Recuperació de la Informació (REIN)

Grau en Enginyeria Informàtica

Departament de Ciències de la Computació (CS)

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Politècnica Superior d'Enginyeria
de Vilanova i la Geltrú

Índex

Respondre a la consulta

Un mal algorisme:

entra consulta q;
for cada document d a la base de dades:
    comprova si d "coincideix" amb q;
    if "coincideix", afegeix el seu `docid` a la llista L;
torna la llista L (ordenada d'alguna manera?);

Respondre a la consulta

Un mal algorisme:

entra consulta q;
for cada document d a la base de dades:
    comprova si d "coincideix" amb q;
    if "coincideix", afegeix el seu `docid` a la llista L;
torna la llista L (ordenada d'alguna manera?);

El temps de resposta a una consulta hauria de ser independent de la mida de la base de dades.

Probablement, hauria de ser proporcional a la mida de la resposta.

Estructura de dades central

Dels termes als documents

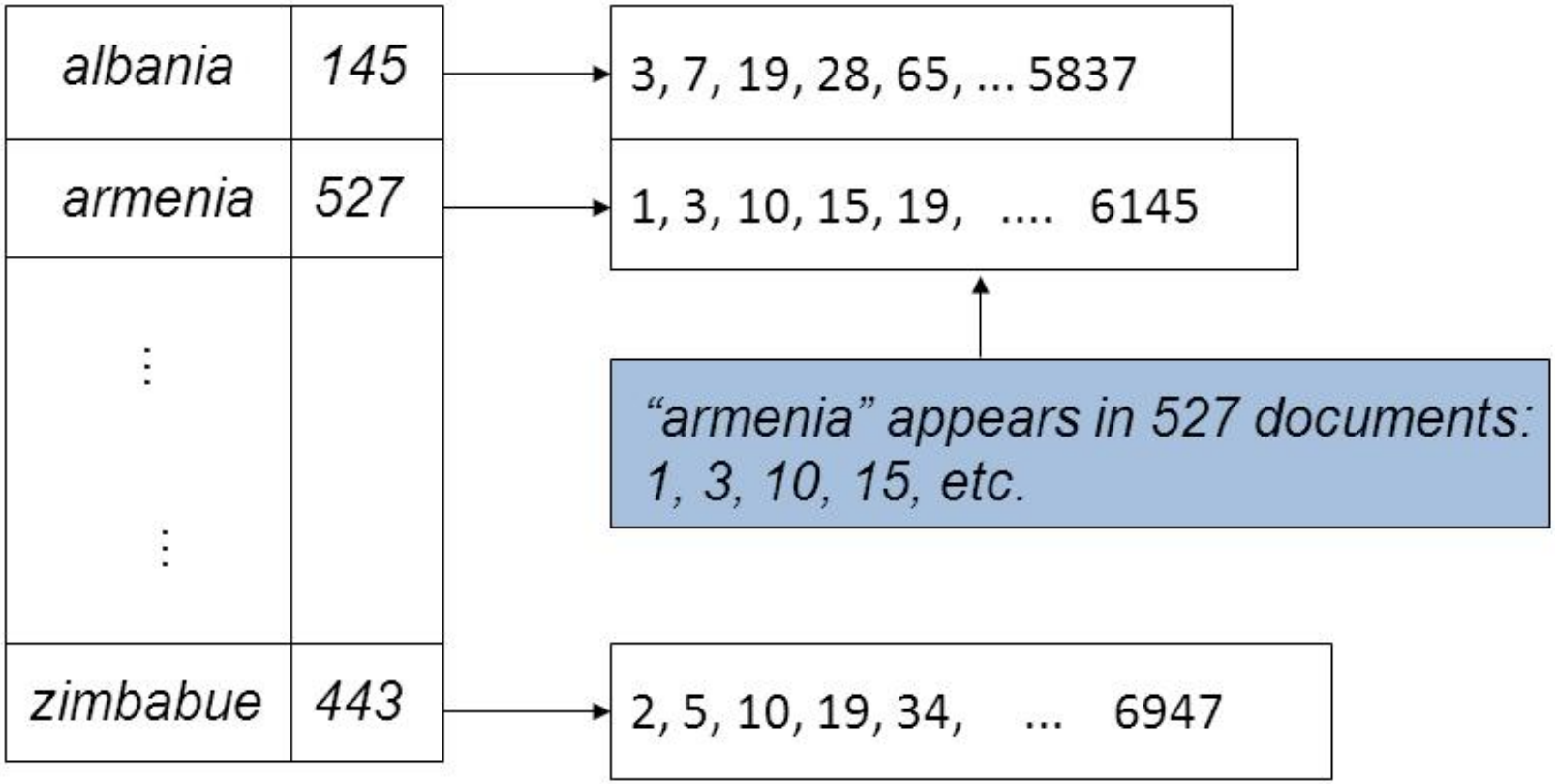
Un vocabulari o lèxic o diccionari, normalment a memòria, que manté tots els termes indexats (*set*, *map*...); i, a més...

Índex invertit

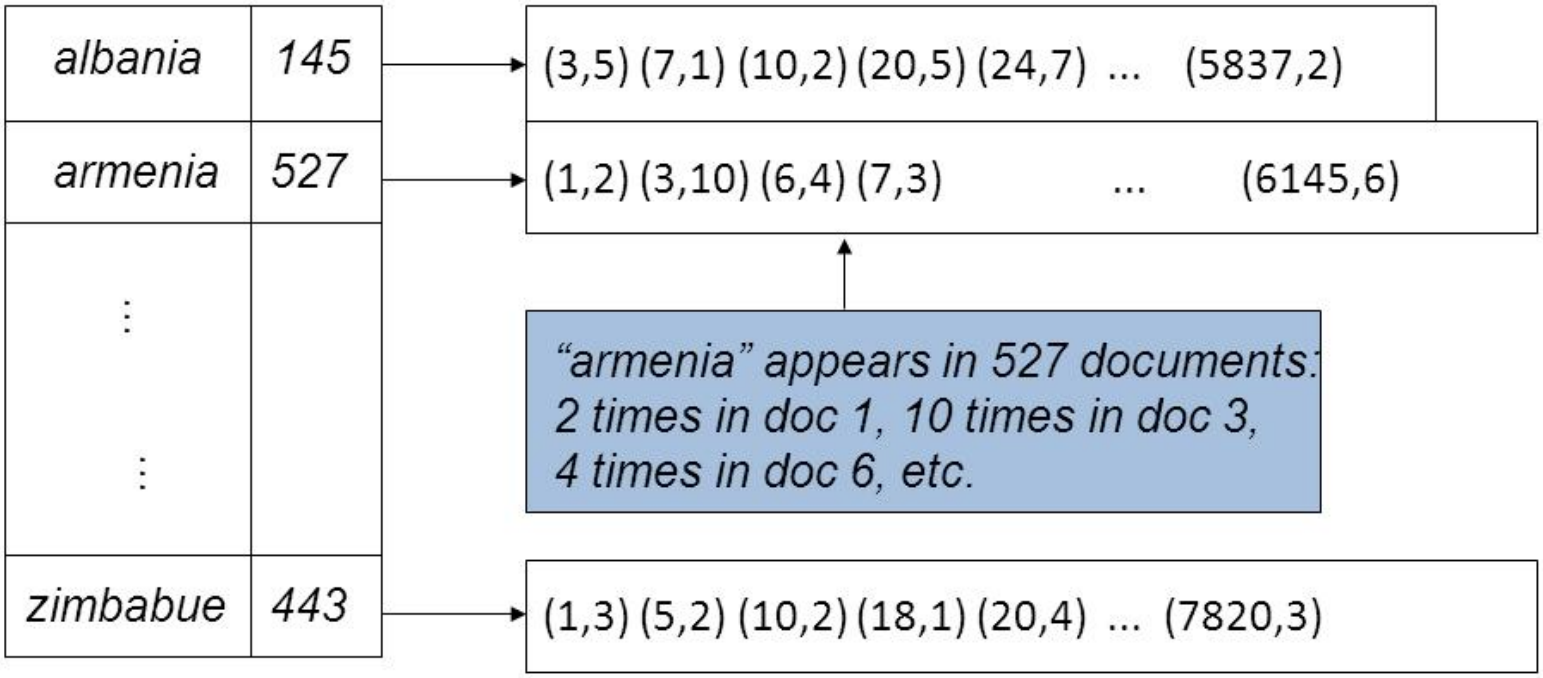
Estructura de dades crucial per la indexació.

- Ha de permetre l'operació:
  - "donat el terme t, recupera tots els documents que el continguin".
- Ha de permetre aquesta operació (i variants) de forma molt eficient.
- Es construeix en temps de preprocessament, no en el moment de la consulta: Pot dedicar-se força temps en la seva construcció.

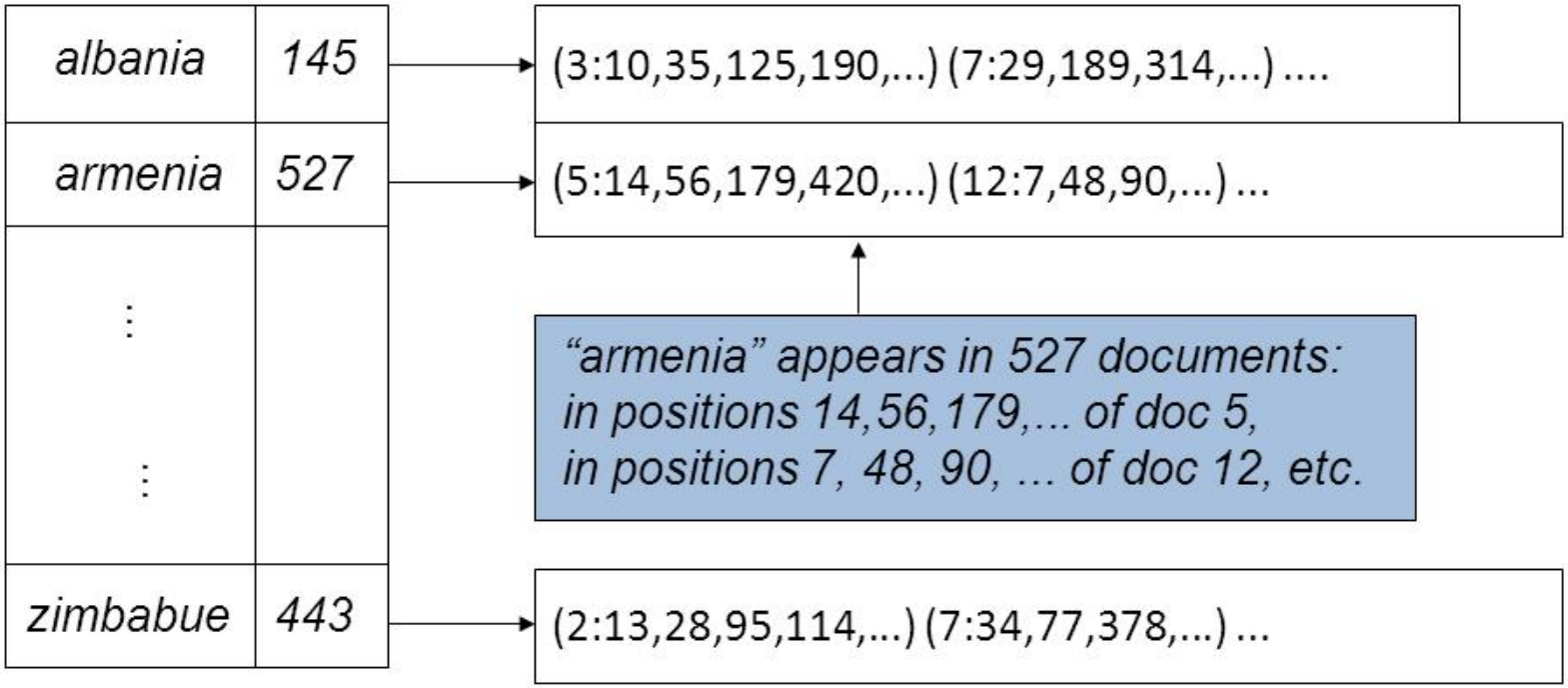
Índex invertit: Variant 1



Índex invertit: Variant 2



Índex invertit: Variant 3



Índex invertit i *Posting lists*

L'índex invertit: llistes d'incidències /*posting lists*

Assignem un `identificador de document`, docid, a cada document. El diccionari pot cabre a la memòria RAM per aplicacions de mida mitjana.

Per cada terme indexat

una *posting list*: llista de *docid* (i potser altra info) on el terme apareix.

- Perfecte si cabés a memòria però és poc probable.
- A més, les *posting lists* estan
  - gairebé sempre ordenades per *docid*,
  - sovint comprimides: per minimitzar la info a llegir de disc!

Implementació del model booleà, I

El més senzill: Recórrer les *posting lists*

Consulta conjuntiva: a AND b

- Intersecció de les *posting lists* d'a i de b.
- Si estan ordenades: intersecció tipus fusió (*merge*).
- Cost (temps): de l'ordre de la suma de les longituds de les llistes.

```
intersect(input lists L1, L2, output list L):
  while ( not L1.end() and not L2.end() )
   if (L1.current() < L2.current()) L1.advance();
   else if (L1.current() > L2.current()) L2.advance();
        else { L.append(L1.current());
               L1.advance(); L2.advance(); }
```

Implementació del model booleà, II

El més senzill

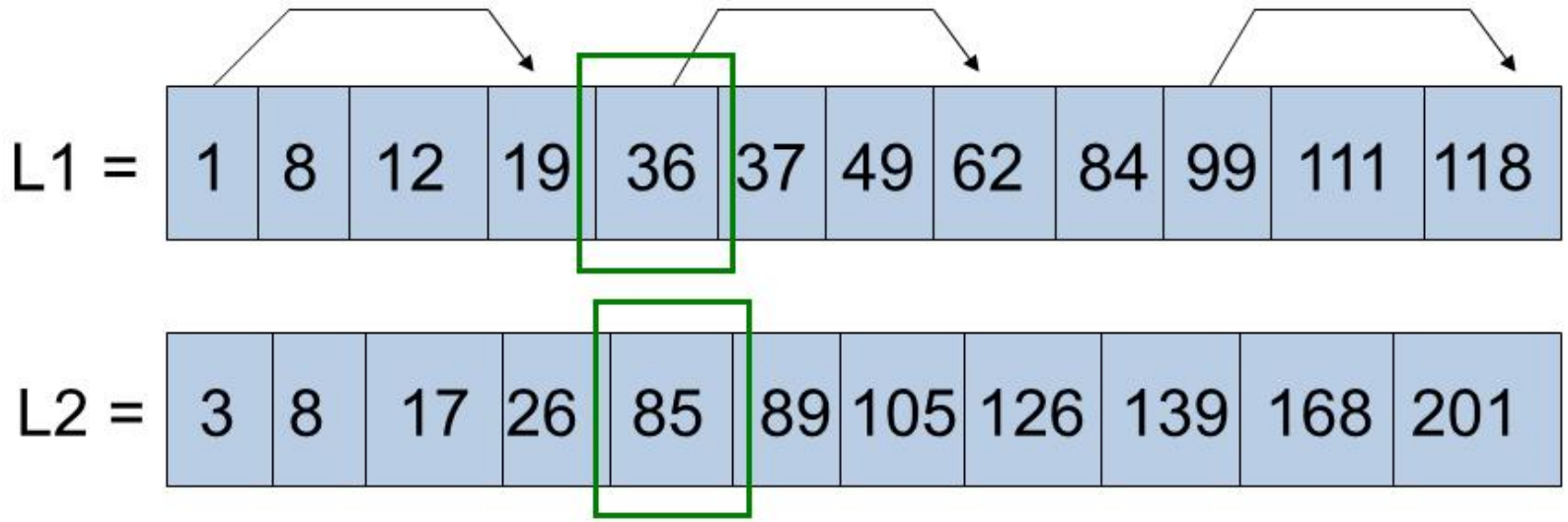
- Per la consulta disjuntiva (OR), fer una fusió de forma similar però fent una unió.
  - Cost (temps): també de l'ordre de la suma de les longituds de les llistes.
- Alternativa: recórrer una llista buscant cada `docid` en l'altra via cerca binària.
  - Cost (temps): longitud de la llista més curta multiplicat pel $\log$ de la longitud de la més llarga.

Exemple:

- $|L1| = 1000$, $|L2| = 1000$:
  - recorregut seqüencial: 2000 comparacions,
  - cerca binària: $1000 * 10 = 10000$ comparacions.
- $|L1| = 100$, $|L2| = 10000$:
  - recorregut seqüencial: 10100 comparacions,
  - cerca binària: $100 * \log(10000) = 1400$ comparacions.

Implementació del model booleà, III

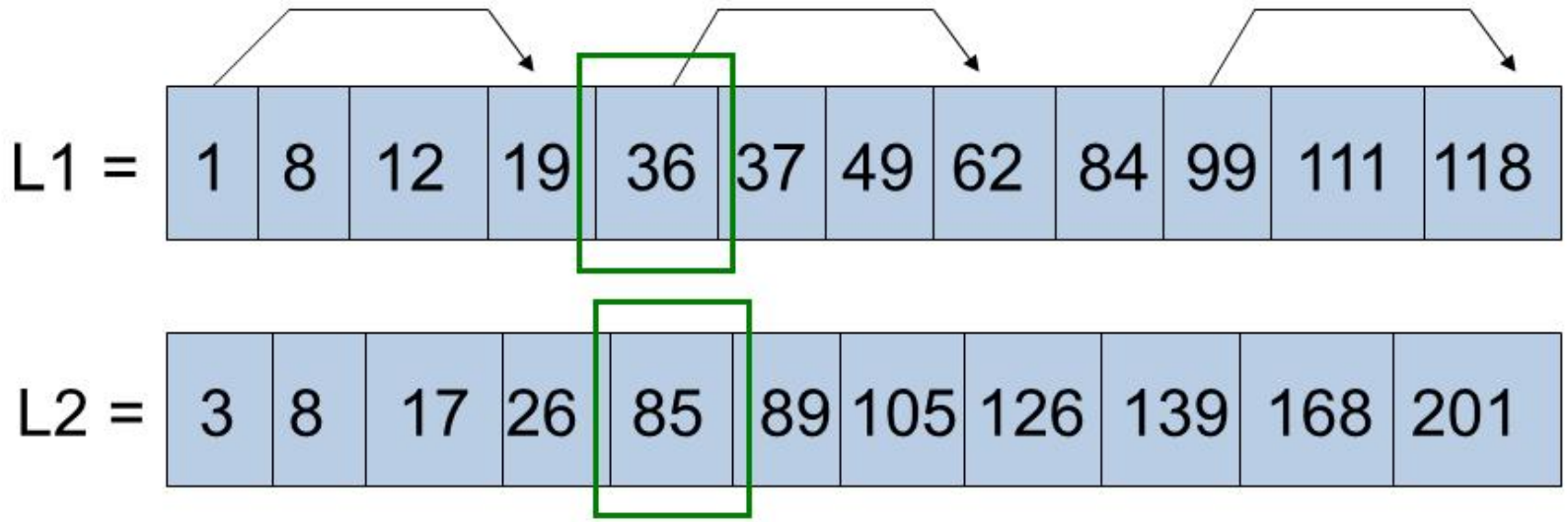
Temps d'intersecció sub-lineal: *Skip pointers*



- Hem fusionat 1...19 i 3...26.
- Ara estem mirant 36 i 85.
- Com que pointer(36)=62 $<$ 85, podem saltar al 84 en la llista L1.

Implementació del model booleà, IV

Temps d'intersecció sub-lineal: *Skip pointers*



- Punter endavant des d'alguns elements.
- O bé saltar cap al segment següent, o bé buscar dins el següent segment (un cop).
- Òptim: a la RAM $\sqrt{|L|}$ punters de longitud $\sqrt{|L|}$.
- Difícil fer-ho bé, especialment si les llistes estan al disc.

Optimització de consultes i estimació de costos, I

Les consultes poden ser avaluades seguint diferents plans d'execució. Per exemple, la consulta a AND b AND c:

- (a AND b) AND c
- (b AND c) AND a
- (a AND c) AND b

O la consulta (a AND b) OR (a AND c):

- a AND (b OR c)

El cost d'un pla d'execució depèn de les longituds de les llistes i de les longituds de les llistes intermèdies.

Optimització de consultes i estimació de costos, II

Exemple

Consulta: (a AND b) OR (a AND c AND d).

Suposem: $|La| = 3000$, $|Lb| = 1000$, $|Lc| = 2500$, $|Ld| = 300$.

- Tres interseccions i la unió, en l'ordre donat: cost màxim 13600.
- Canviant l'ordre, ((d AND c) AND a): redueix el cost màxim a 11400.
- Reescriure-la com a AND (b OR (c AND d)): redueix el cost màxim a 8400.

Implementació del model vectorial, I

Plantejament del problema

Fixada la mesura de similitud $sim(d, q)$:

Recupera

els documents d_i que tenen una similitud amb la consulta q

- o bé
  - sota d'un llindar sim_{min}, o
  - els r primers d'acord amb la similitud, o
  - tots els documents,
- ordenats per ordre decreixent de similitud a la consulta q.

Ha de respondre ràpidament (per tant, cal vigilar la interacció amb el disc) i fer un ús raonable de la memòria.

Implementació del model vectorial, II

Solució no vàlida

Recórrer tots els documents, accedir als termes per calcular la similitud, filtrar segons sim_{min}, i ordenar el resultat...

...no funcionarà.

Implementació del model vectorial, III

Observacions

La majoria dels documents contenen una petita proporció dels termes del vocabulari.

Les consultes, normalment, estan formades per pocs termes.

Implementació del model vectorial, III

Observacions

La majoria dels documents contenen una petita proporció dels termes del vocabulari.

Les consultes, normalment, estan formades per pocs termes.

Només una petita proporció dels documents serà rellevant.

Implementació del model vectorial, III

Observacions

La majoria dels documents contenen una petita proporció dels termes del vocabulari.

Les consultes, normalment, estan formades per pocs termes.

Només una petita proporció dels documents serà rellevant.

Hi ha un llindar conegut a priori per a r, lligat a la mida de la resposta.

Implementació del model vectorial, III

Observacions

La majoria dels documents contenen una petita proporció dels termes del vocabulari.

Les consultes, normalment, estan formades per pocs termes.

Només una petita proporció dels documents serà rellevant.

Hi ha un llindar conegut a priori per a r, lligat a la mida de la resposta.

Tenim l'índex invertit!

Implementació del model vectorial, IV

Idea

Només cal invertir els bucles de l'algorisme inicial:

- Bucle extern sobre els termes t que apareixen en la consulta.
- Bucle intern sobre els documents que contenen el terme t:
  - per això tenim l'índex invertit.
- Acumular la similitud pels documents visitats.
- A l'acabar, es fa la normalització i l'ordenació.

Compressió de l'índex, I

Per què?

Una gran part del temps de consulta-resposta es dedica a

carregar les *posting lists* dels discs a la RAM.

Compressió de l'índex, I

Per què?

Una gran part del temps de consulta-resposta es dedica a

carregar les *posting lists* dels discs a la RAM.

Cal minimitzar la quantitat de bits a transferir.

Compressió de l'índex, I

Per què?

Nbits per representar un nombre -> Log_2 (N)

Una gran part del temps de consulta-resposta es dedica a

carregar les *posting lists* dels discs a la RAM.

Cal minimitzar la quantitat de bits a transferir.

Els esquemes de compressió de l'índex usen:

- Els `docid` ordenats en ordre creixent.
- Les freqüències són, normalment, valors petits.
- Es poden usar menys de 32 bits per cada valor.

si no guardo en in, sino la diferencia entre ells es necesiten menys bits

Compressió de l'índex, II

Tema de treball personal. Mireu, almenys:

- Codificació unària.
- Codificació de longitud variable + Codi Elias Gamma.
- Ratis de compressió de dades.

Podeu buscar informació en els llibres llistats a la secció de Presentació de l'assignatura.