

# Capitolul 10

## Data mining – cautare pe web

# Căutarea pe web

## ◆ Puncte importante :

1. *Rangul paginii*, pentru descoperirea celor mai "importante" pagini de web, utilizat de Google.
2. *Indecși și autorități*, o evaluare mai detaliată a importanței paginilor web utilizând o variantă a calculului de valori proprii utilizată pentru rangul paginii.

# Rangul paginii

- ◆ Intuitiv rezolvăm problema definiției "importanței" recursiv : o pagina este importantă dacă pagini importante conțin legături către ea.
- ◆ Creăm o matrice stochastică a Internetului astfel :
  - ◆ Fiecare pagină  $i$  corespunde liniei  $i$  și coloanei  $i$  a matricii.
  - ◆ Dacă pagina  $j$  are  $n$  succesori (legături), atunci elementul  $i, j$  al matricii este  $1/n$  dacă pagina  $i$  este unul dintre acești succesori ai paginii  $j$  și 0 altfel.

# Rangul paginii

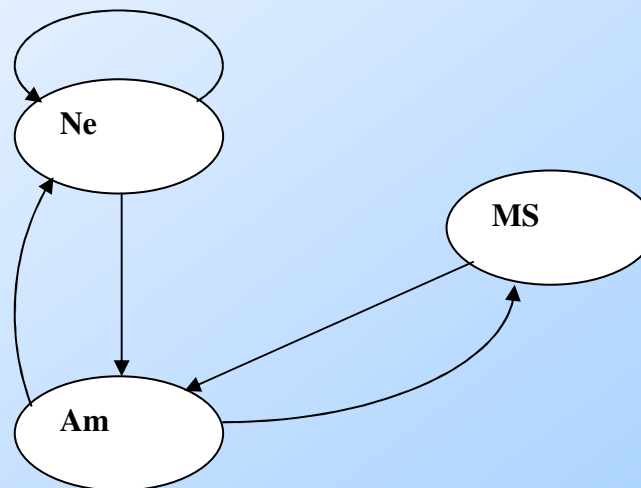
- ◆ Intuiția care stă în spatele acestei matrici este :
- ◆ Să ne imaginăm că inițial fiecare pagină are o unitate de importanță.
- ◆ La fiecare pas fiecare pagină își împarte importanța între succesorii săi și primește noi fracțiuni de importanță de la predecesorii săi.

# Rangul paginii

- ◆ Eventual, importanța fiecărei pagini atinge o limită care este componenta corespunzătoare ei din vectorul principal de valori proprii al matricii.
- ◆ Această importanță este de asemenea probabilitatea ca un navigator pe web, pornind de la o pagină aleatoare și urmând legături aleator alese din fiecare pagină, să ajungă la pagina în discuție după o lungă serie de legături.

# Exemplul 1

- ◆ In 1839 Internetul consta din doar trei pagini : Netscape, Amazon si Microsoft. Legăturile între aceste trei pagini erau ca în figura urmatoare:



# Exemplul 1

- ◆ Fie  $[n, m, a]$  vectorul importanței pentru cele trei pagini : Netscape, Microsoft respectiv Amazon.
- ◆ Atunci ecuația care descrie valorile asimptotice ale acestor trei variabile este :

$$\begin{bmatrix} n \\ m \\ a \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 \\ 1/2 & 1 & 0 \end{bmatrix} \begin{bmatrix} n \\ m \\ a \end{bmatrix}$$

# Exemplul 1

- ◆ Prima coloană a matricii reflectă faptul că Netscape își divide importanța între el însuși și Amazon. A doua coloană că Microsoft dă toată importanța sa către Amazon.
- ◆ Putem rezolva ecuații ca aceasta începând cu aserțiunea că  $n = m = 1$  și aplicând repetat matricea la estimarea curentă a acestor valori.



# Exemplul 1

- ◆ Primele patru iterații dau următoarele estimări :

n	=	1	1	5/4	9/8	5/4
m	=	1	1/2	3/4	1/2	11/16
a	=	1	3/2	1	11/8	17/16

- ◆ La limită, soluția este  $n = a = 6/5$  ;  $m = 3/5$ .

# Observatii

- ◆ De notat că nu putem să obținem niciodată valorile absolute ale lui  $n$ ,  $m$  și  $a$  ci ***doar raportul lor***, de vreme ce aserțiunea inițială că fiecare a pornit de la 1 a fost arbitrară.
- ◆ Deoarece matricea este stochastică (suma pe fiecare coloană este 1), procesul de *relaxare* de mai sus converge către ***vectorul principal de valori proprii*** al matricii.

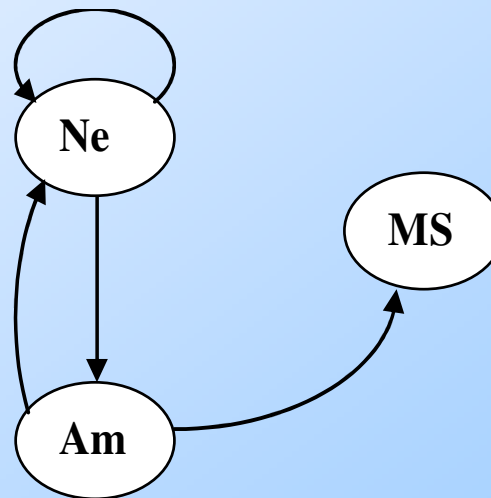
# Probleme cu grafuri reale

◆ 2 tipuri de probleme:

1. *Dead end*: o pagină care nu are succesori nu are către cine să-și trimită importanța. Eventual, toată importanța "se va scurge" din Internet
2. *Capcane* : un grup de una sau mai multe pagini care nu au legături către pagini din afara grupului vor acumula eventual toată importanța din Internet.

## Exemplul 2: Dead end

- ◆ Să presupunem că Microsoft încearcă să profite că este un monopol înlăturând toate legăturile din situl său. Noul Internet este ca în figura următoare:



## Exemplul 2

- ◆ Ecuatia matriciala este in acest caz:

$$\begin{bmatrix} n \\ m \\ a \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 \\ 1/2 & 0 & 0 \end{bmatrix} * \begin{bmatrix} n \\ m \\ a \end{bmatrix}$$

- ◆ Se observa ca suma pe coloane nu mai este intotdeauna 1 (coloane cu suma nula)

## Exemplul 2

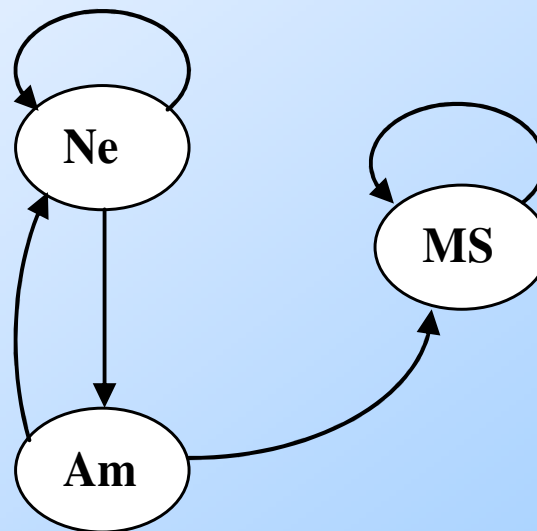
- ◆ Primii patru pași ai soluției iterative sunt :

n	=	1	1	3/4	5/8	1/2
m	=	1	1/2	1/4	1/4	3/16
a	=	1	1/2	1/2	3/8	5/16

- ◆ In acest caz, fiecare dintre  $n$ ,  $m$  și  $a$  tinde catre 0; i.e. toată importanța se scurge afară.

## Exemplul 3

- ◆ Microsoft decide să nu folosească decât legături către el însuși de acum încolo. Acum, Microsoft a devenit o capcană. Noul Internet este în figura:



## Exemplul 3

- ◆ Ecuatia matriciala este in acest caz:

$$\begin{bmatrix} n \\ m \\ a \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 1 & 1/2 \\ 1/2 & 0 & 0 \end{bmatrix} * \begin{bmatrix} n \\ m \\ a \end{bmatrix}$$

- ◆ Suma pe coloane este 1 dar apar valori de 1 pe diagonala principala a matricii.



# Exemplul 3

- ◆ Primii pasi ai algoritmului produc valorile:

n	=	1	1	$3/4$	$5/8$	$1/2$
m	=	1	$3/2$	$7/4$	2	$35/16$
a	=	1	$1/2$	$1/2$	$3/8$	$5/16$

- ◆ Se observa acumularea pe linia m

# Prevenire dead end și capcane

- ◆ În loc de a aplica matricea direct, "taxăm" fiecare pagină cu o fracțiune din importanța sa curentă și distribuim importanța taxată în mod egal tuturor paginilor.
- ◆ Dacă folosim o taxă de 20% ecuația din exemplul 3 devine cea de pe transparentul urmator.

◆ Ecuatia cu taxare:

$$\begin{bmatrix} n \\ m \\ a \end{bmatrix} = 0.8 \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 1 & 1/2 \\ 1/2 & 0 & 0 \end{bmatrix} \begin{bmatrix} n \\ m \\ a \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.2 \\ 0.2 \end{bmatrix}$$

- ◆ Soluția acestei ecuații este  $n = 7/11$ ;  $m = 21/11$ ;  $a = 5/11$ .
- ◆ De notat că suma celor trei valori nu este 3 dar obținem o distribuție mult mai rezonabilă a importanței decât în Exemplul 3.

# Spam

- ◆ "Spamming" este în acest context încercarea multor situri web de a părea că sunt despre un subiect care atrage navigatorii fără ca într-adevăr să fie despre acel subiect.
- ◆ Google, ca și alte motoare de căutare, încearcă să potrivească cuvintele din cererile de căutare cu cuvinte din pagini web.
- ◆ Cu toate acestea, Google, spre deosebire de alte motoare de căutare, tinde să creadă ceea ce spun alții în textul legăturilor despre o pagină web făcând mai greu pentru aceasta să pară ca fiind despre ceva ce nu este.

# Spam

- ◆ Utilizarea rangului paginii pentru a măsura importanța în locul unei măsuri mult mai naive ca "numărul de legături către acea pagină" protejează de asemenea impotriva spamului.
- ◆ Masura naivă poate fi înșelată de un spammer care creează 1000 de pagini care se referă între ele în timp ce rangul paginii recunoaște că nici una dintre acestea nu au importanță reală.

# Indecși si autorități

- ◆ Intuitiv, definim "index" și "autoritate" într-un mod mutual recursiv: un index conține legături către multe autorități iar o autoritate este referită de mulți indecși.
- ◆ Autoritățile pot fi pagini care oferă informații despre un subiect, e.g. pagina Quest despre proiectul IBM de data mining.
- ◆ Indecșii sunt pagini care nu furnizează informații ci spun unde se găsesc informații, e.g. pagina cursului CS345.

# Indecși si autorități

- ◆ Utilizează o formalizare matricială similară cu cea de la rangul paginii dar fără restricția stochastică. Numărăm fiecare legătură ca 1, indiferent de câți succesori sau predecesori are o pagină.
- ◆ Aplicarea repetată a matricii duce la divergență, dar putem introduce un factor de scalare pentru a ține valorile calculate pentru gradul de "autoritate" sau de "indexare" pentru fiecare pagina între limite finite.

# Indecși si autorități

- ◆ Definim matricea  $A$  ale cărei linii și coloane corespund paginilor web având elementul  $A_{ij} = 1$  dacă pagina  $i$  referă pagina  $j$  și 0 altfel.
- ◆ De notat că  $A^T$ , transpusa lui  $A$ , arată ca matricea utilizată pentru calculul rangului paginilor dar  $A^T$  are 1 acolo unde matricea pentru rang are fracții.



# Indecși si autorități

- ◆ Fie  $a$  si  $h$  doi vectori iar componenta lor  $i$  corespunde gradului de autoritate respectiv indexare a paginii  $i$ . Fie  $\lambda$  și  $\mu$  factorii de scalare corespunzători care vor fi calculați mai târziu. Atunci putem afirma că:
- ◆  $h = \lambda A a$ . Adică gradul de indexare al fiecărei pagini este suma gradelor de autoritate ale tuturor paginilor referite, scalată cu  $\lambda$ .
- ◆  $a = \mu A^T h$ . Adică gradul de autoritate al fiecărei pagini este suma gradelor de indexare ale tuturor paginilor care o referă, scalată cu  $\mu$ .

# Indecși si autorități

- ◆ Din ecuațiile (1) și (2) putem deduce folosind substituția, două ecuații care leagă vectorii  $a$  și  $h$  doar de ei înșiși:

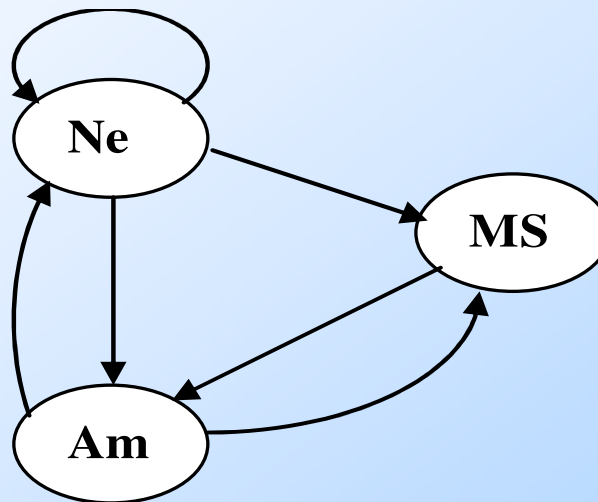
$$a = \lambda \mu A^T A a$$

$$h = \lambda \mu A A^T h$$

- ◆ Ca urmare, putem calcula  $h$  și  $a$  prin relaxare, obținând vectorul principal de valori proprii al matricilor  $A A^T$  și respectiv  $A^T A$

# Exemplul 5

◆ Fie graful urmator:



# Exemplul 5

◆ Matricile sunt:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad A^T = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad AA^T = \begin{bmatrix} 3 & 1 & 2 \\ 0 & 0 & 1 \\ 2 & 0 & 2 \end{bmatrix} \quad A^TA = \begin{bmatrix} 2 & 2 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

◆ Dacă utilizăm  $\lambda = \mu = 1$  și considerăm că vectorii  $h = [h_n, h_m, h_a]$  și  $a = [a_n, a_m, a_a]$  sunt inițial fiecare  $[1, 1, 1]$ , primele trei iterații ale ecuațiilor pentru  $a$  și  $h$  sunt:

# Exemplul 5

◆ Seccesiunea de valori pentru  $a$  este:

$a_n$	=	1	5	24	114
$a_m$	=	1	5	24	114
$a_a$	=	1	4	18	84

◆ Iar pentru  $h$ :

$h_n$	=	1	6	28	132
$h_m$	=	1	2	8	36
$h_a$	=	1	4	20	96

## Exemplul 5

- ◆ Vectorul  $a$ , ***scalat corespunzător***, va converge către un vector în care
  - ◆  $a_n = a_m$  și
  - ◆ fiecare dintre aceste numere este mai mare ca  $a_a$  în raportul  $1 + \sqrt{3} / 2$  sau aproximativ 1.36

# Extragerea de cunoștințe din web

## ◆ Puncte importante:

- 1. Numărarea dinamică a mulțimilor de articole:* Căutarea de mulțimi *interesante* de articole într-un spațiu mult prea mare pentru a se putea lua în considerare fiecare pereche de articole.
- 2. "Cărți și autori":* Intrigantul experiment al lui Sergey Brin de extragere de date relaționale din web.

# Numarare dinamica

- ◆ Problema este de a găsi mulțimi de cuvinte care apar "neobișnuit de des" împreună pe web, e.g. "New" și "York" sau {Ducesa, de, York}.
- ◆ "Neobișnuit de des" poate fi definit în diverse moduri pentru a încorpora ideea că numărul de documente web conținând mulțimea de cuvinte este mult mai mare decât cel așteptat în cazul în care cuvintele ar fi fost alese la întâmplare, fiecare cuvânt cu probabilitatea sa de apariție într-un document.



# Numarare dinamica

- ◆ Un mod adecvat este *entropia per cuvânt din mulțime*. Formal, **interesul** unei mulțimi de cuvinte  $S$  este:

$$\frac{\log_2\left(\frac{\text{prob}(S)}{\prod_{w \text{ in } S} \text{prob}(w)}\right)}{|S|}$$

- ◆ De notat că împărțim la dimensiunea lui  $S$  pentru a evita "efectul Bonferroni", în care sunt atât de multe mulțimi de o dimensiune dată încât unele, din motive probabilistice, par a fi corelate.
- ◆ Exemplu: Dacă  $a$ ,  $b$  și  $c$  (cuvinte) apar fiecare în 1% din toate documentele și  $S=\{a, b, c\}$  apar în 0.1% din documente, interesul lui  $S$  este  $(\log_2(0.001/(0.01 \times 0.01 \times 0.01)))/3 = \log_2(1000)/3$  adică aproximativ 3.3.

# Numarare dinamica

- ◆ Problema tehnică: interesul nu este monoton sau "închis în jos" în modul de la produse cu larg suport.
- ◆ Asta înseamnă că putem avea o mulțime  $S$  cu o valoare mare a interesului și totuși unele sau chiar toate submulțimile sale stricte să nu fie interesante.
- ◆ Prin contrast, dacă  $S$  are suport larg, atunci toate submulțimile sale au cel puțin același suport.
- ◆ Observatie: Cu mai mult de 108 cuvinte diferite apărând pe web nu este posibil nici măcar să considerăm toate perechile de cuvinte.

# DICE

- ◆ DICE (dynamic itemset counting engine) vizitează repetat paginile web într-un mod de tip "round-robin".
- ◆ De fiecare dată numără aparițiile anumitor mulțimi de cuvinte și ale fiecărui cuvânt din aceste mulțimi.
- ◆ Numarul de mulțimi numărate este suficient de mic încât contorii lor încap în memoria centrală.

# DICE

- ◆ Din când în când, să spunem la fiecare 5000 de pagini, DICE își reconsideră mulțimile pentru care numără. Înlătură acele mulțimi care au cel mai mic interes și le înlocuiește cu alte mulțimi.
- ◆ Alegerea noilor mulțimi se bazează pe proprietatea numită *heavy edge* care este o observație justificată experimental că acele cuvinte care apar în mulțimi cu interes ridicat au probabilitatea mai mare să apară în alte mulțimi cu interes ridicat.

# DICE

- ◆ Astfel, când selectează noi mulțimi pentru a începe numărarea, DICE este direcționat în favoarea cuvintelor care apar deja în mulțimi cu interes ridicat.
- ◆ Totuși, el nu se bazează exclusiv pe aceste cuvinte altfel nu ar putea niciodată să găsească mulțimi cu interes ridicat compuse din multe cuvinte pe care nu le-a considerat niciodată.
- ◆ Unele (dar nu toate) din construcțiile pe care le utilizează DICE pentru crearea noilor mulțimi sunt:

# DICE: noi multimi

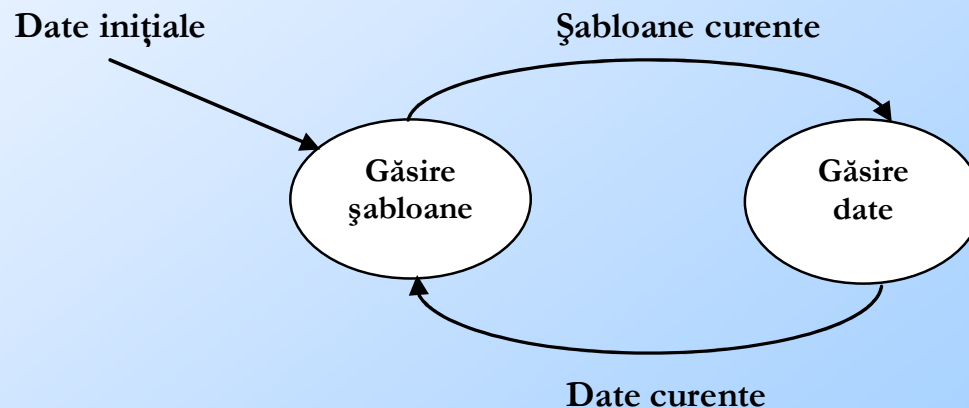
1. Două cuvinte aleatoare. Aceasta este singura regulă independentă de aserțiunea muchiei grele și ajută noi cuvinte să ajungă în mulțime.
2. Un cuvânt dintr-una din mulțimile interesante și un cuvânt aleator.
3. Două cuvinte din două mulțimi interesante diferite.

# DICE: noi mulțimi

- 4. Reuniunea a două mulțimi interesante a căror intersecție are dimensiunea 2 sau mai mult.
- 5.  $\{a, b, c\}$  dacă toate mulțimile  $\{a, b\}$ ,  $\{a, c\}$  și  $\{b, c\}$  sunt găsite ca fiind interesante.
- ◆ Bineînțeles, în general sunt mult prea multe opțiuni de a aplica cele de mai sus în toate modurile posibile astfel încât se utilizează o selecție aleatoare a opțiunilor dând o anumită șansă fiecăreia dintre ele.

# Carti si autori

- ◆ Ideea principală este de a căuta pe web fapte de un anumit tip, de genul celor care ar putea forma o relație de genul *Cărți(titlu, autor)*. Procesarea este sugerată de figura următoare:





# Cum lucreaza

1. Se pornește de la un eșantion al tuplelor care se doresc găsite. În exemplul discutat în lucrarea lui Brin au fost folosite cinci exemple de titluri de cărți și autori ai acestora.
2. Pe baza exemplelor cunoscute, se caută pagini unde apar aceste date pe web. Dacă se găsește un șablon care identifică un număr de tuple cunoscute și este suficient de specific încât e puțin probabil să identifice prea mult, atunci se acceptă acest șablon.

# Cum lucreaza

3. Fiind dată o mulțime de șabloane acceptate, se caută date care satisfac aceste șabloane și se adaugă la mulțimea datelor cunoscute.
4. Se repetă pașii (2) și (3) de un număr de ori. În exemplul citat au fost utilizate patru ciclări care au dus la 15,000 de tuple; aprox. 95% au fost perechi adevărate titlu-autor.

# Ce este un sablon

- ◆ Are 5 componente
  1. *Ordinea*; i.e. daca titlul apare în text înaintea autorului sau vice-versa. Într-un caz general, în care tuplele au mai mult de 2 componente, ordinea va fi dată de permutarea componentelor.
  2. *Prefixul adresei web (URL)*.
  3. *Prefixul* textului, care apare înaintea primului dintre titlu și autor
  4. *Mijlocul*: text care apare între cele două elemente de date.
  5. *Sufixul* textului care urmează după al doilea dintre cele două elemente de date. Atât prefixul cât și sufixul au fost limitate la 10 caractere.

# Exemplu

◆ Un șablon posibil poate consta din următoarele:

1. **Ordinea:** titlul și apoi autorul.
2. **Prefixul URL:** [www.stanford.edu/class](http://www.stanford.edu/class)
3. **Prefixul**, mijlocul și sufixul de forma următoare:

<LI><I>titlu</I> de autor<P>

Aici **prefixul** este <LI><I>, **mijlocul** este </I> de (inclusiv spațiul după "de") și **sufixul** este <P>. Titlul este orice apare între prefix și mijloc; autorul este orice apare între mijloc și sufix.

# Tuning sablon

- ◆ Definim ***specificitatea*** unui șablon ca fiind produsul lungimilor prefixului, mijlocului, sufixului și prefixului URL. În mare, specificitatea măsoară cât de posibil este să găsim date care corespund șablonului; cu cât specificitatea este mai mare, cu atât ne așteptăm la mai puține apariții ale acestuia în date.
- ◆ Apoi șablonul trebuie să îndeplinească două condiții pentru a fi acceptat:
  1. Trebuie să fie cel puțin 2 elemente de date cunoscute care apar conform aceluși șablon.
  2. Produsul specificității șablonului cu numărul de apariții de date conform acestuia trebuie să depășească un anumit prag  $T$  (nespecificat).

# Pasii executiei

1. Găsirea aparițiilor pornind de la datele cunoscute
2. Construcția șabloanelor din aparițiile de date
3. Găsirea aparițiilor pornind de la șabloane

# Aparitie

- ◆ O apariție a unui tuplu este asociată cu un șablon după care acestea apar; i.e., același titlu și autor pot să apară după diferite șabloane. Astfel, o apariție a datelor constă în:
  1. Un anumit titlu și autor.
  2. Adresa Internet completa (URL) și nu doar prefixul ca în cazul șablonului.
  3. Ordinea, prefixul, mijlocul și sufixul șablonului după care au apărut titlul și autorul respectiv.

# Constructia sabloanelor

1. Se grupează aparițiile de date după ordinea și mijlocul lor. De exemplu, un grup din acest "group-by" poate corespunde ordinii "titlu-apoi-autor" și mijlocului "</I> de".
2. Pentru fiecare grup se găsește cel mai lung prefix, sufix și prefix URL comun.
3. Dacă testul de specificitate pentru acest șablon este îndeplinit, se acceptă șablonul.



# Constructia sabloanelor

4. Dacă testul de specificitate *nu* este îndeplinit, se încearcă spargerea grupului în doua prin extinderea lungimii prefixului URL cu un caracter și apoi se repetă pasul (2). Dacă este imposibil să spargem grupul (pentru că există doar un URL) atunci am eșuat în a produce un nou șablon din acel grup.
- ◆ **Exemplu:** Să presupunem că grupul conține trei URL-uri:
- ◆ [www.stanford.edu/class/cs345/index.html](http://www.stanford.edu/class/cs345/index.html)
  - ◆ [www.stanford.edu/class/cs145/index.html](http://www.stanford.edu/class/cs145/index.html)
  - ◆ [www.stanford.edu/class/cs340/readings.html](http://www.stanford.edu/class/cs340/readings.html)

# Constructia sabloanelor

- ◆ Prefixul comun este [www.stanford.edu/class/cs](http://www.stanford.edu/class/cs).
- ◆ Dacă trebuie să spargem grupul, atunci următorul caracter, 3 sau 1, sparge grupul în două, cu acele apariții ale datelor din prima pagină (pot fi multe astfel de apariții) mergând într-un prim grup și aparițiile din celelalte două pagini în celălalt:
  - ◆ [www.stanford.edu/class/cs345/index.html](http://www.stanford.edu/class/cs345/index.html)
  - ◆ [www.stanford.edu/class/cs340/readings.html](http://www.stanford.edu/class/cs340/readings.html)
- Si
- ◆ [www.stanford.edu/class/cs145/index.html](http://www.stanford.edu/class/cs145/index.html)

# Gasire aparitii din sabloane

1. Se găsesc toate URL-urile care se potrivesc cu prefixul URL al cel puțin unui șablon.
2. Pentru fiecare astfel de pagină se parcurge textul folosind o expresie regulată construită din prefixul, mijlocul și sufixul șablonului.
3. Se extrage din fiecare potrivire titlul și autorul, după ordinea specificată în șablon.

# Bibliografie

- ◆ J.D.Ullman - CS345 --- Lecture Notes:  
PageRank, Hubs-and-Authorities , Web Mining

<http://infolab.stanford.edu/~ullman/cs345-notes.html>

# Sfârșitul capitolului 10