindeksen vår også. Hvis vi allerede har en “stor” samling, får vi sansynligvis “få” nye termer.

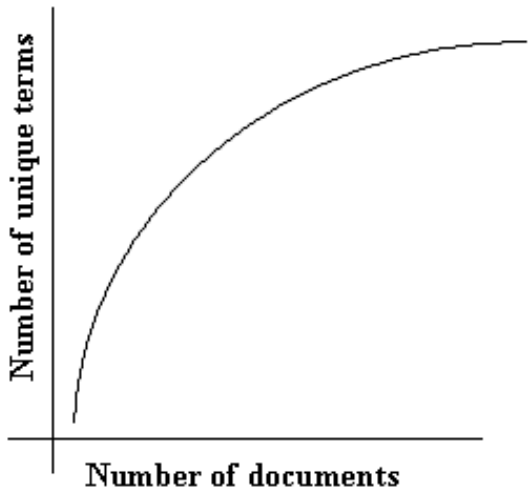


Heaps lov

Dette er bra! Vi sparer plass!

Hvor mange termer trenger vi?

Når vi legger inn et dokument i samlingen vår oppdaterer vi termindeksen vår også. Hvis vi allerede har en “stor” samling, får vi sansynligvis “få” nye termer.



Heaps lov

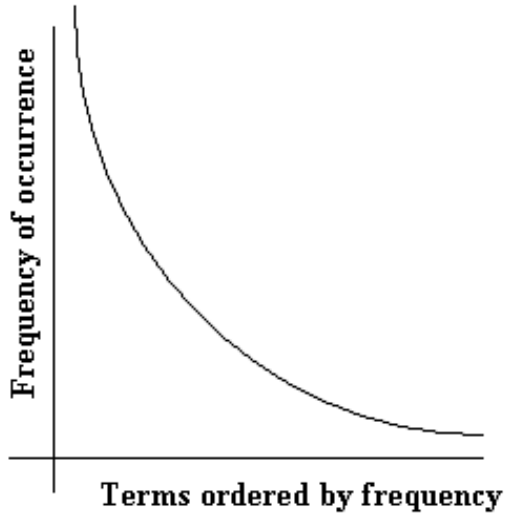
Dette er bra! Vi sparer plass! ... men de fleste ordene i teksten forekommer også i andre, og egner seg dårlig til å peke ut det nye dokumentet.

Hvor egnet er et ord til å peke ut et dokument?

De aller vanligste ordene forekommer *svært* ofte. I en stor samling, vil frekvente ord peke ut *mange* dokumenter.

Hvor egnet er et ord til å peke ut et dokument?

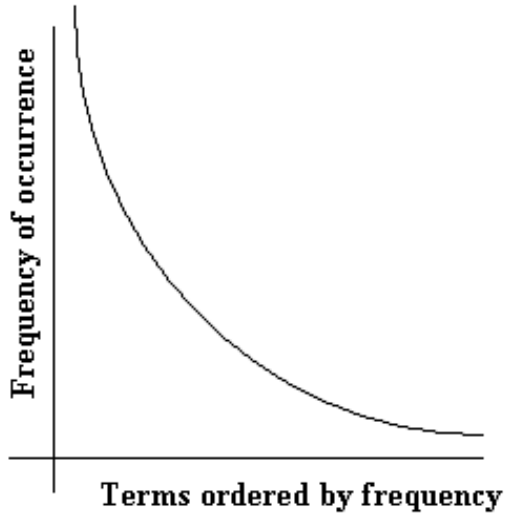
De aller vanligste ordene forekommer *svært* ofte. I en stor samling, vil frekvente ord peke ut *mange* dokumenter.



Zipfs lov

Hvor egnet er et ord til å peke ut et dokument?

De aller vanligste ordene forekommer *svært* ofte. I en stor samling, vil frekvente ord peke ut *mange* dokumenter.



Zipfs lov

Vi gjør databasen vår mindre ved å ignorere de mest frekvente ordene. Dessuten kan det gi bedre nøyaktighet.

Hvordan bruker vi en term som grunnlag for et søk?

Hvordan bruker vi en term som grunnlag for et søk?

Forslag: *“Sjekker om søketermen forekommer i dokumentet.”*

Men vil ikke f.eks. “*database*” forekomme spesielt ofte i tekster om databaser?

Hvordan bruker vi en term som grunnlag for et søk?

Forslag: *“Sjekker om søketermen forekommer i dokumentet.”*

Men vil ikke f.eks. “*database*” forekomme spesielt ofte i tekster om databaser?

Forslag: *“Sjekker hvor ofte søketermen forekommer i dokumentet.”*

Hvordan bruker vi en term som grunnlag for et søk?

Forslag: *"Sjekker om søketermen forekommer i dokumentet."*

Men vil ikke f.eks. "*database*" forekomme spesielt ofte i tekster om databaser?

Forslag: *"Sjekker hvor ofte søketermen forekommer i dokumentet."*

Hva om en bok om administrasjon inneholder noen paragrafer hvor "*database*" forekommer ofte?

Forslag: *"Sjekker hvor og hvor ofte søketermen forekommer i dokumentet."*

Term-posisjon

For hver søketerm som forekommer i et dokument registrerer vi

1. Hvilket dokument det forekommer i (som før),
2. hvilket kapittel/avsnitt,
3. hvilken setning, og
4. hvilket ord søketermen forekommer som.

Term-posisjon

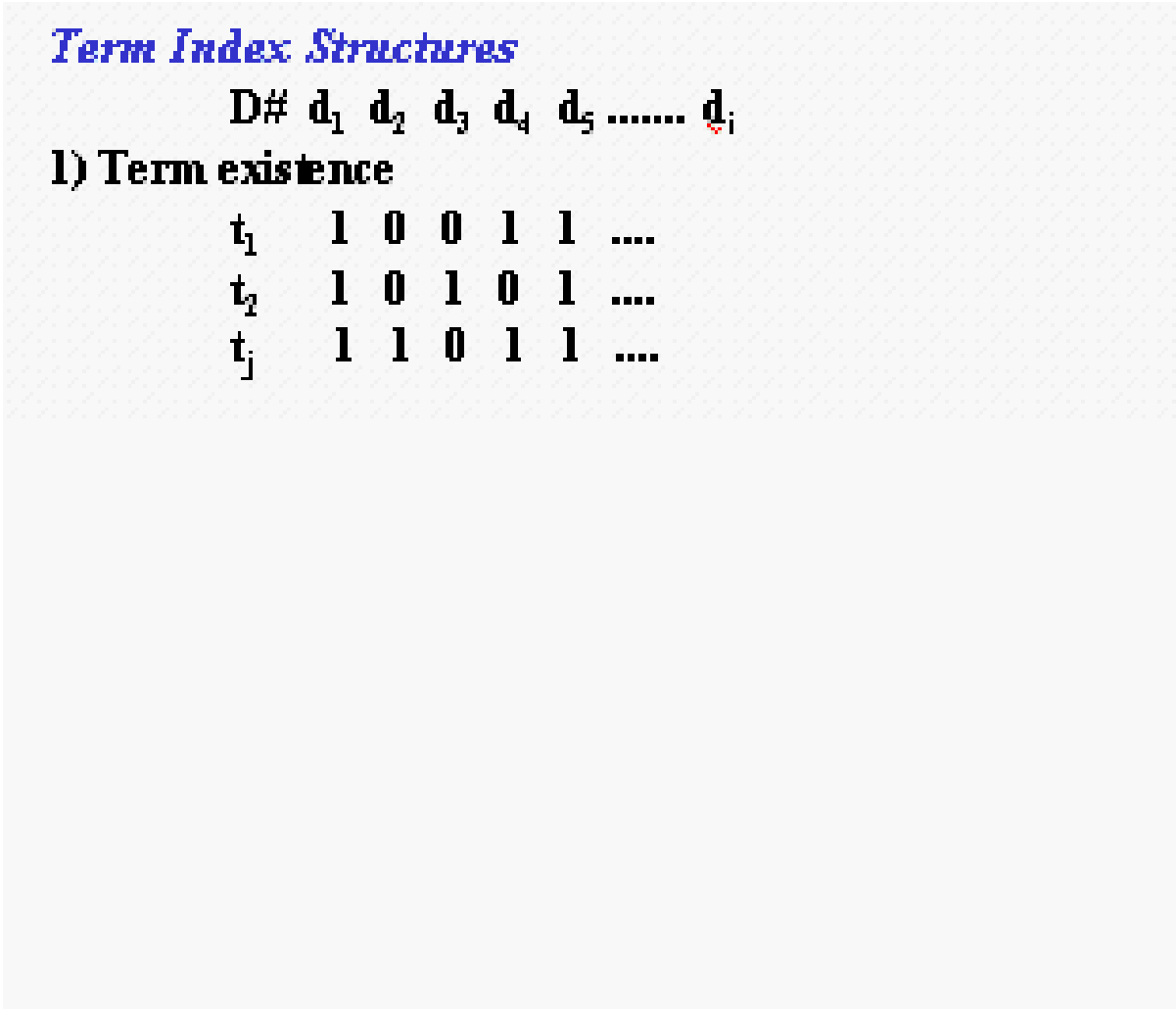
For hver søketerm som forekommer i et dokument registrerer vi

1. Hvilket dokument det forekommer i (som før),
2. hvilket kapittel/avsnitt,
3. hvilken setning, og
4. hvilket ord søketermen forekommer som.

Vi kan nå bruke dette (spesielt) som grunnlag for bedre rangering.

Termindekser

Hvor mye ekstra struktur må vi nå behandle?



Termindekser

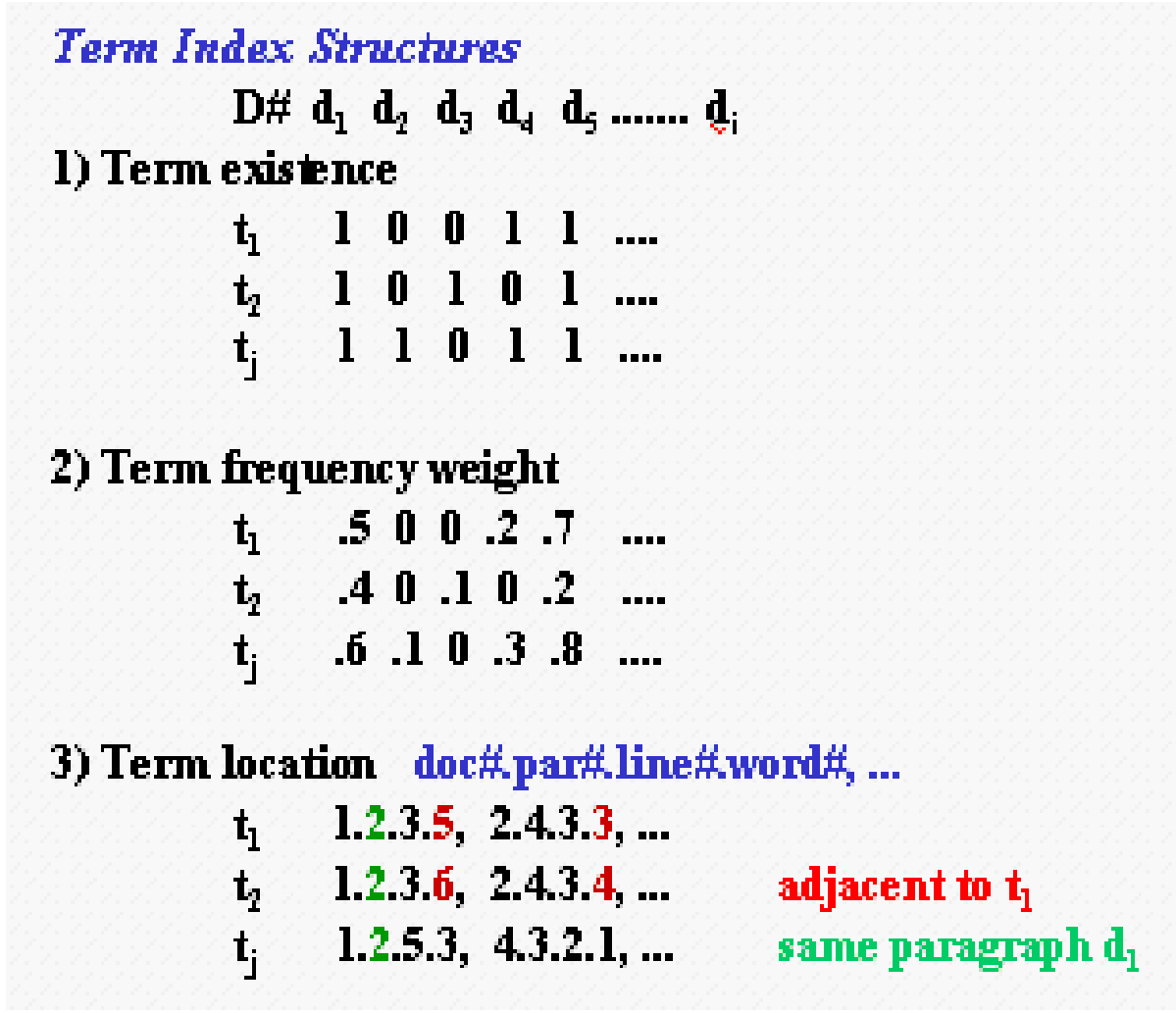
Hvor mye ekstra struktur må vi nå behandle?

Term Index Structures

	D#	d_1	d_2	d_3	d_4	d_5	.....	d_i
1) Term existence								
	t_1	1	0	0	1	1	....	
	t_2	1	0	1	0	1	....	
	t_j	1	1	0	1	1	....	
2) Term frequency weight								
	t_1	.5	0	0	.2	.7	....	
	t_2	.4	0	.1	0	.2	....	
	t_j	.6	.1	0	.3	.8	....	

Termindekser

Hvor mye ekstra struktur må vi nå behandle?



Termfrekvens

Utenom hvilke termer som forekommer i hvilke dokumenter, kan vi også holde styr på frekvens.

Vi antar at ord som forekommer (relativt) oftere i et gitt dokument enn i andre er nært knyttet til (det semantiske) innholdet i dokumentet.

Termfrekvens

Utenom hvilke termer som forekommer i hvilke dokumenter, kan vi også holde styr på frekvens.

Vi antar at ord som forekommer (relativt) oftere i et gitt dokument enn i andre er nært knyttet til (det semantiske) innholdet i dokumentet.

For å få en godt anslag av hvor signifikant en frekvens er, regner vi den ut relativ til dokumentlengden.

Mye lettere å *rangere* resultatene enn med forekomststesting.

Relativ frekvens

La[1]

W_{ij} representere verdien i celle (i, j)

tf_{ij} term i's frekvens i dokument j $= \frac{n_{ij}}{\sum_k n_{kj}}$

N antall dokumenter

df_i antall dokumenter hvor term i forekommer

[1] OBS! Motsatt rekkefølge på subteksten.

Relativ frekvens

La[1]

W_{ij} representere verdien i celle (i, j)

tf_{ij} term i's frekvens i dokument j $= \frac{n_{ij}}{\sum_k n_{kj}}$

N antall dokumenter

df_i antall dokumenter hvor term i forekommer

Mye brukt relativ frekvens (*tf-idf*):

$$W_{ij} = tf_{ij} \cdot \underbrace{\log\left(\frac{N}{df_i}\right)}_{idf_i}$$

Her vil idf_i (*invertert dokument frekvens*) være høy dersom term i forekommer i *få* dokumenter.

[1] OBS! Motsatt rekkefølge på subteksten.

Søking

Forslag til søketermer?

ONCE upon a midnight dreary, while I pondered, weak and weary,
Over many a quaint and curious volume of forgotten lore—
While I nodded, nearly napping, suddenly there came a tapping,
As of some one gently rapping, rapping at my chamber door.
“ ’Tis some visitor,” I muttered, “tapping at my chamber door—
Only this and nothing more.”

Søking

Forslag til søketermer?

Metadata angitt av en ekspert fra NYPL

Image Caption: Miniature of St. Martin dividing his cloak, full border design with human figure, rubric, large blue initial.

In: Renaissance and medieval manuscripts collection, ca. 850-ca. 1600. > [Book of Hours, use of Rouen] (created ca. 1500)

Alternate Title: [Book of Hours, use of Rouen]

Library Division: Humanities and Social Sciences Library / Manuscripts and Archives Division

Description: ff. ii + 99 + ii, 150 x 106 mm.

Item/Page/Plate Number: f. 23

Specific Material Type: mss. text

Collection Guide: Medieval and Renaissance Illuminated Manuscripts from Western Europe

Digital Image ID: 426030

Digital Record ID: 248347

NYPL Call Number: MA 49

CBIR - Content Based Image Retrieval

Å søke basert på en beskrivelse av *innholdet* i (en del av) et bilde (lyd/video) kalles *Content Based Image Retrieval*.

Vi har mange attributter som kan danne grunnlag for søking:

Atomære attributter (kolonner i database tabellen)

Term indekser for termer fra semantisk metadata (tittel, beskrivelse, ...)

Strukturelle indekser automatisk beregnede egenskaper (størrelse, farge, ...)

Semantiske egenskaper automatisk beregnede egenskaper (objekter, hendelser, ...)

CBIR - Content Based Image Retrieval

Det finnes ingen gode generelle algoritmiske løsninger for å trekke ut semantisk mening fra et lyd/bilde/video filer.

CBIR - Content Based Image Retrieval

Det finnes ingen gode generelle algoritmiske løsninger for å trekke ut semantisk mening fra et lyd/bilde/video filer.

... men! Det finnes mange gode domene-spesifikke algoritmer:

Medisin	identifisering av benbrudd, svulster, ...
CSI	fingeravtrykk, ansikt, ...
Militær	ubåter, skikkelser, ...
Tale-til-tekst	tillater søking som i tekstdokumenter
Takt	BPM, gjentagende rytmer, ...
Bevegelses	følge et objekt som beveger seg i rommet, ...

Google et. al.

Hvordan vet Google at dette er et bilde av Bergen?[2]

[2] “Hvorfor tror ...”

Google et. al.

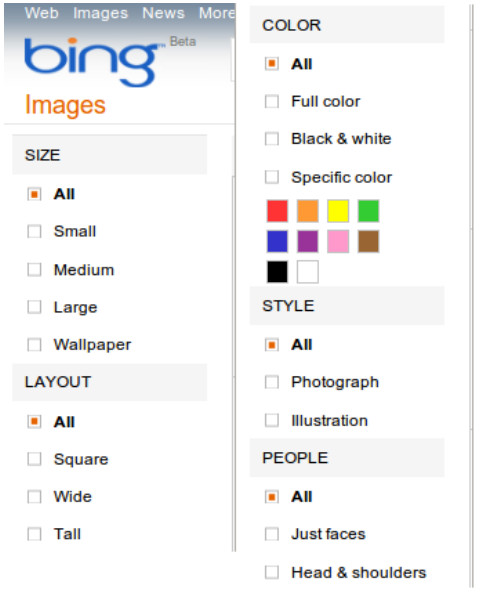
... fordi det kom herfra:

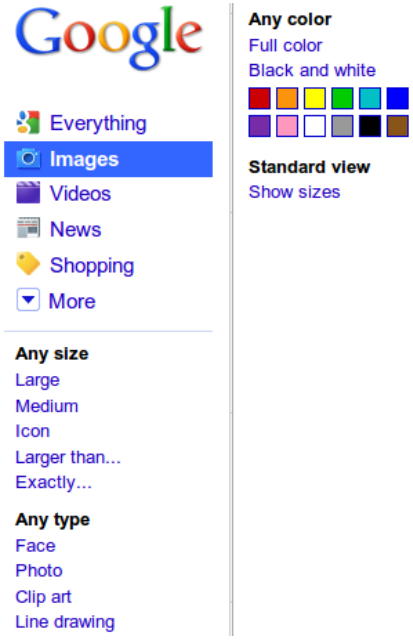
Multimedia

Vev-sider, bøker og lignende dokumenter:

1. Del inn dokumentet i kapitler/avsnitt
   (hold rede på strukturen hvis du deler dokumentet)
2. Finn termer i teksten og konstruer en term posisjon indeks
3. Ord som forekommer nær bildet legger vi til som beskrivelse av bildet
4. Beregn/trekk ut eventuelle andre metadata som skal legges inn i bildedatabasen

Bildesøk på internett





Spennende teknologier:
`http://www.youtube.com/watch?v=zGlsnEnKFoI` (SoundHound)
`http://www.youtube.com/watch?v=7kOotqpJyUg` (Google Goggles)
`http://www.youtube.com/watch?v=5Om48Yz3X8k` (Search by Sketch)

BSBI-indeksering

Problem: indeksering av stor tekstmengde.

For å spare plass gir vi hver term en unik ID.

Da kan vi lagre (termID `INT`, docID `INT`)
i stedet for f.eks. (term `VARCHAR(25)`, docID `ID`).

Dette krever en ekstra struktur for term $\rightarrow$ termID.

Vi kan ikke hoppe frem og tilbake i hele (termID, docID)-listen,
den er for stor til å passe i minnet.

BSBI

```
BSBIndexConstruction()
1  n ← 0
2  while (all documents have not been processed)
3  do n ← n + 1
4     block ← ParseNextBlock()
5     BSBI-Invert(block)
6     WriteBlockToDisk(block, f_n)
7  MergeBlocks(f_1, ..., f_n; f_merged)
```

ParseNextBlock leser inn en del av dokumentmassen og leverer en liste med (`termID`,`docID`)-par, og tar hånd om `termID`ene gitt til termene.
BSBI-Invert sorterer listen fht. `termID` og slår sammen duplikater.

BSBI

```
BSBIndexConstruction()
1  n ← 0
2  while (all documents have not been processed)
3  do n ← n + 1
4     block ← ParseNextBlock()
5     BSBI-Invert(block)
6     WriteBlockToDisk(block, f_n)
7  MergeBlocks(f_1, ..., f_n; f_merged)
```

Deler hele dokumentmassen i deler som alle passer i minnet.

Konstruerer en sortert indeks for hver bolk. (5)
(sorter, så slå sammen elementer med lik termID)

Fletter dem sammen. (7)

Store samlinger er *for* store

Hvis vi har store dokumentsamlinger er indekseringsprosessen svært omfattende.

Vi må gjøre mye analyse på mye data.

Heldigvis kan arbeidsoppgavene deles opp.

Store samlinger er *for* store

Hvis vi har store dokumentsamlinger er indekseringsprosessen svært omfattende.

Vi må gjøre mye analyse på mye data.

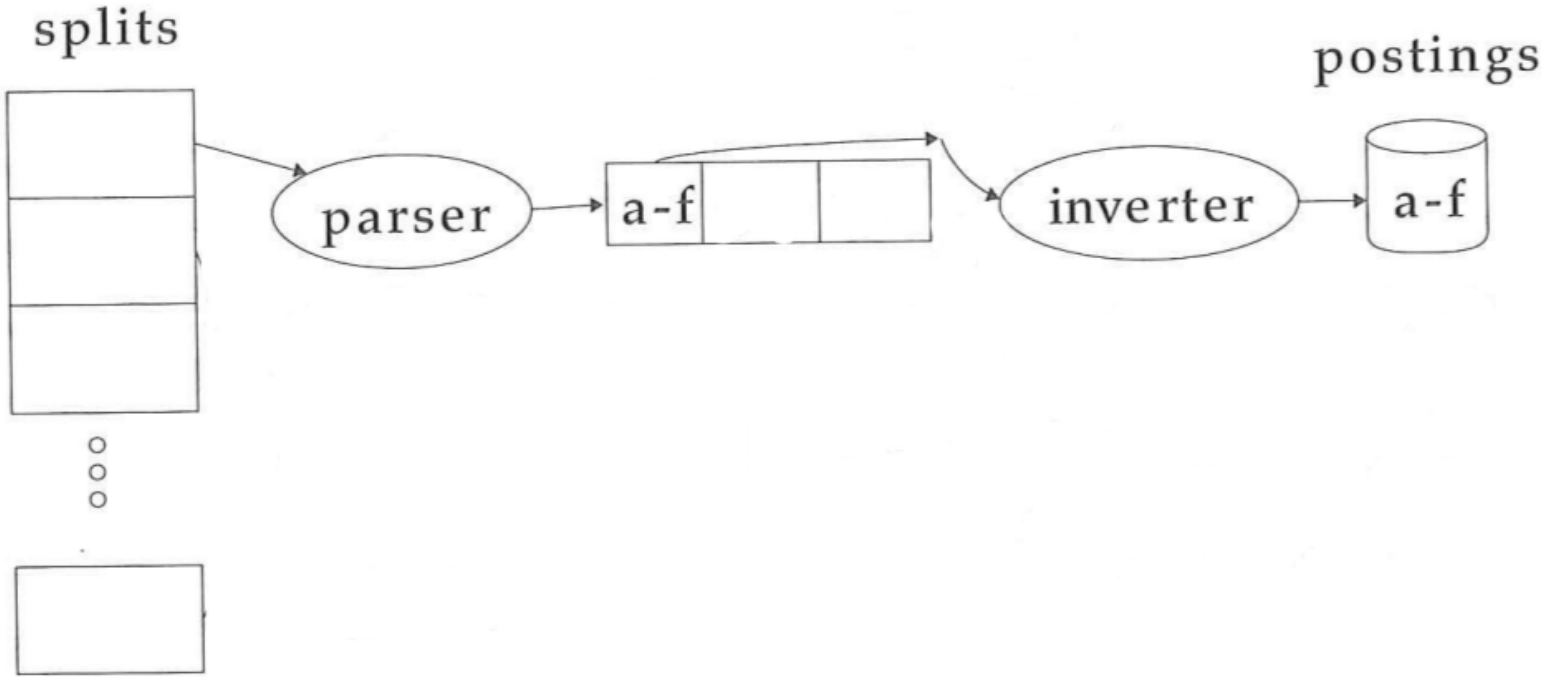
Heldigvis kan arbeidsoppgavene deles opp... nesten.

map del opp samlingen til håndterlige biter, fordel bitene for parsing

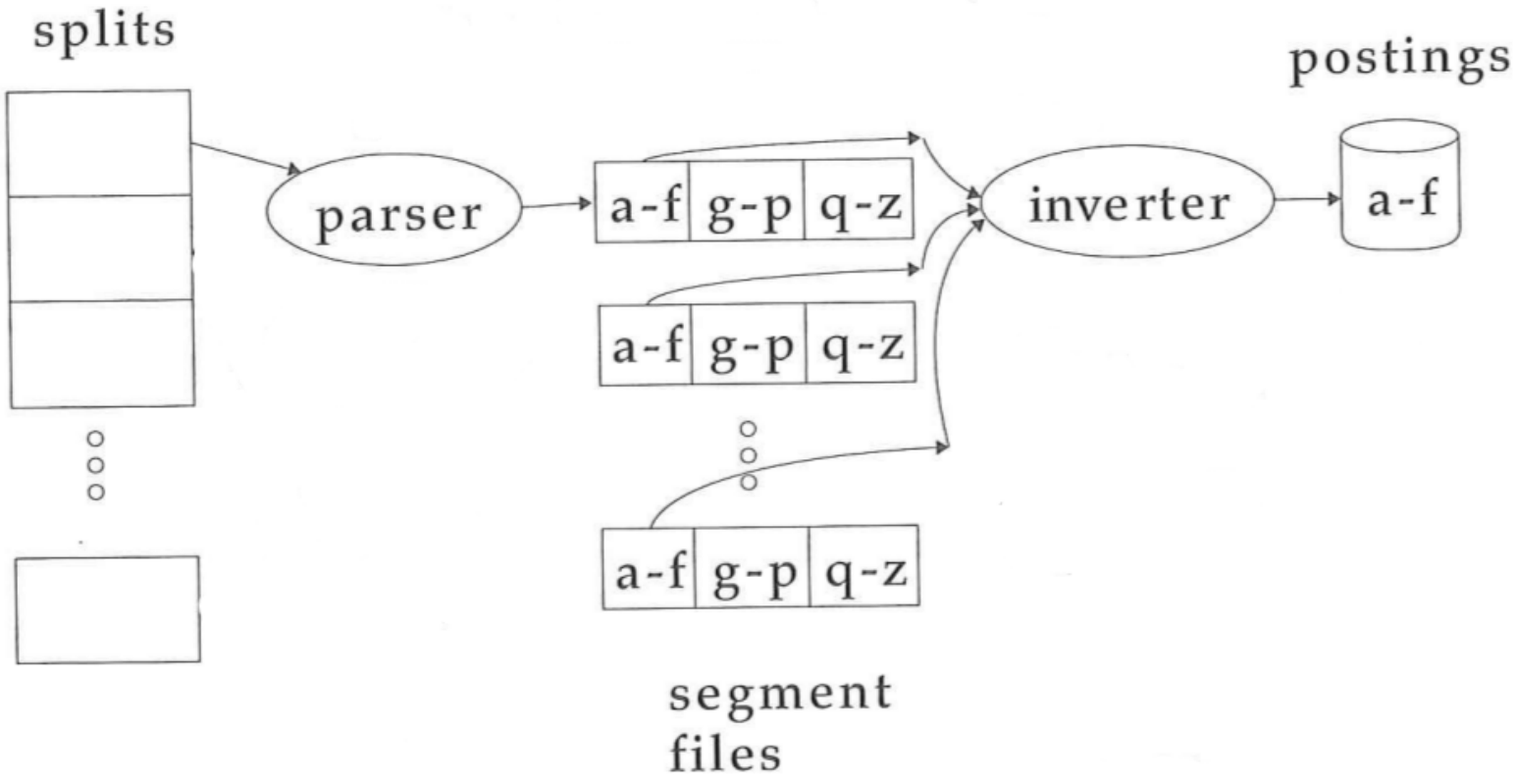
reduce konstruer en indeksering for hvert segment (fra alle segmentfilene for det segmentet)

Kordinering av term $\rightarrow$ termID er et problem.

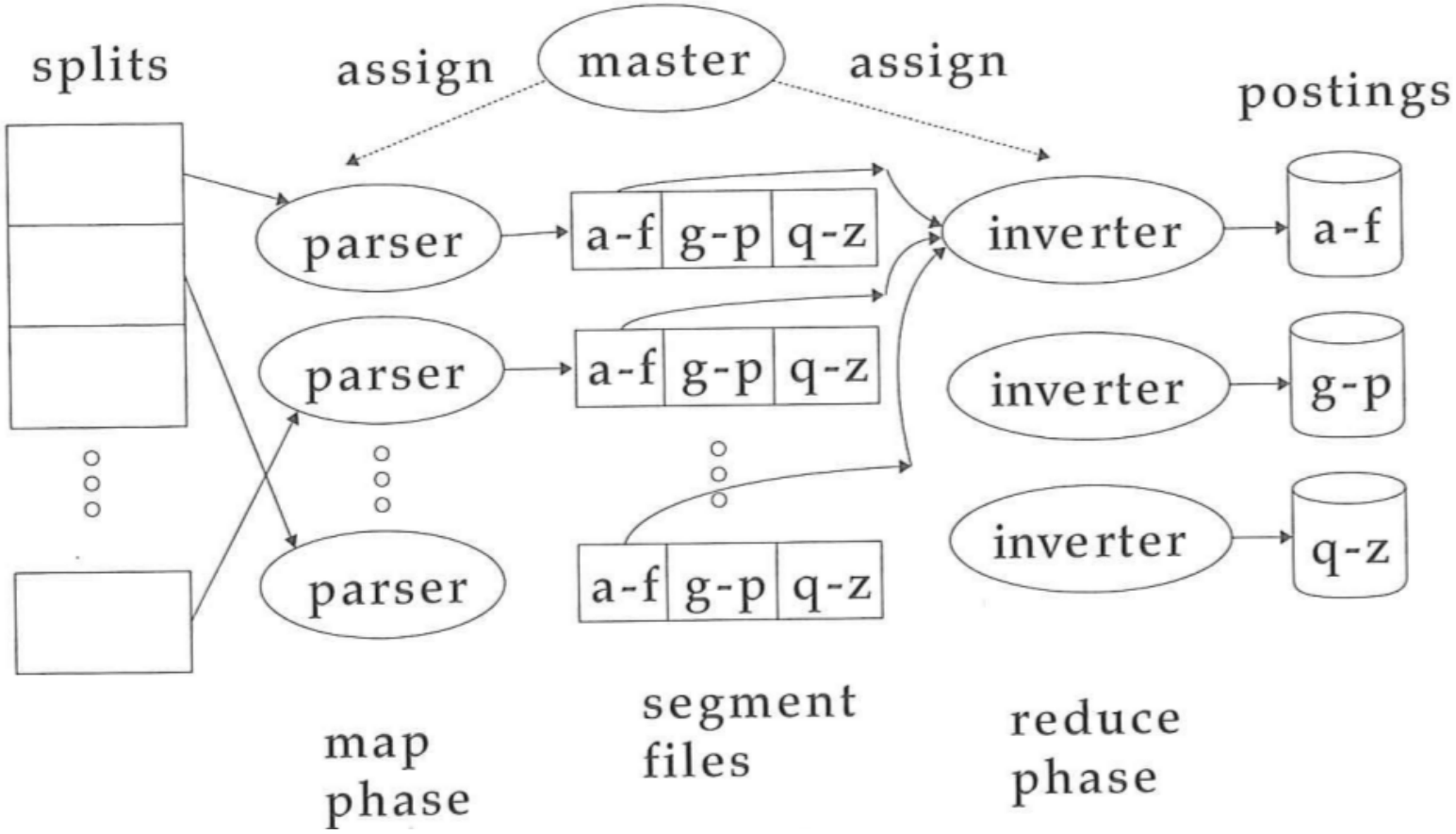
Distribuert



Distribuert



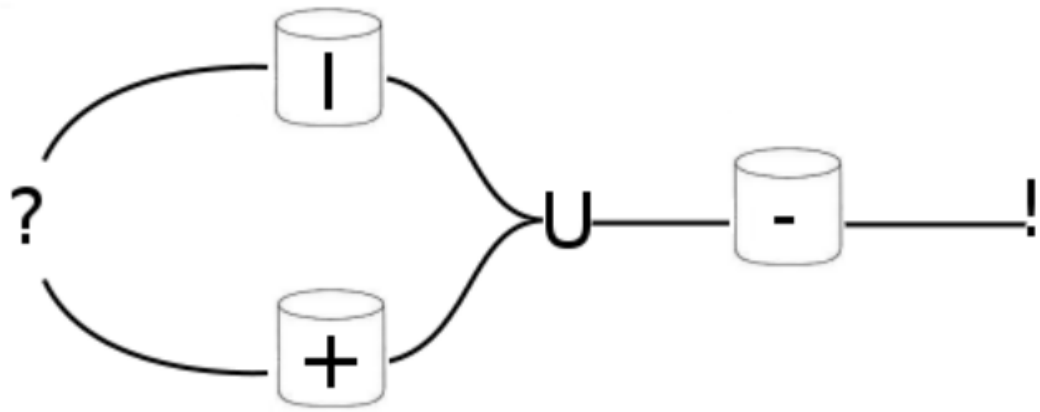
Distribuert



Dynamisk

Hvis dataene våre endrer seg, blir indekseringen vår uriktig.

Da har vi to alternativer: konstruer en ny indeksering, eller



Dynamisk

Hvis dataene våre endrer seg, blir indekseringen vår uriktig.

Da har vi to alternativer: konstruer en ny indeksering, eller