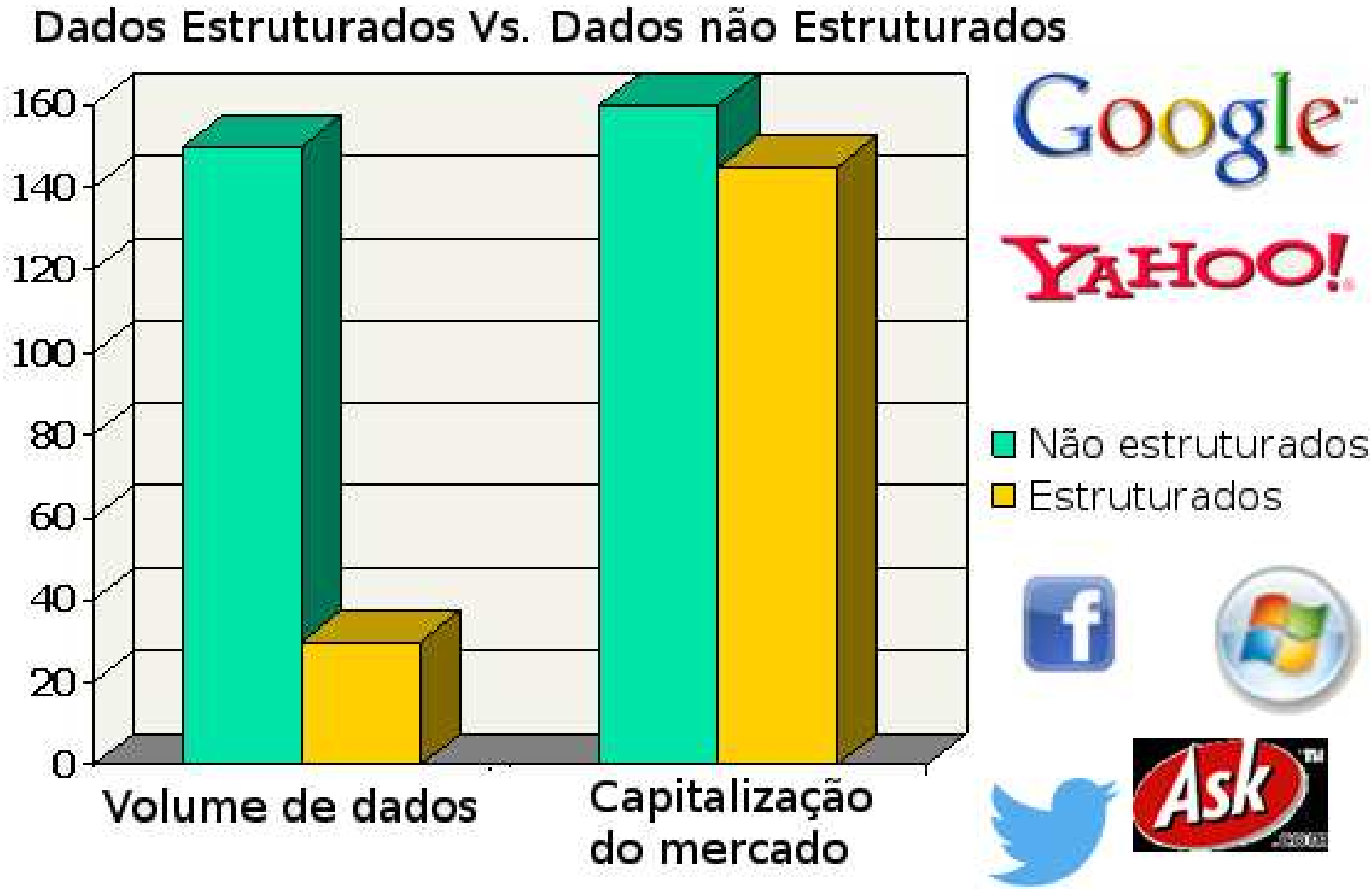
Em 2006:



Em 2012: 2.8 ZB (2.8 bilhões de terabytes) de dados (Fonte: IDC)

Criar listas de índices, determinar frequência de documentos

termo	docID
ambicioso	2
assim	2
brutus	1
brutus	2
capitólio	1
com	2
césar	1
césar	2
césar	2
disse	2
encenei	1
era	2
eu	1
eu	1
fui	1
júlio	1
matou	1
me	1
morto	1
no	1
nobre	2
o	2
que	2
que	2
seja	2
vos	2

$\Longrightarrow$

termo	doc.freq	→	listas de índices
ambicioso	1	→	2
assim	1	→	2
brutus	2	→	1 → 2
capitólio	1	→	1
césar	2	→	1 → 2
disse	1	→	2
encenei	1	→	1
era	1	→	2
eu	2	→	1 → 2
fui	1	→	1
júlio	1	→	1
matou	1	→	1
me	1	→	1
morto	1	→	1
no	1	→	1
nobre	1	→	2
o	1	→	2
que	1	→	2
seja	1	→	2
vos	1	→	2

Algoritmo otimizado para intesecção de consultas conjuntivas

```
INTERSECÇÃO(⟨t_1, ..., t_n⟩)
 1  termos ← ORDENARPORFREQUENCIACRESCENTE(⟨t_1, ..., t_n⟩)
 2  resultado ← referencias(primeiro(termos))
 3  termos ← resto(termos)
 4  while termos ≠ NIL and resultado ≠ NIL
 5  do
 6      lista ← REFERENCIAS(PRIMEIRO(termos))
 7      resultado ← INTERSECÇÃO(resultado, lista)
 8      termos ← resto(termos)
 9  return resultado
```

Como fazer quando existe `OR` ? Como fazer quando existe `NOT` ?

Japonês

ノーベル平和賞を受賞したワンガリ・マータイさんが名誉会長を務めるＭＯＴＴＡＩＮＡＩキャンペーンの一環として、毎日新聞社とマガジンハウスは「私の、もったいない」を募集します。皆様が日ごろ「もったいない」と感じて実践していることや、それにまつわるエピソードを８００字以内の文章にまとめ、簡単な写真、イラスト、図などを添えて１０月２０日までにお送りください。大賞受賞者には、５０万円相当の旅行券とエコ製品２点の副賞が贈られます。

4 “alfabetos” diferentes: caracteres chineses; sílabas hiragana para sufixos de inflexão e palavras de função; sílabas do katakana para transcrição de palavras estrangeiras e outros usos; e caracteres romanos. Sem espaços.

Stop words (palavras de parada)

- stop words = palavras extremamente comuns e de pouco valor para selecionar documentos para o usuário
- exemplos: *a*, *an*, *and*, *are*, *as*, *at*, *be*, *by*, *for*, *from*, *has*, *he*, *in*, *is*, *it*, *its*, *of*, *on*, *that*, *the*, *to*, *was*, *were*, *will*, *com*
- *um*, *uns*, *o*, *a*, *os*, *as*, *que*, *se*, *para*, *de*, *sobre*, . . .
- Eliminação de *stop words* costumava ser padrão em ORI antigos
- Mas são necessárias em consultas como: e.g. "Rei da Espanha"

Três stemmers: comparação

Amostra de texto: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Stemmer de Porter: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

Stemmer de Lovins: such an analys can reve featur that ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog transpar and acces to interpres

Stemmer de Paice: such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

Exemplo: índices posicionais

Consulta: *"to*$_1$ *be*$_2$*"*

TO, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

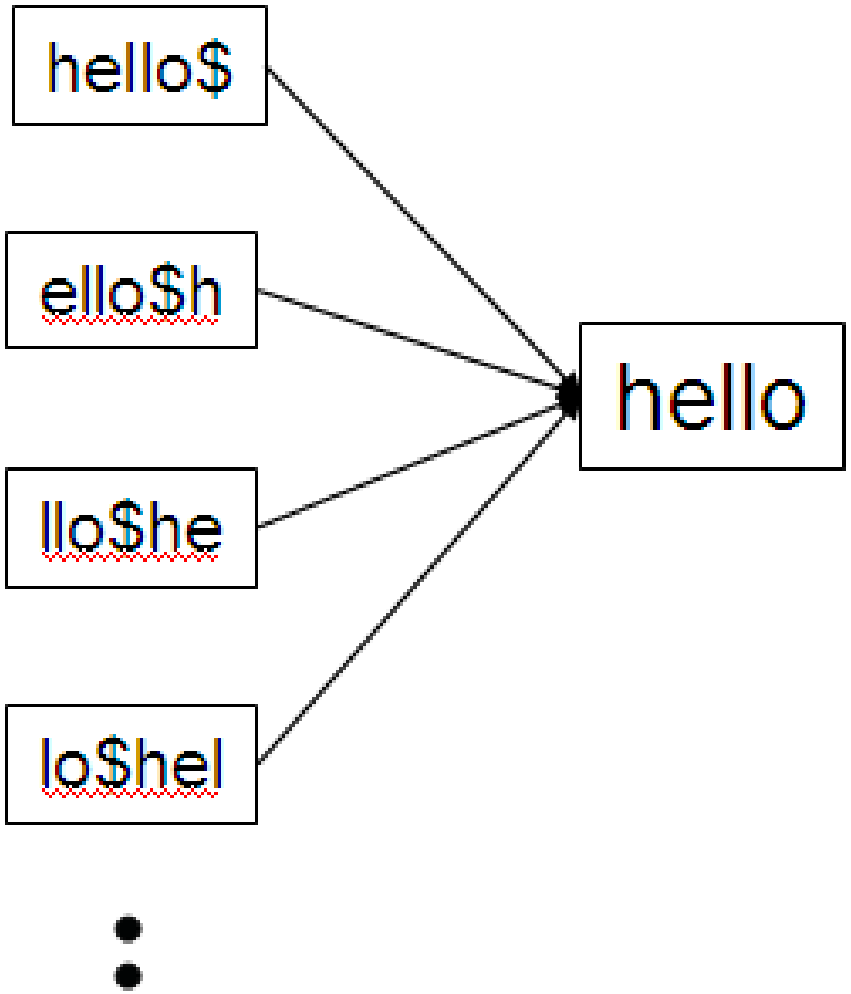
BE, 178239:

⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Intersecção de "proximidade"

```
INTERSECCAOPOSICIONAL(p1, p2, k)
 1  resultado ← ⟨ ⟩
 2  while p1 ≠ NIL e p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then l ← ⟨ ⟩
 5             pp1 ← posicoes(p1)
 6             pp2 ← posicoes(p2)
 7             while pp1 ≠ NIL
 8             do while pp2 ≠ NIL
 9                do if |pos(pp1) − pos(pp2)| ≤ k
10                      then ADICIONAR(l, pos(pp2))
11                      else if pos(pp2) > pos(pp1)
12                              then interromper
13                   pp2 ← proximo(pp2)
14                while l ≠ ⟨ ⟩ e |l[0] − pos(pp1)| > k
15                do REMOVE(l[0])
16                for each ps ∈ l
17                do ADICIONAR(resultado, ⟨docID(p1), pos(pp1), ps⟩)
18                pp1 ← proximo(pp1)
19             p1 ← proximo(p1)
20             p2 ← proximo(p2)
21        else if docID(p1) < docID(p2)
22                then p1 ← proximo(p1)
23                else p2 ← proximo(p2)
24  return resposta
```

Permuterm → termo mapeado



Distância de Levenshtein

DISTANCIALEVENSHTEIN(s_1, s_2)

```
 1  for i ← 0 to |s1|
 2  do m[i, 0] = i
 3  for j ← 0 to |s2|
 4  do m[0, j] = j
 5  for i ← 1 to |s1|
 6  do for j ← 1 to |s2|
 7       do if s1[i] = s2[j]
 8            then m[i, j] = min{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]}
 9            else  m[i, j] = min{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]+1}
10  return m[|s1|, |s2|]
```

operações: inserção (custo 1), remoção (custo 1), substituição (custo 1), cópia (custo 0)

		c	a	t	c	a	t
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**	**5** **5**	**6** **6**
c	**1** **1**	**0** 2 2 **0**	2 3 **1** **1**	3 4 **2** **2**	**3** 5 **3** **3**	5 6 **4** **4**	6 7 **5** **5**
a	**2** **2**	2 **1** 3 **1**	**0** 2 2 **0**	2 3 **1** **1**	3 4 **2** **2**	**3** 5 **3** **3**	5 6 **4** **4**
t	**3** **3**	3 **2** 4 **2**	2 **1** 3 **1**	**0** 2 2 **0**	2 3 **1** **1**	3 4 **2** **2**	**3** 5 **3** **3**

custo	operação	entrada	saída
0	(cópia)	c	c
0	(cópia)	a	a
0	(cópia)	t	t
1	inserção	*	c
1	inserção	*	a
1	inserção	*	t

Exemplo: Soundex de *HERMAN*

- Reter H
- *ERMAN* $\rightarrow$ *0RM0N*
- *0RM0N* $\rightarrow$ *06505*
- *06505* $\rightarrow$ *06505*
- *06505* $\rightarrow$ *655*
- Retornar *H655*
- Note: *HERMANN* gerará o mesmo código

Indexação baseada em blocos ordenados

Indexação baseada em blocos ordenados = Blocked Sort-Based Indexing (BSBI)

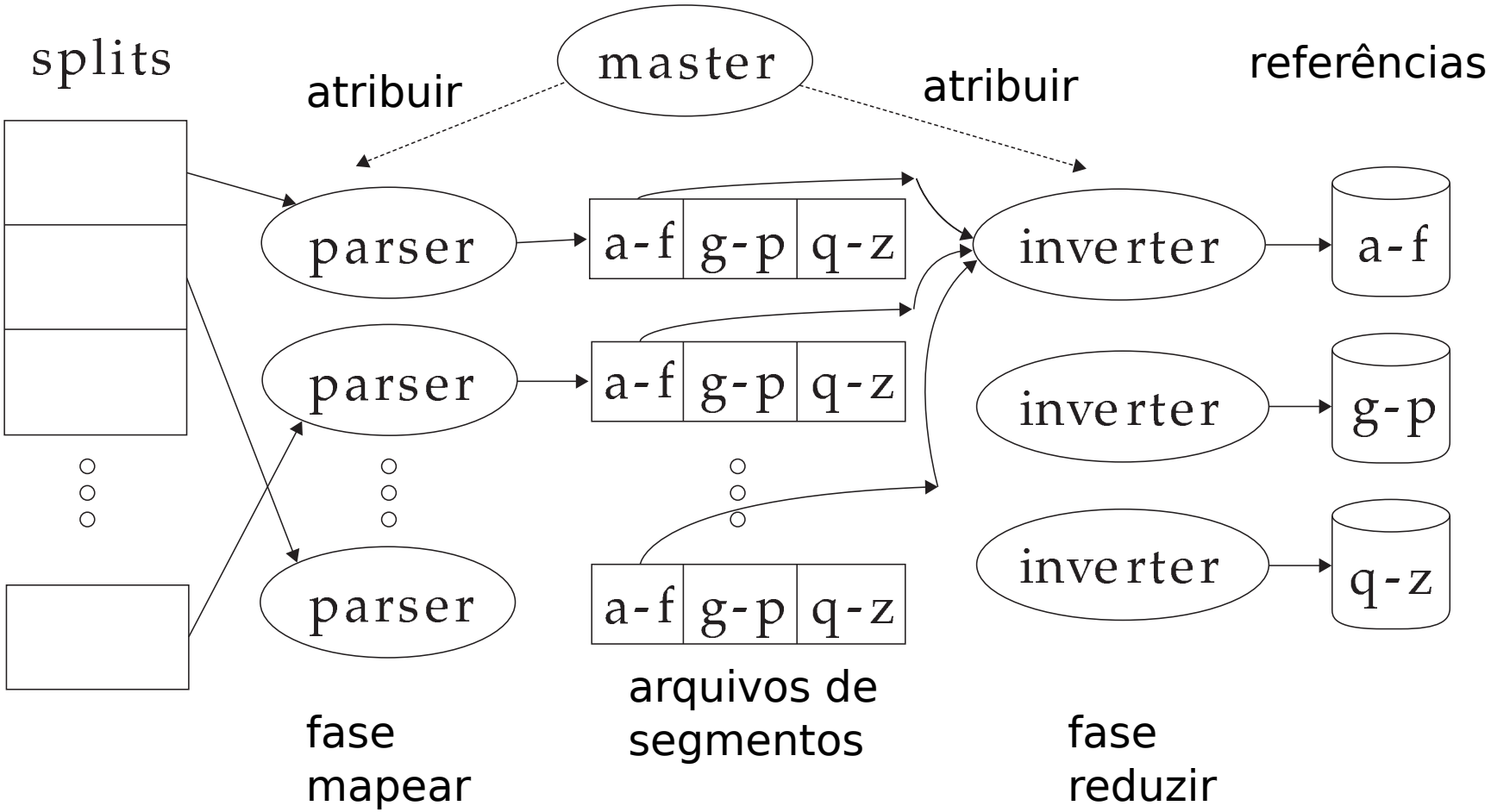
BSBIndexConstruction()
1 $n \leftarrow 0$
2 **while** (todos documentos não foram processados)
3 **do** $n \leftarrow n + 1$
4 $bloco \leftarrow$ ProcessarProximoBloco()
5 BSBI-Invert($bloco$)
6 EscreverBlocoParaDisco($block, f_n$)
7 Merge-Blocos($f_1, \ldots, f_n; f_{\text{juntos}}$)

SPIMI-Invert

```
SPIMI-Invert(fluxo_de_tokens)
 1  arquivo_saida ← NewFile()
 2  dicionario ← NewHash()
 3  while (memória livre disponivel)
 4  do token ← next(fluxo_de_tokens)
 5      if termo(token) ∉ dicionario
 6        then lista_de_referencias ← AddToDictionary(dicionario, termo(token))
 7        else lista_de_referencias ← GetPostingsList(dicionario, termo(token))
 8      if esta_cheia(lista_de_referencias)
 9        then lista_de_referencias ← DoublePostingsList(dicionario, termo(token))
10      AddToPostingsList(lista_de_referencias, docID(token))
11  termos_ordenados ← OrdenarTermos(dicionario)
12  EscreverBlocoEmDisco(termos_ordenados, dicionario, arquivo_saida)
13  return arquivo_saida
```

`GetPostingsList` = pegar lista de referências de um termo associado a um token no dicionário

Caminho dos dados



Problemas com índices auxiliar e principal

- Junções frequentes
- Desempenho de busca ruim durante junção de índice
- Na verdade:
  - Junção de índice auxiliar no índice principal não é muito custoso se mantermos um arquivo separado para cada lista de referências
  - Junção é o mesmo que um simples inserção no arquivo
  - Mas teríamos muitos arquivos – ineficiente
- Na realidade, usar um esquema entre as duas situações: por exemplo, dividir listas de referência muito grandes em vários arquivos, juntas pequenas listas de referências em um arquivo só

Peso tf-idf

- O peso tf-idf de um termo é o **produto de peso tf e seu peso idf**.
- $$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$
- peso tf
- peso idf
- Esquema bastante conhecido em RI.
- Outros nomes: tf.idf, tf x idf

Similaridade coseno entre consulta e documento

$$\cos(\vec{q},\vec{d}) = \text{SIM}(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

- q_i é o peso tf-idf do termo i na consulta.
- d_i é o peso tf-idf do termo i no documento.
- $|\vec{q}|$ e $|\vec{d}|$ são as magnitudes de $\vec{q}$ e $\vec{d}$.
- Esta é a similaridade **coseno** entre $\vec{q}$ e $\vec{d}$ . . . . . ou, de maneira equivalente, o coseno do ângulo entre $\vec{q}$ e $\vec{d}$.

Computando a pontuação cosseno

```
PontuacaoCosseno(q)
 1  float Pontuacao[N] = 0 // pontuacao de cada documento
 2  float Tamanho[N] // tamanho de cada documento
 3  for each termo consulta t
 4  do calcular w_{t,q} e obter lista de referências para t
 5      for each par(d, tf_{t,d}) na lista de referências
 6      do Pontuacao[d] += w_{t,d} × w_{t,q}
 7  for each d
 8  do // normalização
 9      Pontuacao[d] = Pontuacao[d]/Tamanho[d]
10  return Top K componentes da Pontuacao[]
```

Rapidly scanning the results

Note scan pattern:

Page 3:
Result 1
Result 2
Result 3
Result 4
Result 3
Result 2
Result 4
Result 5
Result 6 <click>

Q: Why do this?

A: What's learned later influences judgment of earlier content.

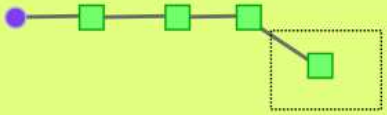
Google

Kinds of behaviors we see in the data

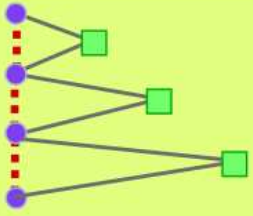
Short / Nav

Topic exploration

Topic switch

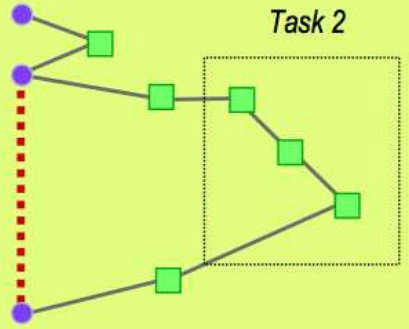


Methodical results exploration

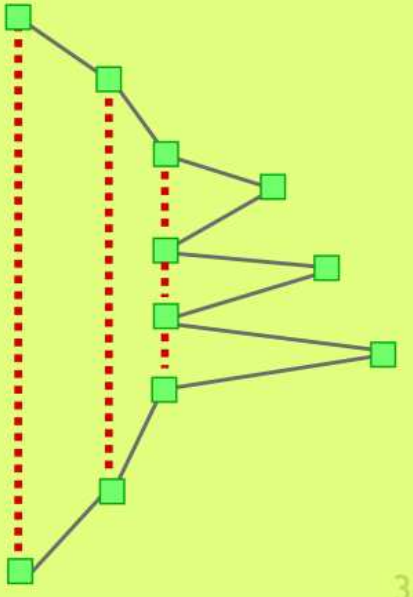


Query reform

Multitasking

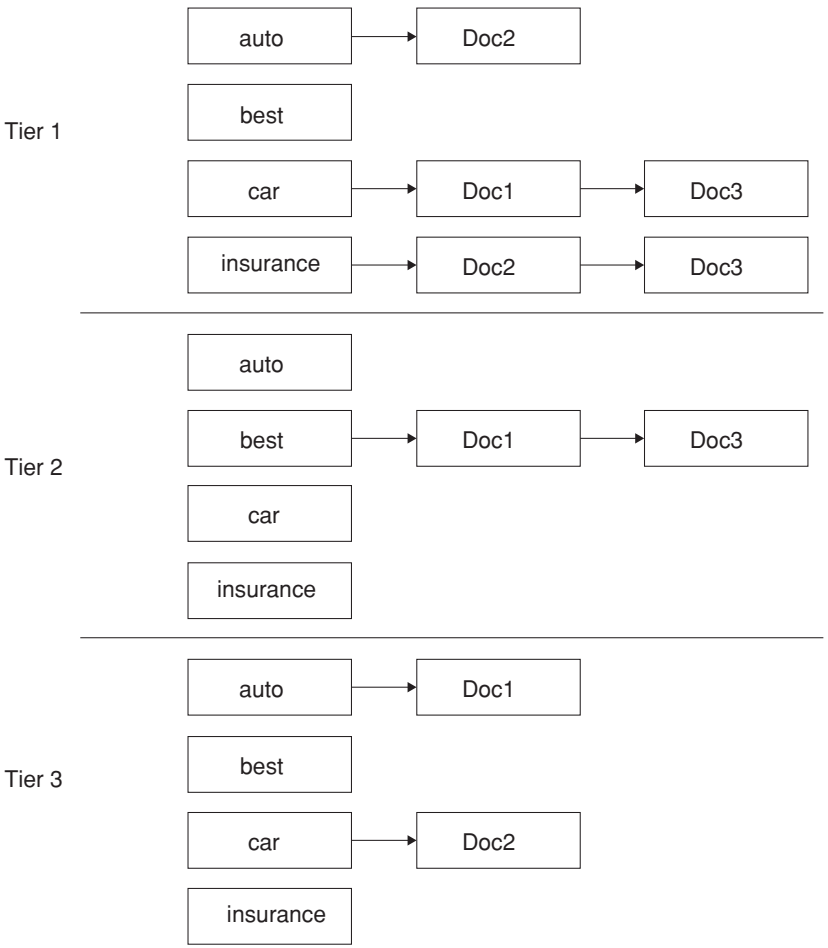


Stacking behavior

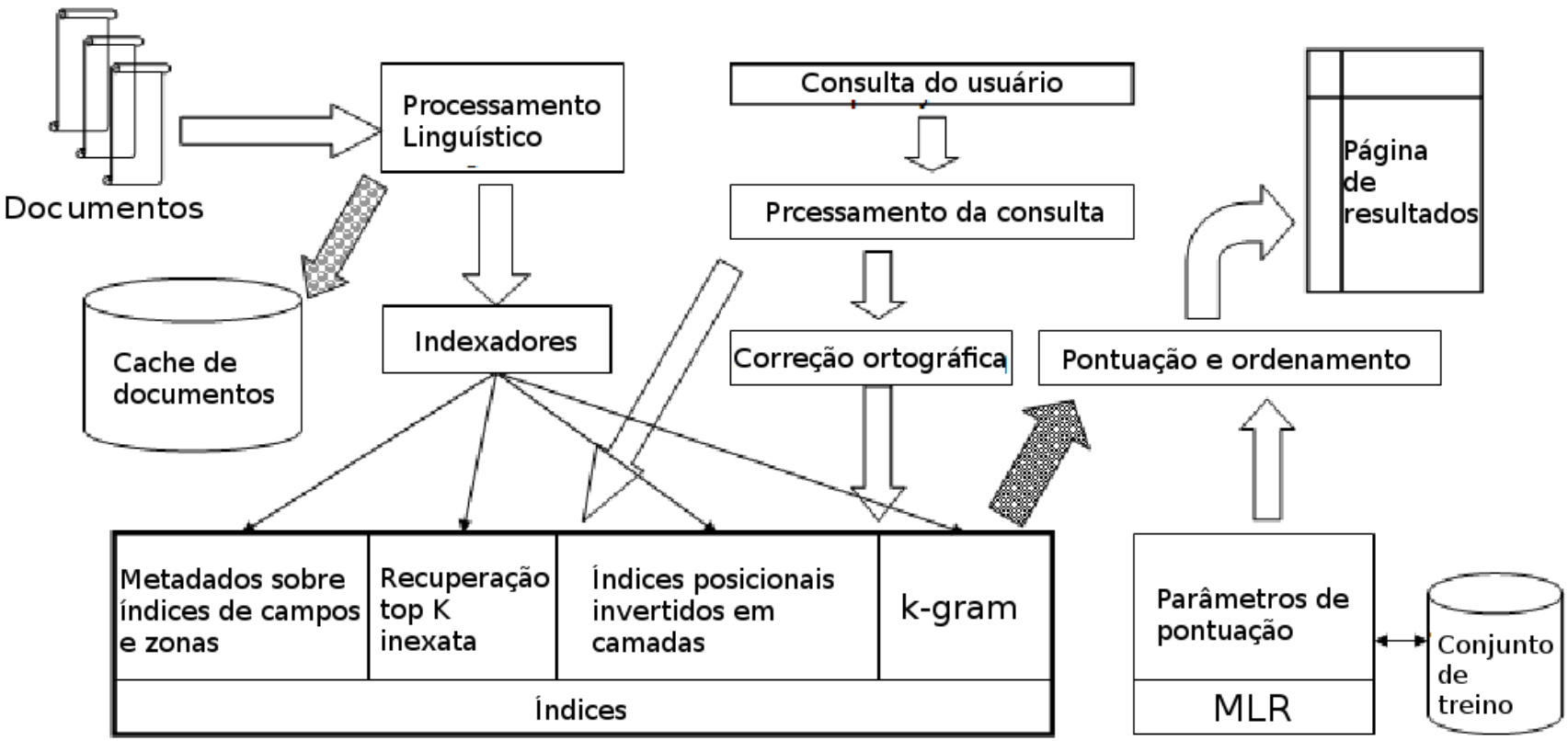


Google

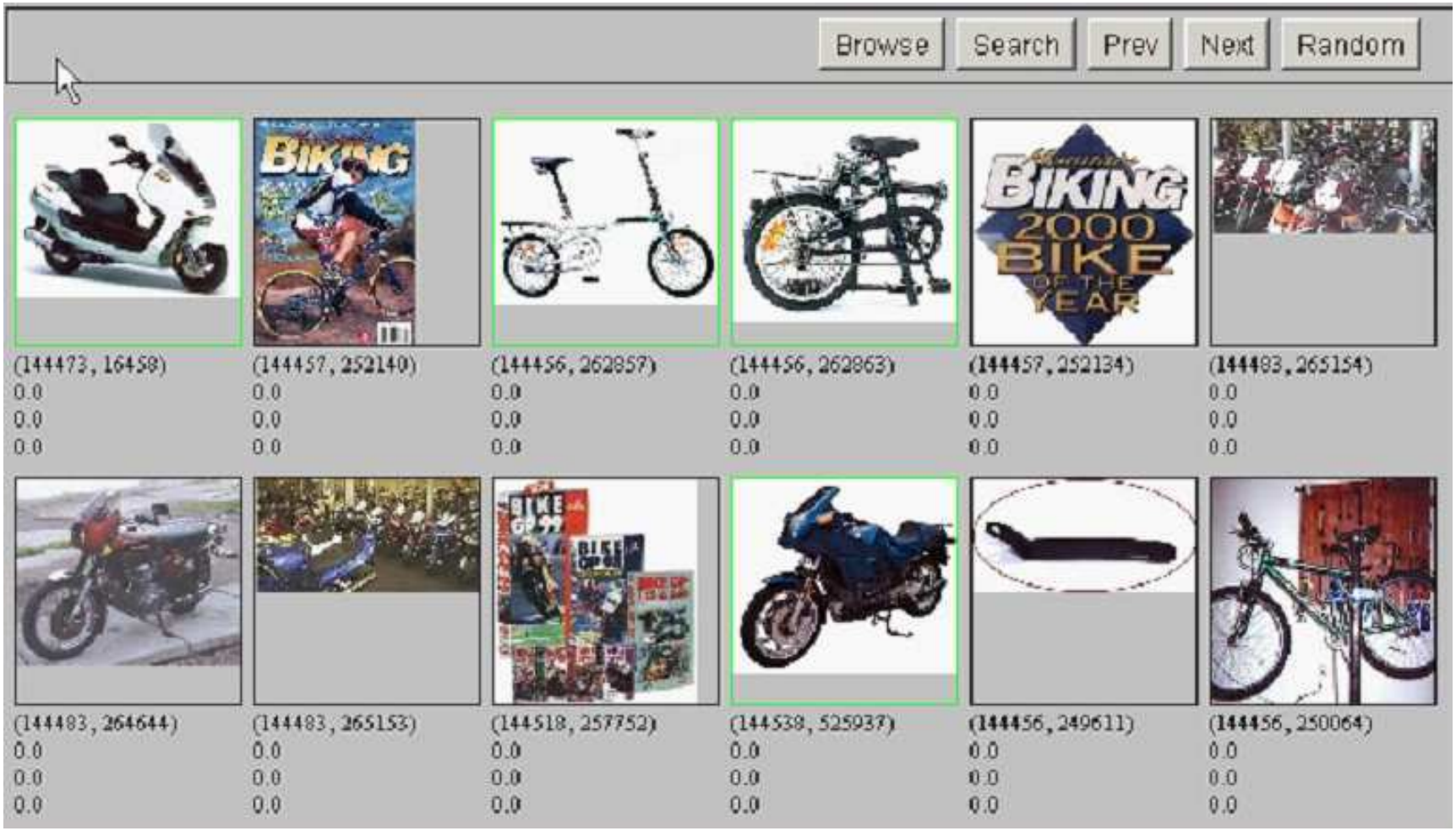
Índices em camadas



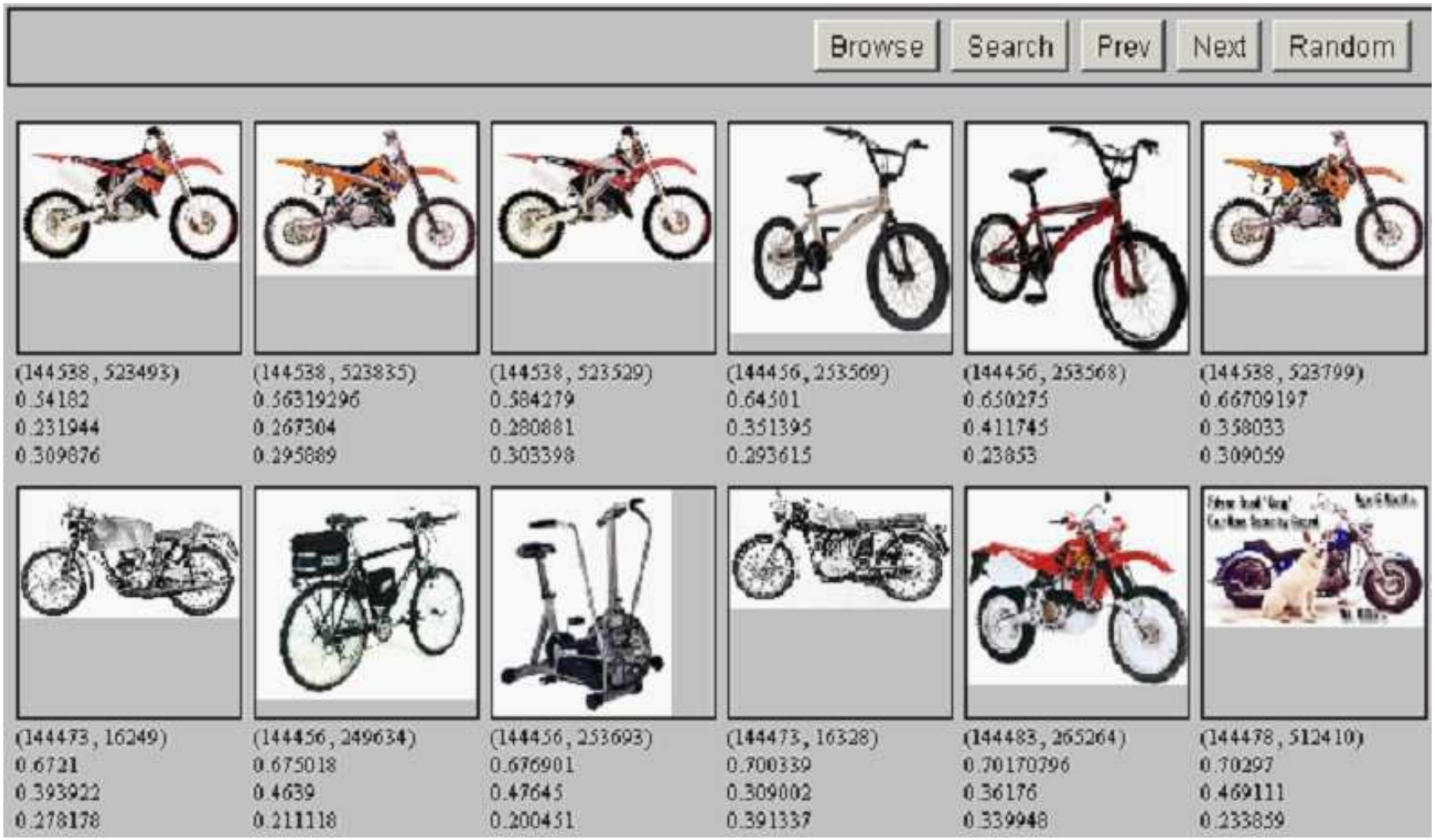
Um sistema de busca completo



Retorno do usuário: selecionar o que é relevante



Resultados após retorno de relevância



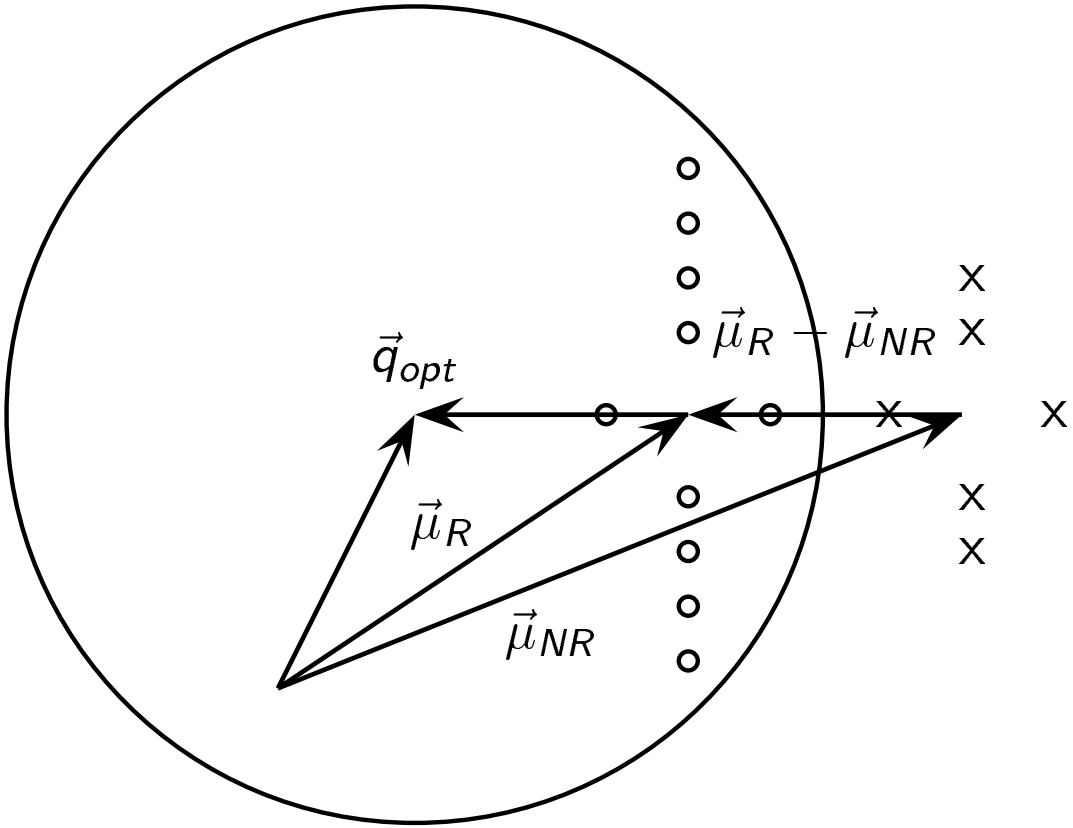
Algoritmo de Rocchio

- O vetor ótimo de consulta é:

$$\begin{aligned}\vec{q}_{opt} &= \mu(D_r) + [\mu(D_r) - \mu(D_{nr})] \\ &= \frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r} \vec{d}_j + [\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r} \vec{d}_j - \frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}} \vec{d}_j]\end{aligned}$$

- Movemos o centroide dos documentos relevantes pela diferença entre os dois centroides.

Rocchio



$\vec{q}_{opt}$ separa relevantes/não-relevantes perfeitamente.

Tesauro baseado em co-ocorrência: exemplos

Palavras	Vizinhos mais próximos
totalmente	completamente, irremediavelmente
mediação	reconciliação, negociação, diplomacia
litografias	desenhos, Picasso, Dali, Gauguin
patógenos	toxinas, bacterias, organismos, parasitas
alface	hortaliças, acelga, hortelã
nada	tudo

Expansão de consultas em motores de busca

- Fonte principal de expansão de consultas em motores de busca: históricos de consultas
- Exemplo 1: Após fazer a consulta [ervas], usuários frequentemente buscam por [ervas medicinais].
  - $\rightarrow$ "ervas medicionais" é uma potencial expansão de "erva".
- Exemplo 2: usuários procurando por [foto de flor] frequentemente clicam no URL photobucket.com/flower. usuários procurando por [desenho de flor] frequentemente clicam no mesmo URL.
  - $\rightarrow$ "desenho de flor" e "foto de flor" são potenciais expansões entre si
  - Mineração de sequências de cliques.

Derivando uma função de ranking para termos da consulta

Equivalente: rankear documentos usando log das taxas de chances para os termos na consulta c_t:

$$c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \log \frac{p_t}{(1-p_t)} - \log \frac{u_t}{1-u_t}$$

- A taxa de chances é a taxa de duas probabilidades: (i) a probabilidade de termos aparecerem se o documento é relevante ($p_t/(1-p_t)$), e (ii) a probabilidade do termo aparecer se o documento é não-relevante ($u_t/(1-u_t)$)
- $c_t = 0$: termo tem iguais chances de aparecer em docs relevantes e não-relevantes
- c_t positivo: chances maiores de aparecer em docs relevantes
- c_t negativo: chances maiores de aparecer em docs não-relevantes

Como estimar probabilidades

Para cada termo t em uma consulta, estimar c_t na coleção completa usando uma tabela de contingência de contagem de documentos na
coleção, onde df_t é o número de documentos que contém o termo t:

	docs	relevante	não-relevante	Total
Termo presente	$x_t = 1$	s	$\mathrm{df}_t - s$	df_t
Termo ausente	$x_t = 0$	$S - s$	$(N - \mathrm{df}_t) - (S - s)$	$N - \mathrm{df}_t$
	Total	S	$N - S$	N

$$p_t = s/S$$

$$u_t = (\mathrm{df}_t - s)/(N - S)$$

$$c_t = K(N, \mathrm{df}_t, S, s) = \log \frac{s/(S-s)}{(\mathrm{df}_t - s)/((N - \mathrm{df}_t) - (S - s))}$$

Pesagem Okapi BM25 para consultas longas

- Para consultas longas, use pesagem similar para consulta de termos

$$RSV_d = \sum_{t \in q} \left[\log \frac{N}{\mathrm{df}_t} \right] \cdot \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1-b) + b \times (L_d/L_{\mathrm{ave}})) + \mathrm{tf}_{td}} \cdot \frac{(k_3 + 1)\mathrm{tf}_{tq}}{k_3 + \mathrm{tf}_{tq}}$$

- tf_{tq}: frequência de termos na consulta q
- k_3: parâmetro de controle da escalonagem da frequência de termos da consulta, $k_3 \geq 0$
- Sem normalização de tamanho de consultas (porque recuperação é feito em relação a uma única consulta fixa)
- A definição de parâmetros deve idealmente ser configurado para otimizar o desempenho em uma coleção de testes para uso durante o desenvolvimento. Na ausência de tal otimização, experimentos têm mostrado resultados razoáveis para k_1 e k_3 estão entre 1.2 e 2 e $b = 0.75$

Minimizar o erro total ϵ: Exemplo (1)

Exemplos de treino

Exemplo	DocID	Consulta	s_T	s_B	Julgamento
Φ_1	37	linux	1	1	1 (relevante)
Φ_2	37	penguin	0	1	0 (não-relevante)
Φ_3	238	system	0	1	1 (relevante)
Φ_4	238	penguin	0	0	0 (não-relevante)
Φ_5	1741	kernel	1	1	1 (relevante)
Φ_6	2094	driver	0	1	1 (relevante)
Φ_7	3194	driver	1	0	0 (não-relevante)

- Computar pontuação:
  $pontuacao(d_j, q_j) = g \cdot s_T(d_j, q_j) + (1 - g) \cdot s_B(d_j, q_j)$
- Computar erro total: $\sum_j \epsilon(g, \Phi_j)$, onde
  $\epsilon(g, \Phi_j) = (r(d_j, q_j) - pontuacao(d_j, q_j))^2$
- Escolher o valor de g que minimiza o erro total

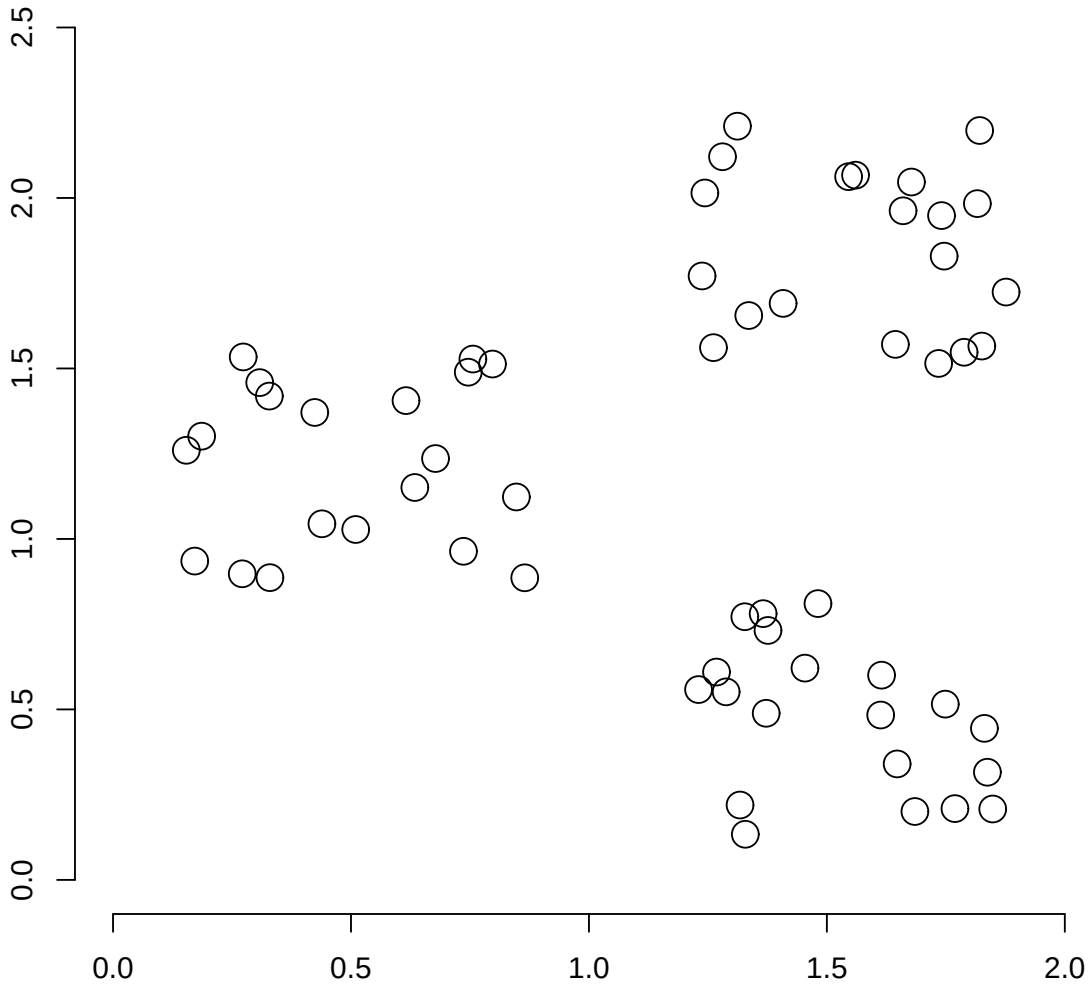
Características usadas pela Microsoft Research (1)

- Campos: corpo, âncora, título, url, doc completo
- Características derivadas de modelos padrões de recuperação de informação: número de termos da consulta, taxa de termos da consulta, idf, soma da frequência de termos, mínimo da frequência de termos, máximos da frequência de termos, média da frequência de termos, variância de frequência de termos, soma da frequência de termos normalizada, ..., soma de tf-idf, modelo booleano, BM25 etc.

Características usadas pela Microsoft Research (2)

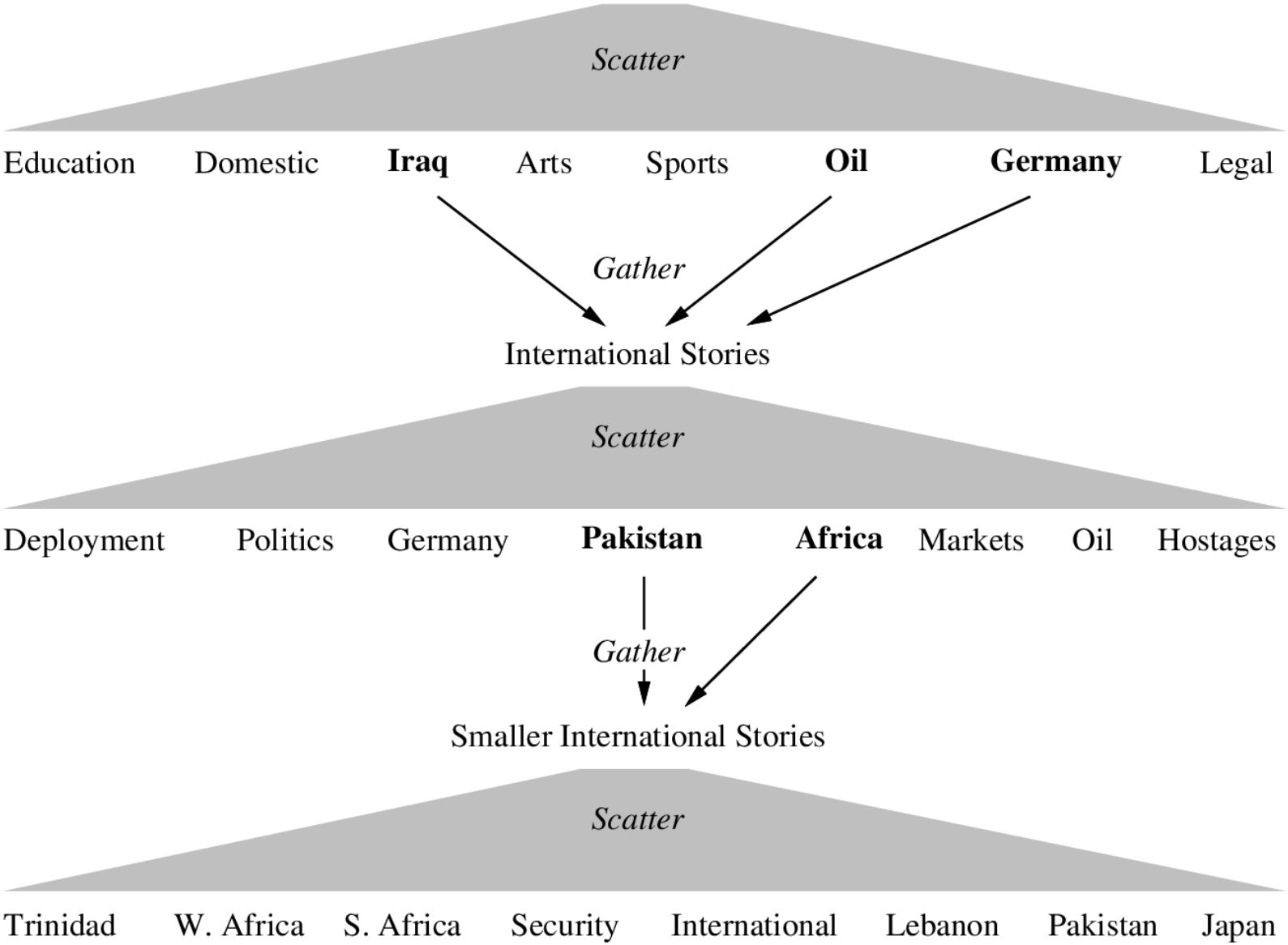
- Características de modelos de linguagem: LMIR.ABS, LMIR.DIR, LMIR.JM
- Características específicas para web: número de barras no url, tamanho do url, número de inlinks, número de outlinks, PageRank, SiteRank
- Características anti-spam: QualityScore
- Características baseadas em histórico: contagem de cliques consulta-url, contagem de cliques da url, tempo na url
- Ver: `http://research.microsoft.com/en-us/projects/mslr/`

Exercício: algoritmo para agrupar dados

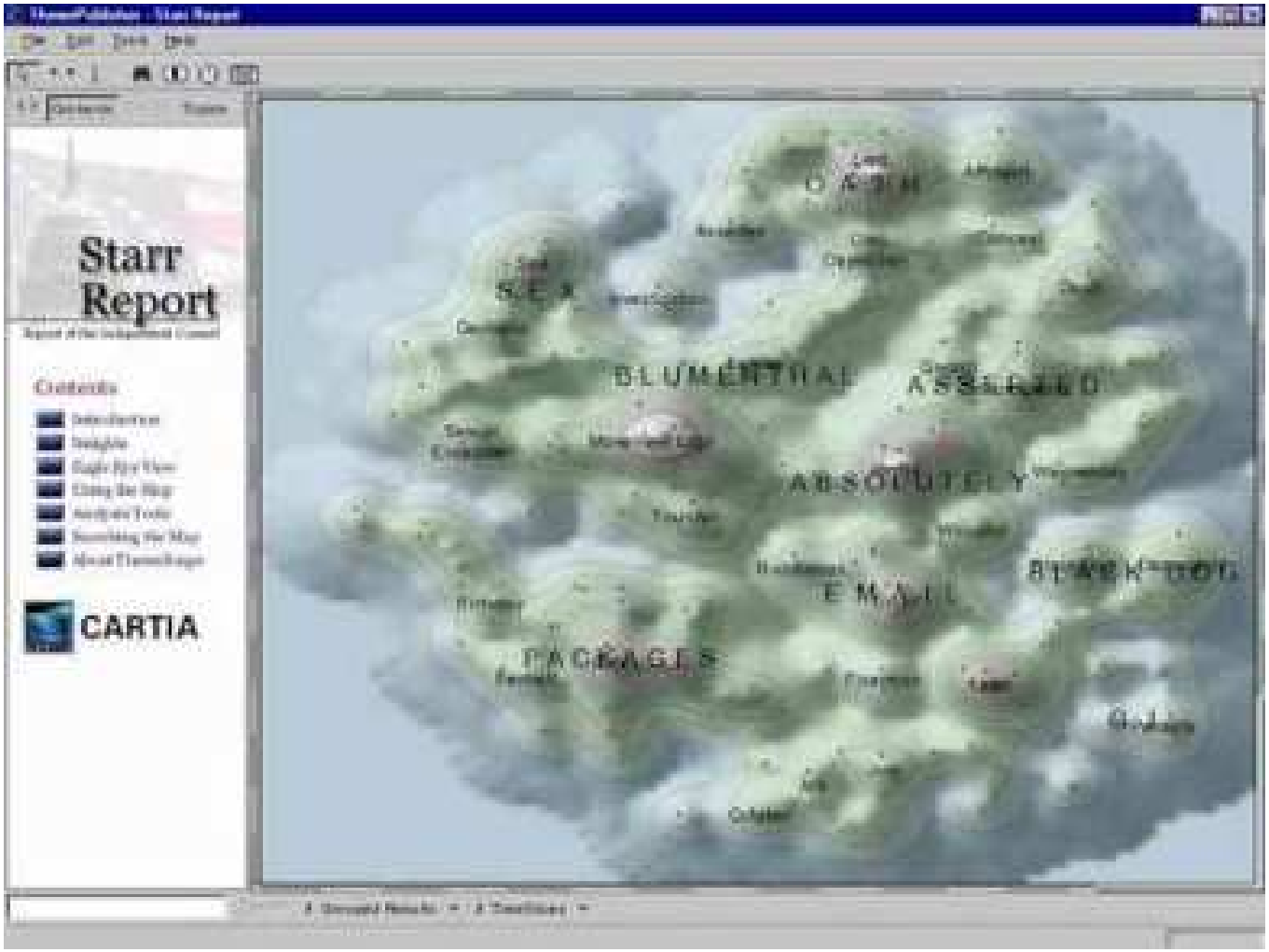


Propor algoritmo para encontrar a estrutura de agrupa-mento nessa figura

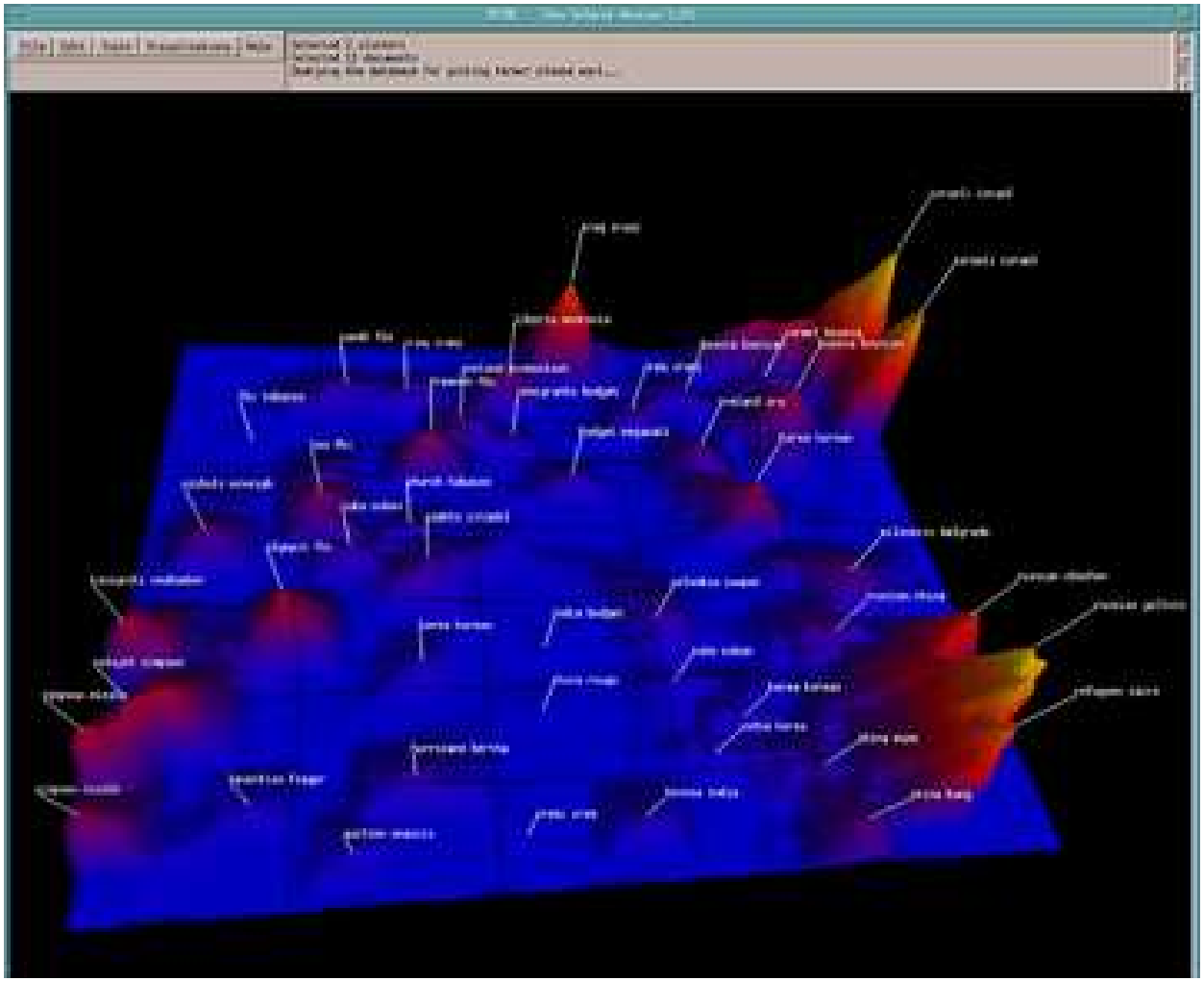
Scatter-Gather



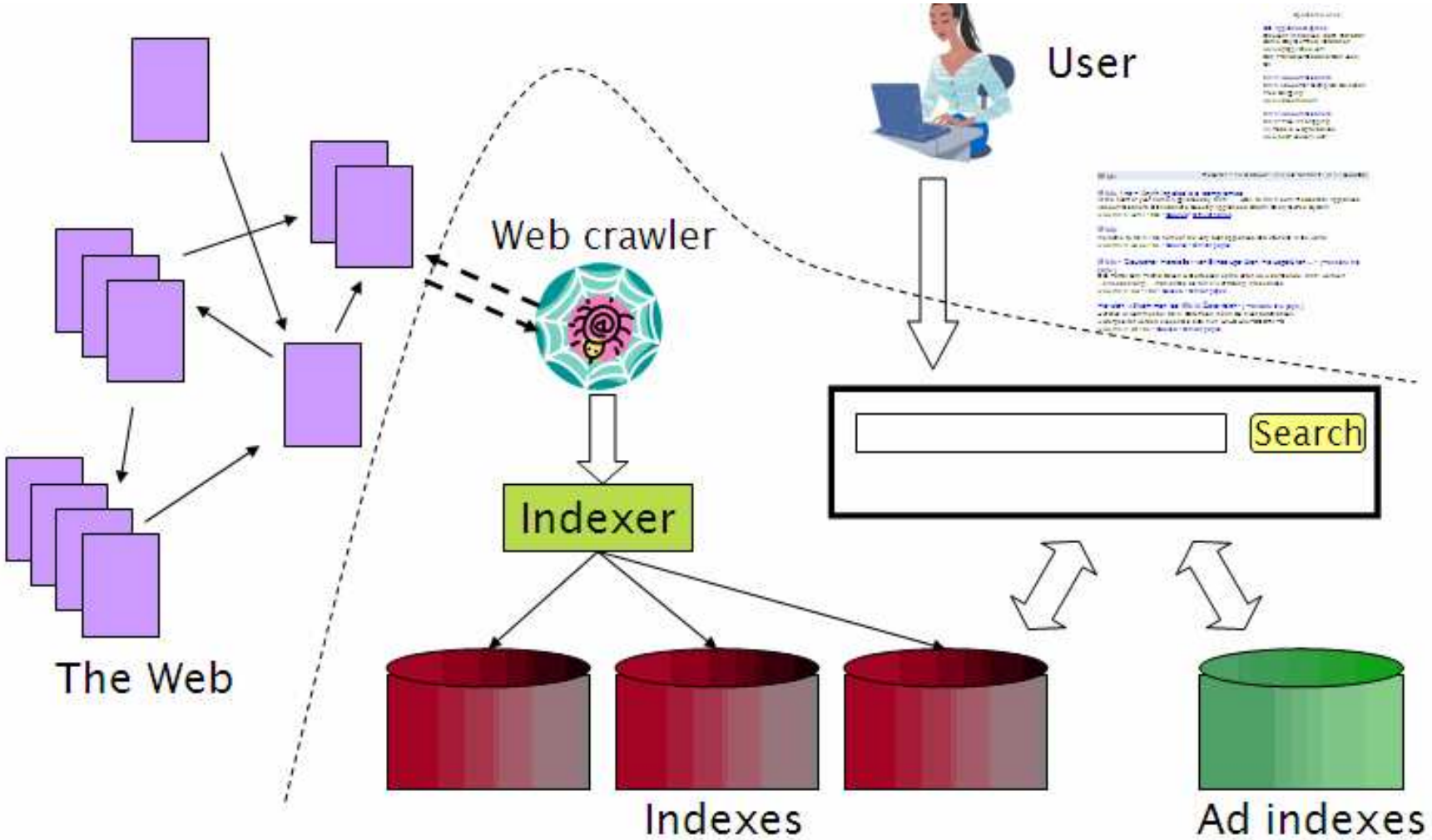
Navegação global combinada com visualização (1)



Navegação global combinada com visualização (2)



Visão geral sobre busca na Web



Primeira geração de anúncios: Goto (1996)

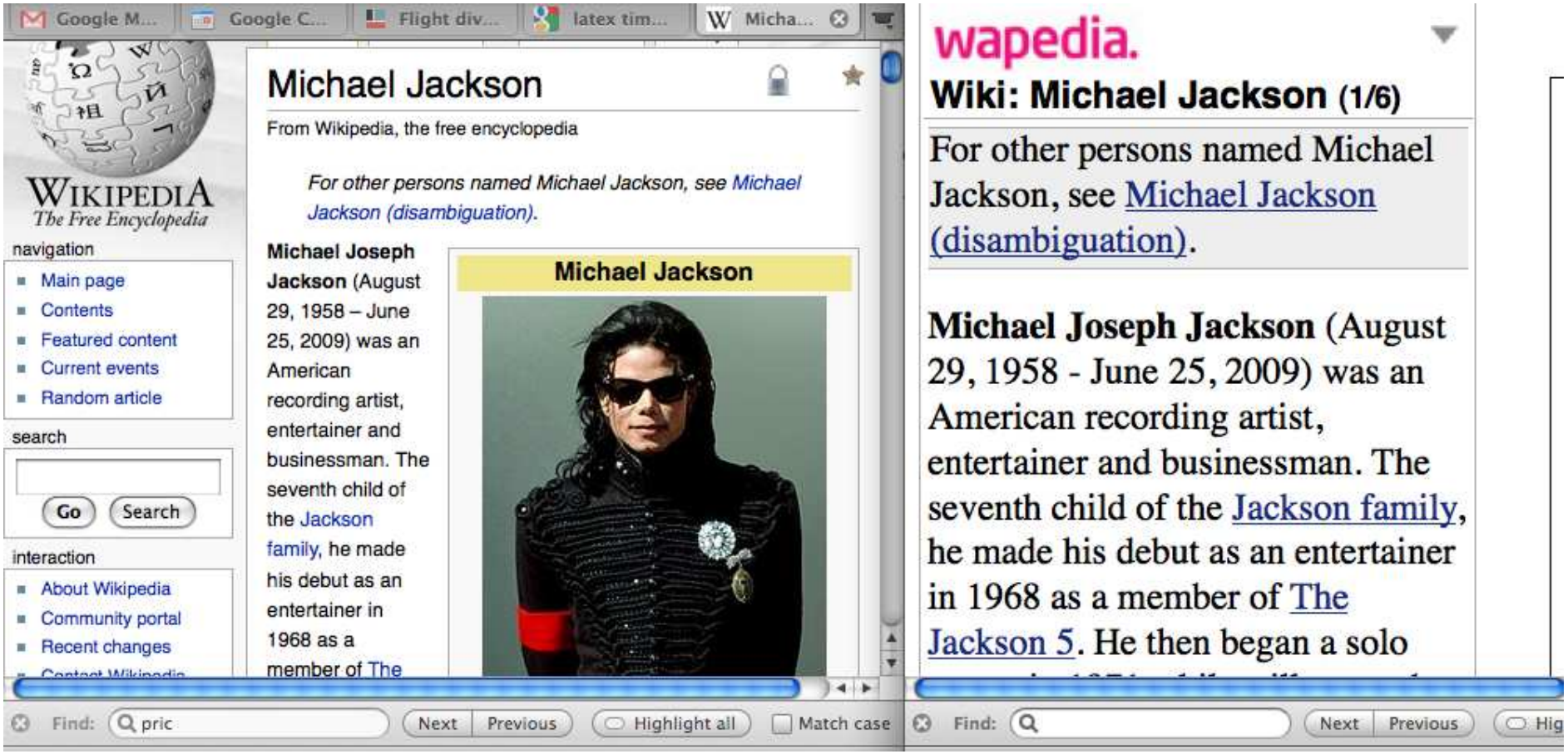
Leilão de segunda posição do Google (simplificado)

anunciante	lance	TDC	valor rank	rank	preço
A	\$4.00	0.01	0.04	4	(mínimo)
B	\$3.00	0.03	0.09	2	\$2.68
C	\$2.00	0.06	0.12	1	\$1.51
D	\$1.00	0.08	0.08	3	\$0.51

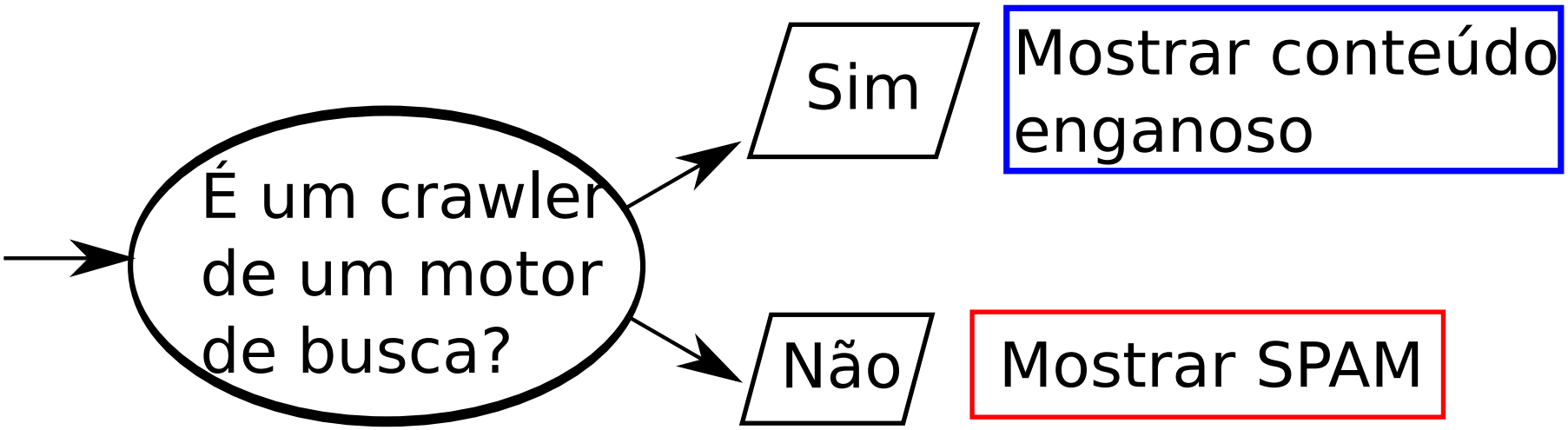
- lance: lance máximo de um clique pelo anunciante
- TDC: taxa de cliques: quando um anúncio é mostrado, qual porcentagem de vezes que usuários clicam nele? TDC é uma medida de relevância.
- rank: rank no leilão
- preço: pago pelo anunciante

$$\text{preço}_{\text{rank}} = \text{lance}_{\text{rank}+1} \frac{\text{TDC}_{\text{rank}+1}}{\text{TDC}_{\text{rank}}}$$

Exemplo: quase-duplicatas

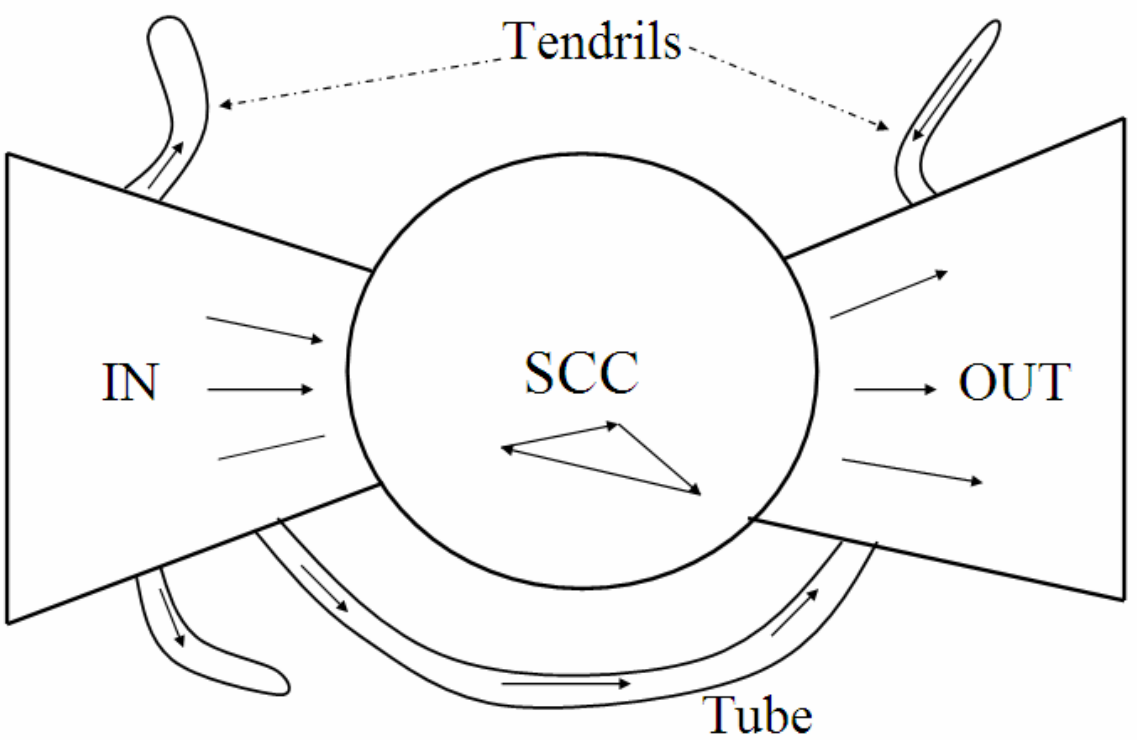


Técnica Spam: ocultação



- Oferecer conteúdo falso para o capturador de páginas do buscador
- Então penalizamos isso sempre?
- Não: usos legítimos (e.g., conteúdos diferentes para EUA vs. Europa)

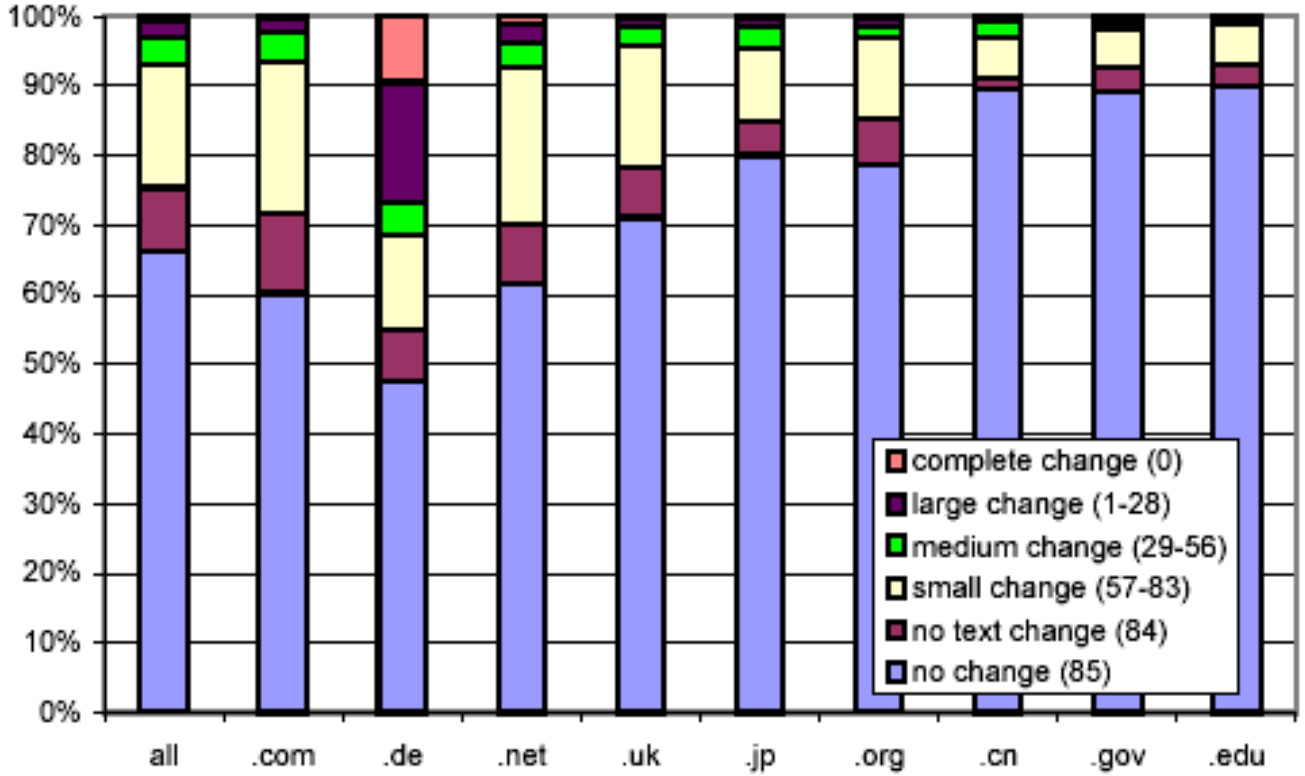
Estrutura de gravata-borboleta da web



- Componente fortemente conectado (SCC) no centro
- Muitas páginas que são linkadas, mas não linkam (OUT)
- Muitas páginas que linkam para outras, mas não recebem links (IN)
- Tendrils (tentáculos). Tubos (caminhos entre regiões IN, OUT, SCC) e ilhas no SCC

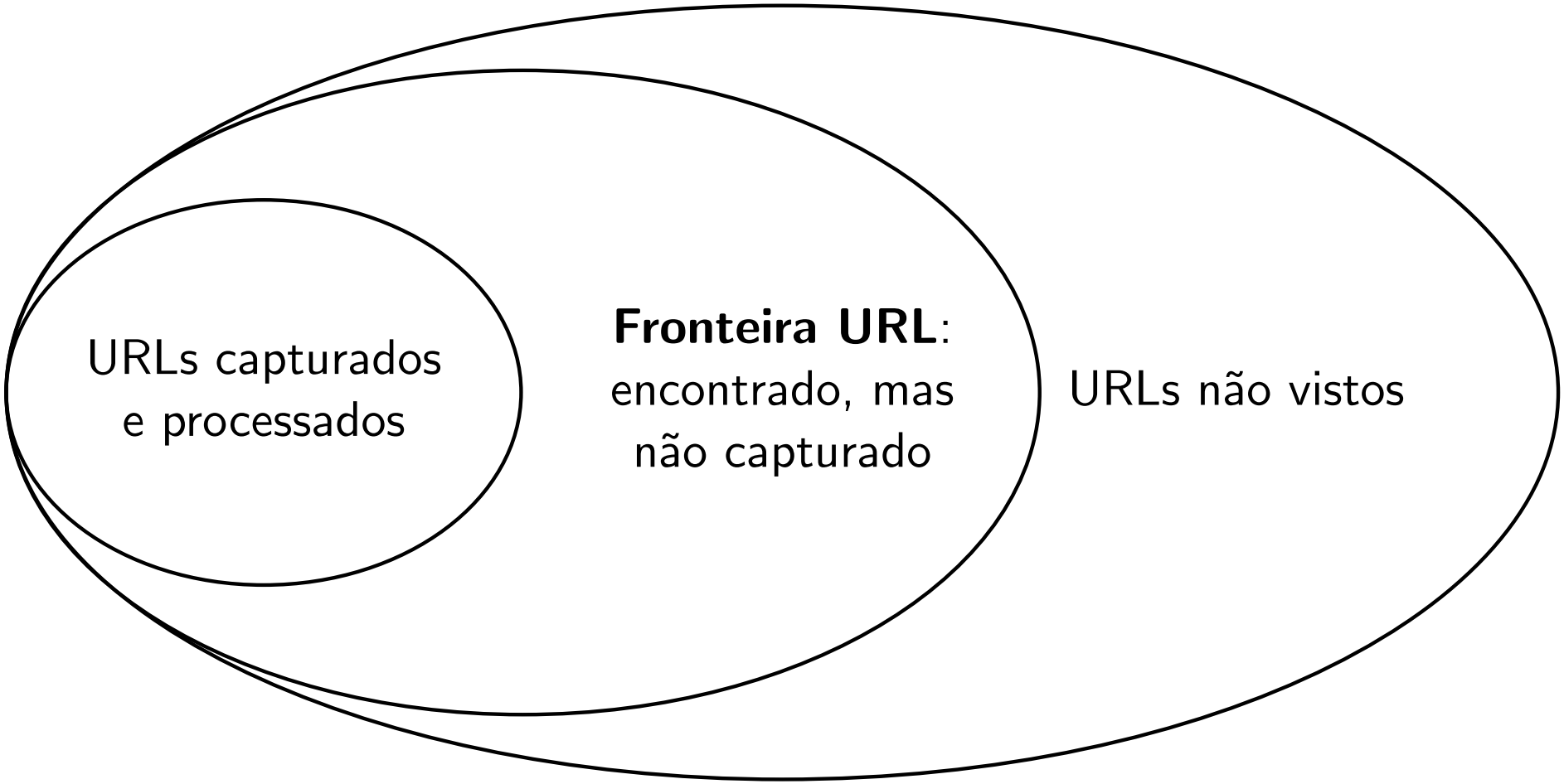
Páginas Web pages mudam frequentemente

A Large-Scale Study of the Evolution of Web Pages, Fetterly 1997

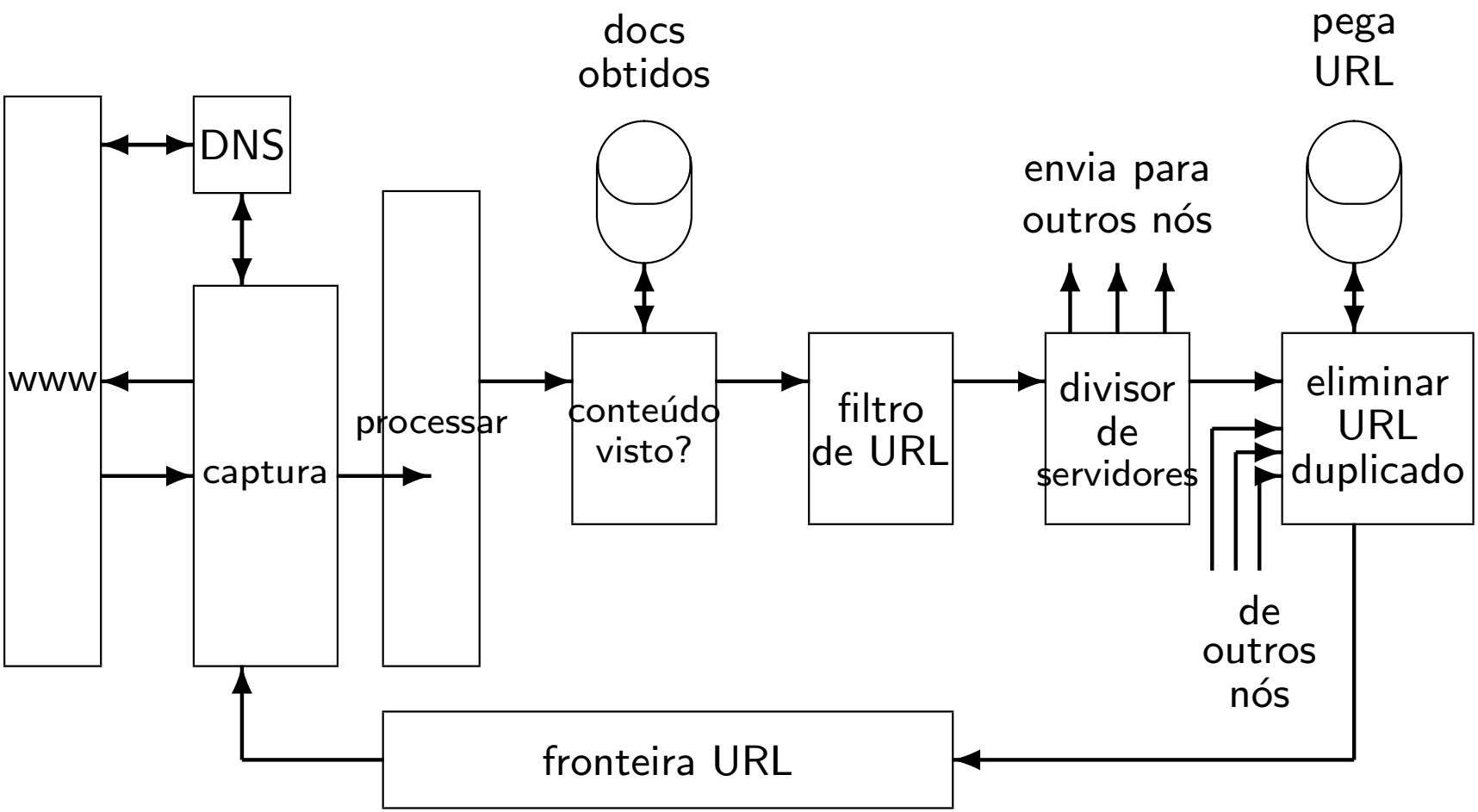


Mudanças em 10 dias.

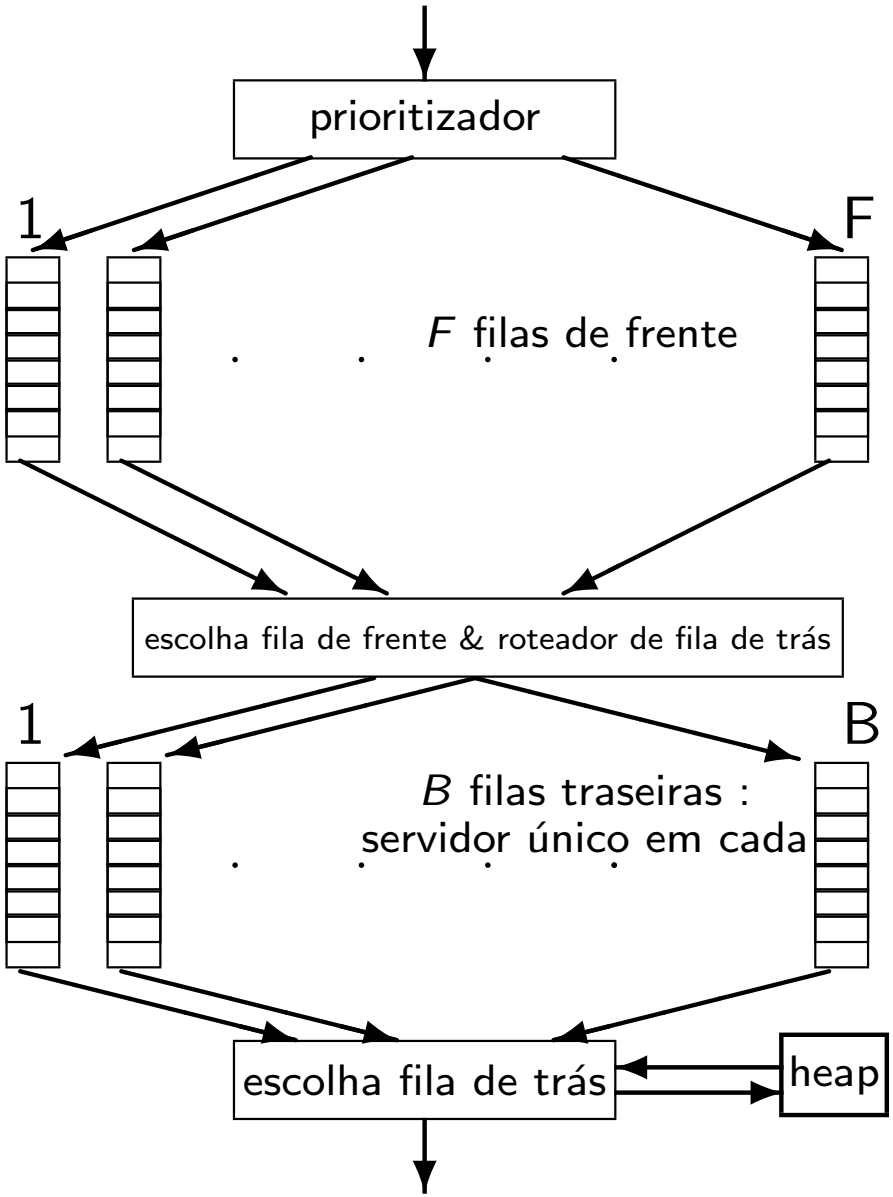
Fronteira URL



Crawler distribuído

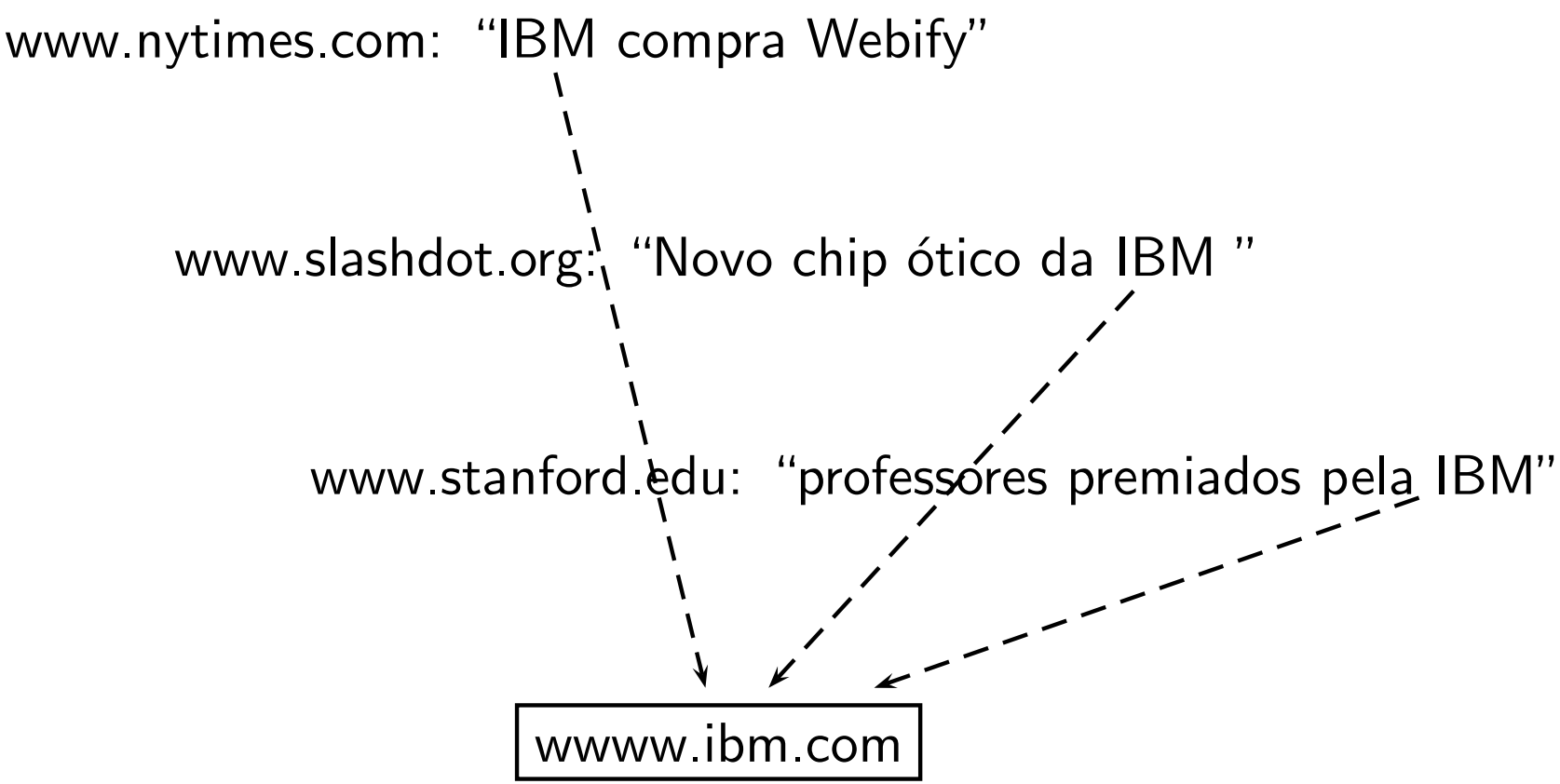


Fronteira URL de Mercator

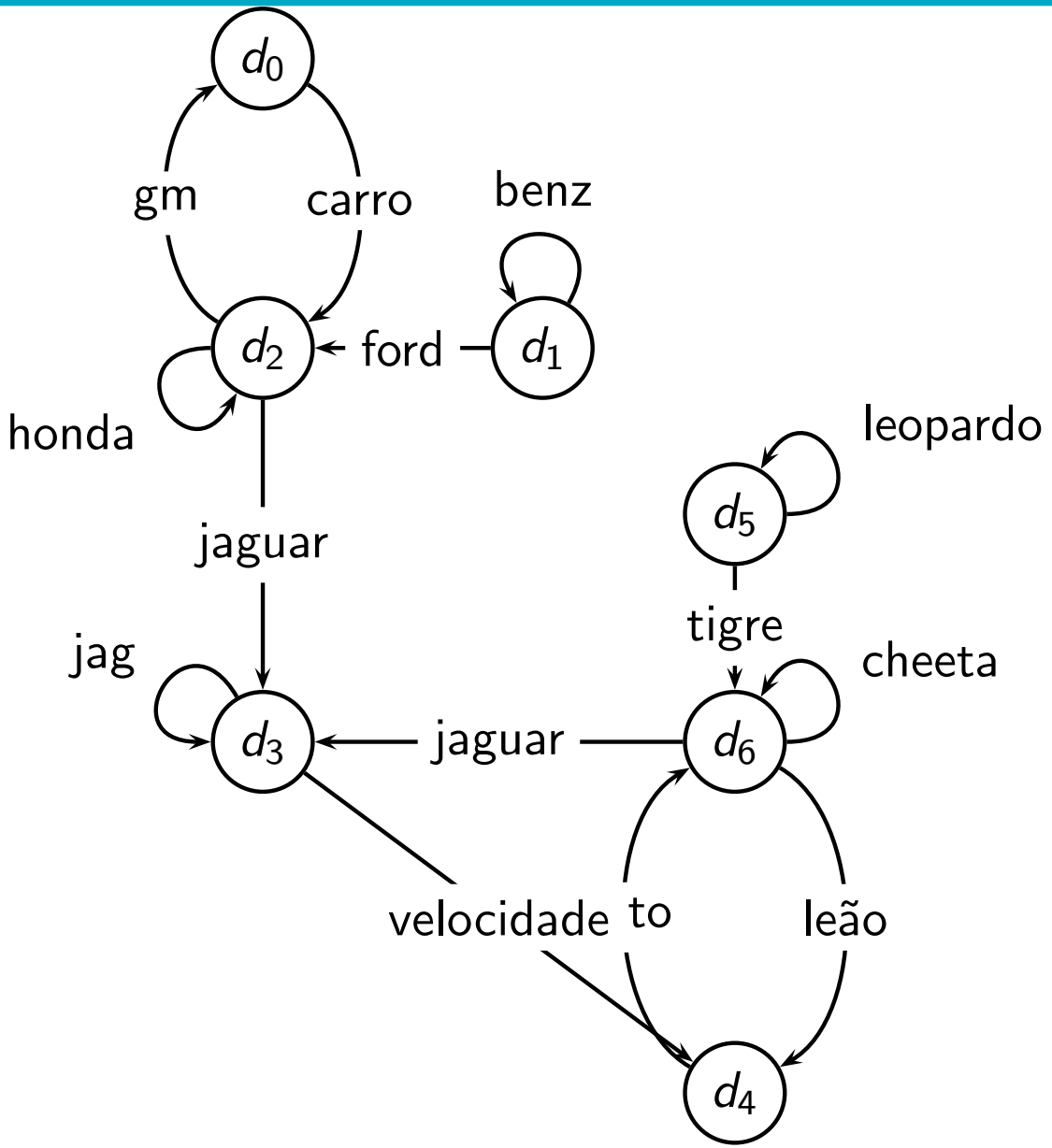


- Fluxo de URLs de entrada do topo para a fronteira
- Filas de frente gerenciam prioridades
- Filas traseiras garantem cortesia

Texto âncora contendo *IBM* apontando para www.ibm.com



Exemplo grafo web

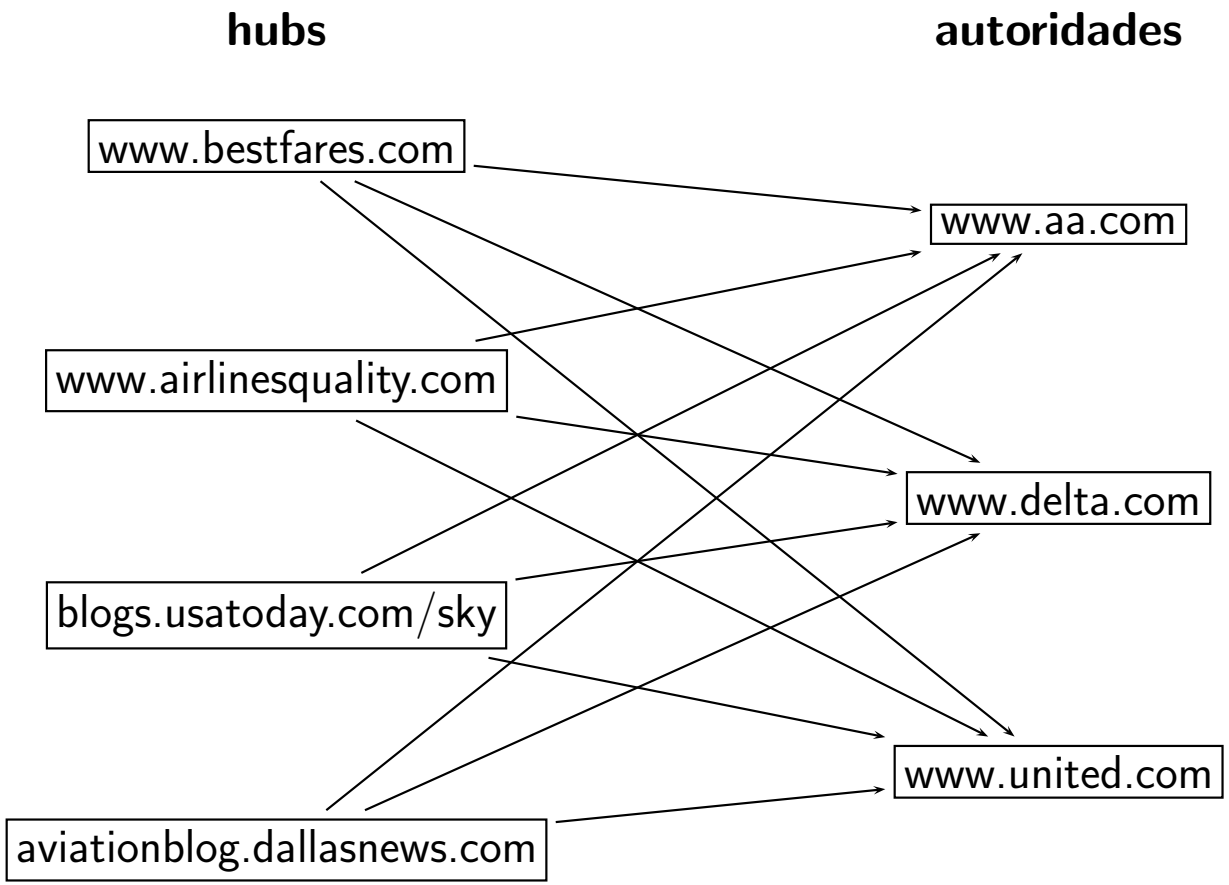


	PageRank
d_0	0.05
d_1	0.04
d_2	0.11
d_3	0.25
d_4	0.21
d_5	0.04
d_6	0.31

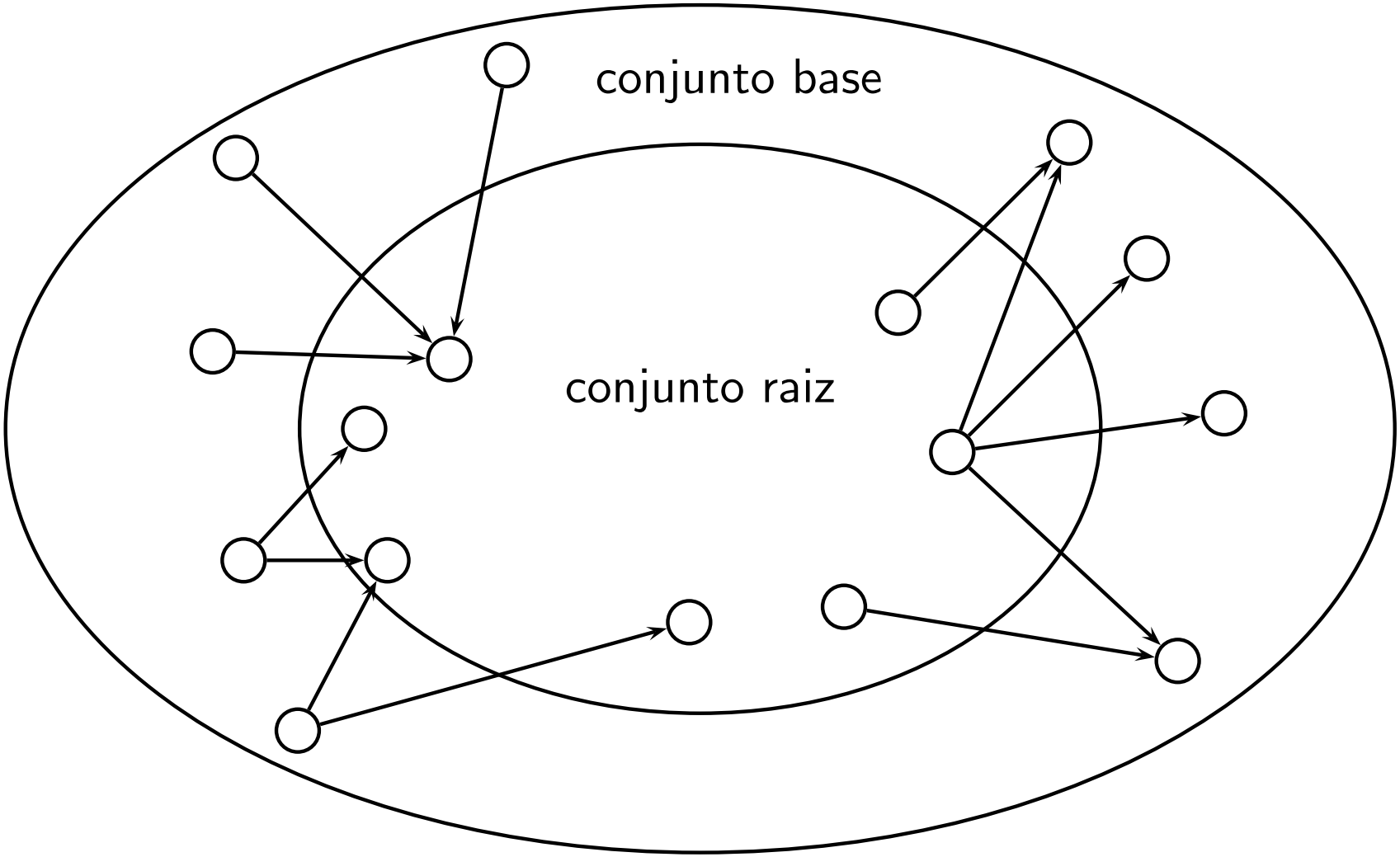
PageRank(d2)< PageRank(d6): porquê?

	a	h
d_0	0.10	0.03
d_1	0.01	0.04

Exemplo de hubs e autoridades



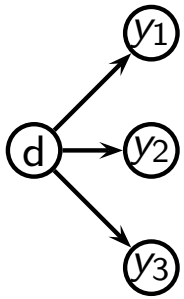
Conjunto raiz e conjunto base (1)



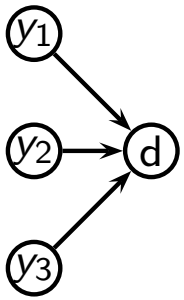
O conjunto raiz Nós para os quais os nós do conjunto raiz nodes linkam Nós que linkam para os nós do conjunto raiz O conjunto base

Atualização iterativa

- Para todos d: $h(d) = \sum_{d \mapsto y} a(y)$



- Para todos d: $a(d) = \sum_{y \mapsto d} h(y)$

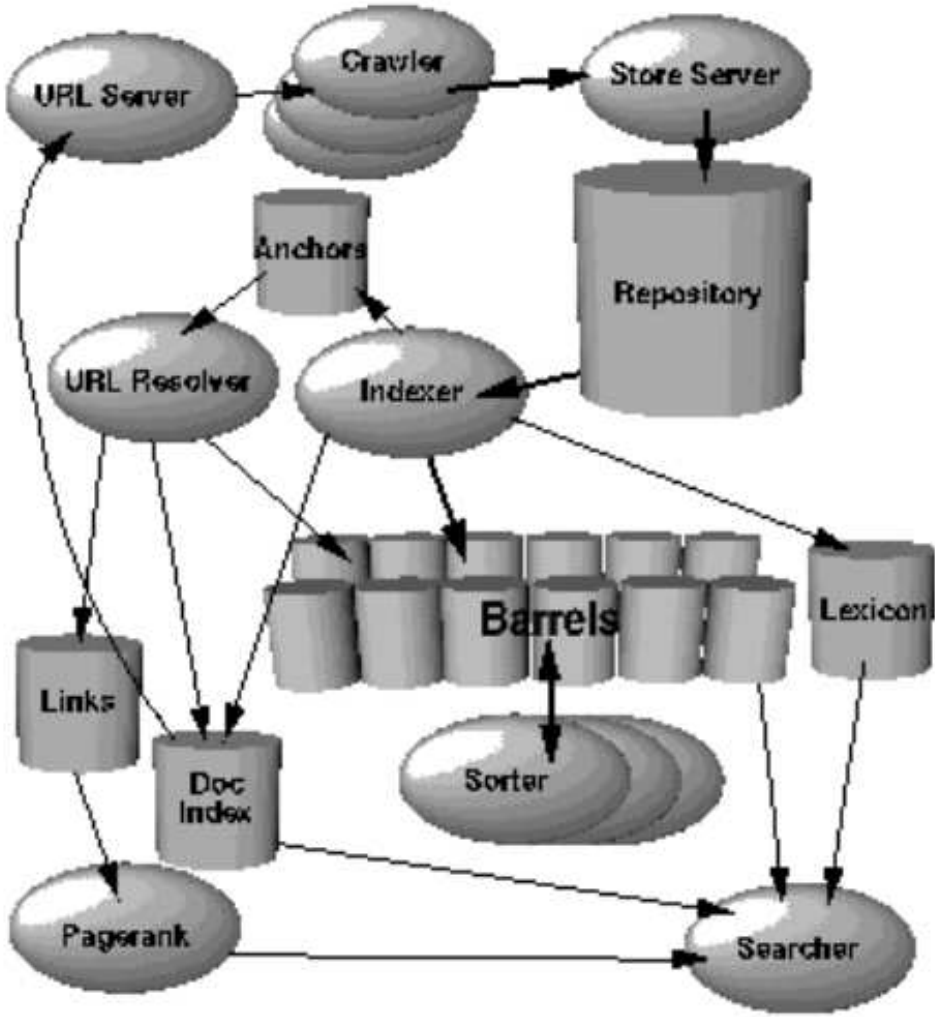


- Iterar esses dois passos até convergência

Anatomia do sistema

- Estruturas de dados
- Crawling
- Indexação
- Busca

Implementação em C e C++.
Execução em sistemas Solaris e Linux.



Listas de `hits`

- Contém lista de ocorrências
- Inclui:
  - posição no documento (`imp` no tipo `plain`)
  - fonte usada
  - maiúsculas/minúsculas
- Ocupa maior parte do espaço usado para índice direto e invertido
- Uso de representação otimizada "à mão"
- Dois tipos de `hits`:
  - `fancy` – em URL, título, texto âncora, meta tag
  - `plain` – o resto

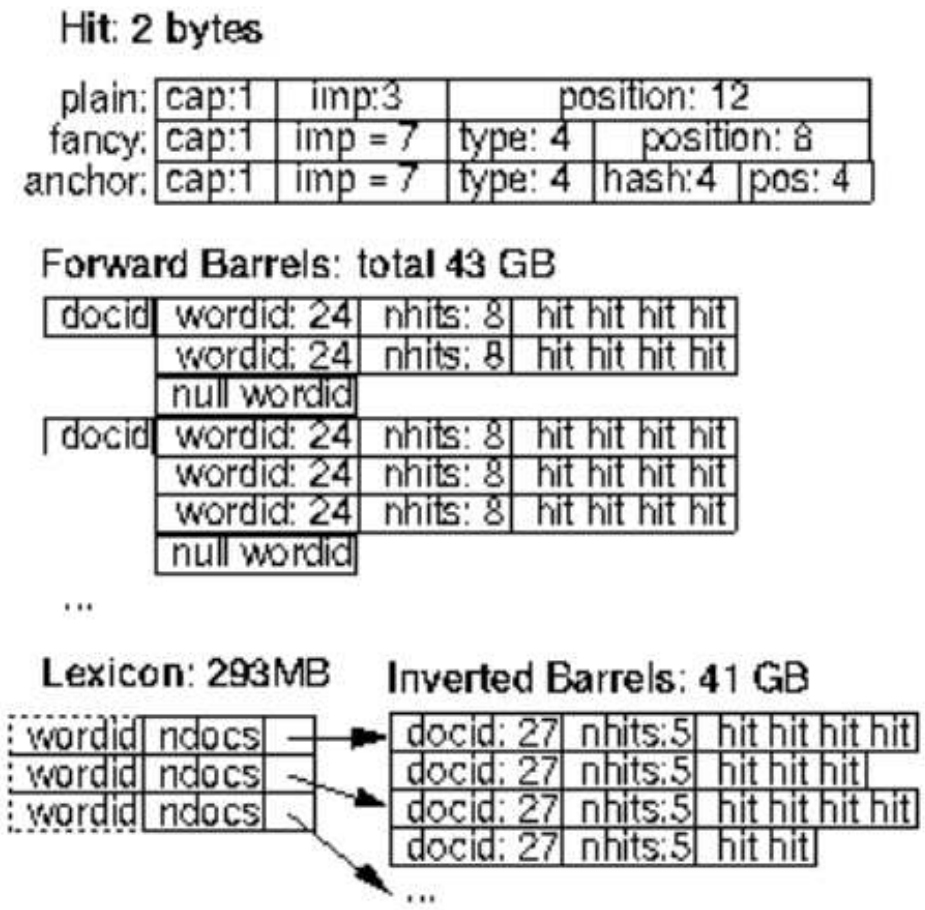


Figure 3. Forward and Reverse Indexes

Índice direto

- Índice direto é criado parcialmente ordenado
- Dividido em 64 barris (barrels)
- cada barrels recebe uma faixa de `wordID`

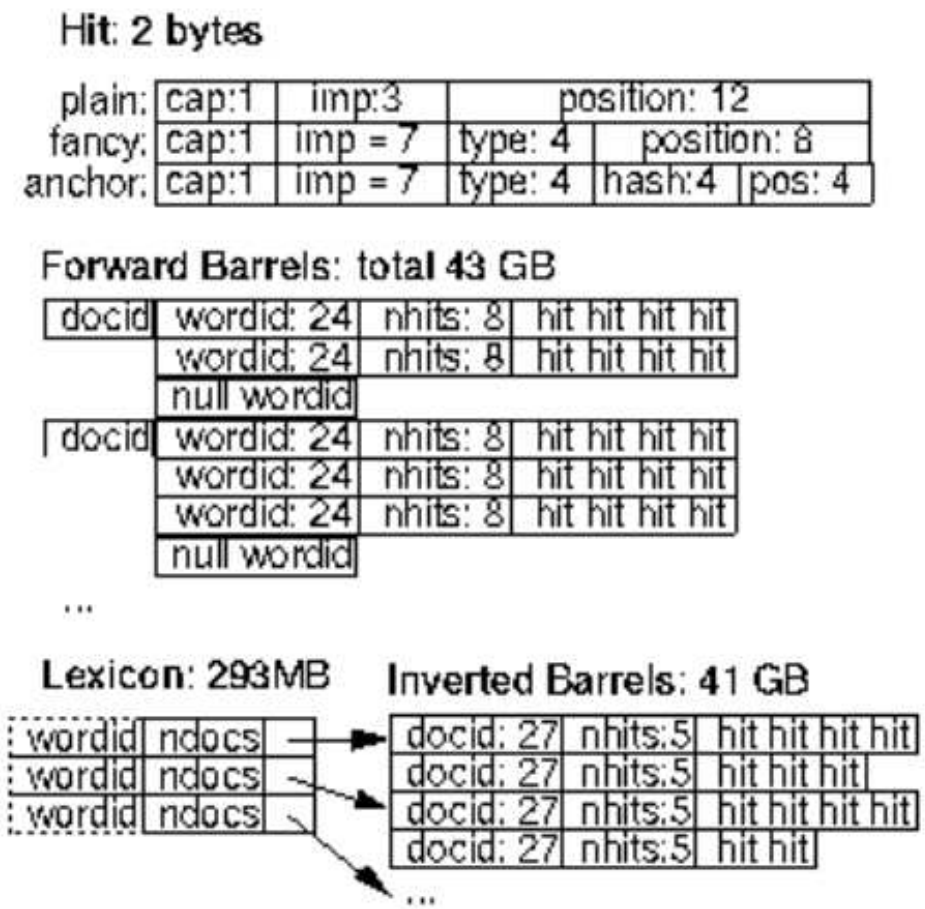


Figure 3. Forward and Reverse Indexes

Índice invertido

- Índice invertido consiste dos mesmos `barrels` que o índice direto
- Porém já foi processado pelo `sorter`
- `doclist` lista `docIDs` ordenados
- Mantém um índice invertido para o título, âncoras e outro índice invertido para o resto

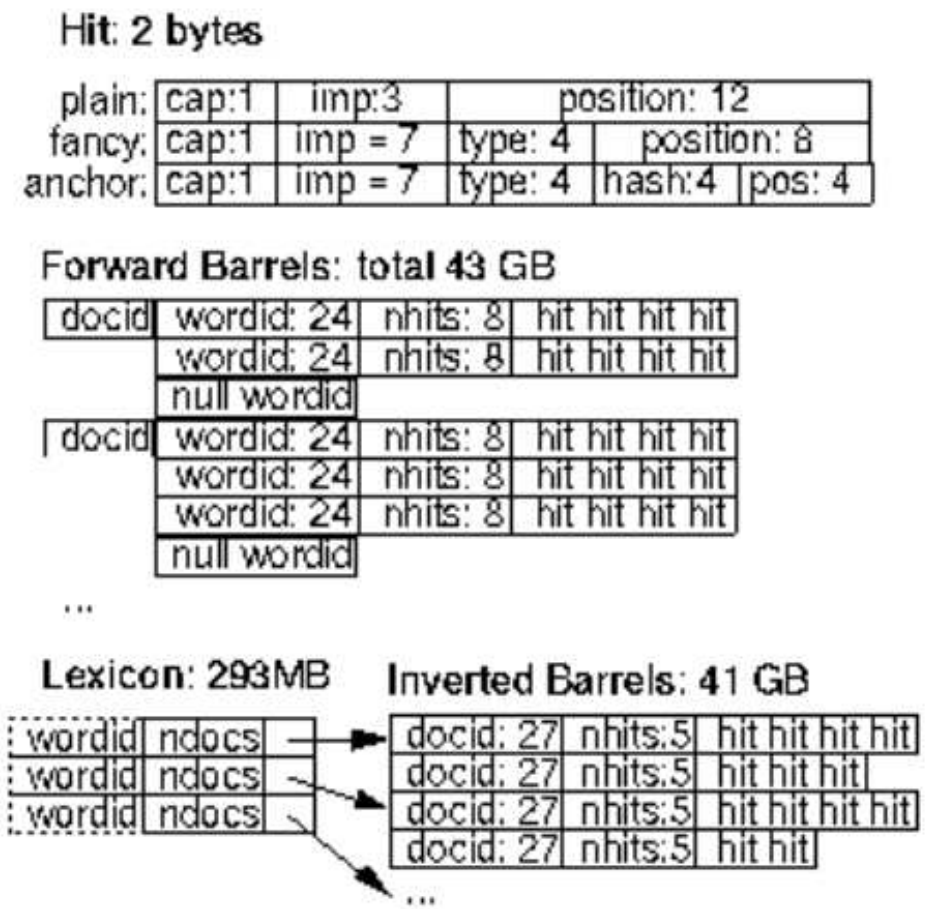


Figure 3. Forward and Reverse Indexes

Desempenho do sistema

- Cerca de 9 dias para baixar 26 milhões de páginas
- Indexador processava 54 páginas por segundo
- 4 máquinas para o `sorter`: 24 horas
- Tempo de consulta: 1 a 10 segundos
  - Culpado: `BigFile` rodando no NFS

	Initial Query		Same Query Repeated (IO mostly cached)	
Query	**CPU Time(s)**	**Total Time(s)**	**CPU Time(s)**	**Total Time(s)**
al gore	0.09	2.13	0.06	0.06
vice president	1.77	3.84	1.66	1.80
hard disks	0.25	4.86	0.20	0.24
search engines	1.31	9.63	1.16	1.16

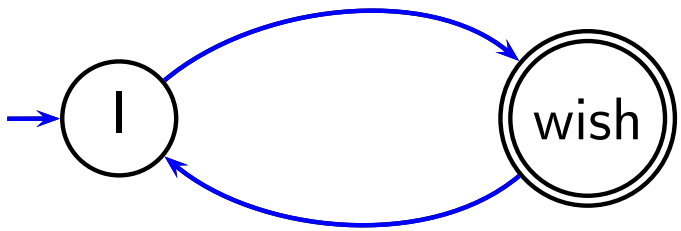
Table 2. Search Times

Trabalho futuro de 1997

- Meta imediata: aumentar capacidade para 100 milhões de docs
  - Em setembro de 2013, Google indexa aproximadamente 40 bilhões de documentos
- Cache de consultas
- Subindices
- Melhoria da atualização de documentos – recrawling
- Operadores booleanos
- Negação
- Stemming
- Obtenção de retorno de relevância
- Uso de clustering (no momento somente de nome do servidor)
- Uso da localidade do usuário
- Expansão da região do texto de âncora

What is a language model?

We can view a finite state automaton as a deterministic language



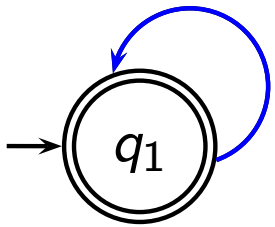
model.

I wish I wish I wish I wish . . . Cannot generate: "wish I wish"

or "I wish I" Our basic model: each document was generated by a

different automaton like this except that these automata are probabilistic.

A probabilistic language model



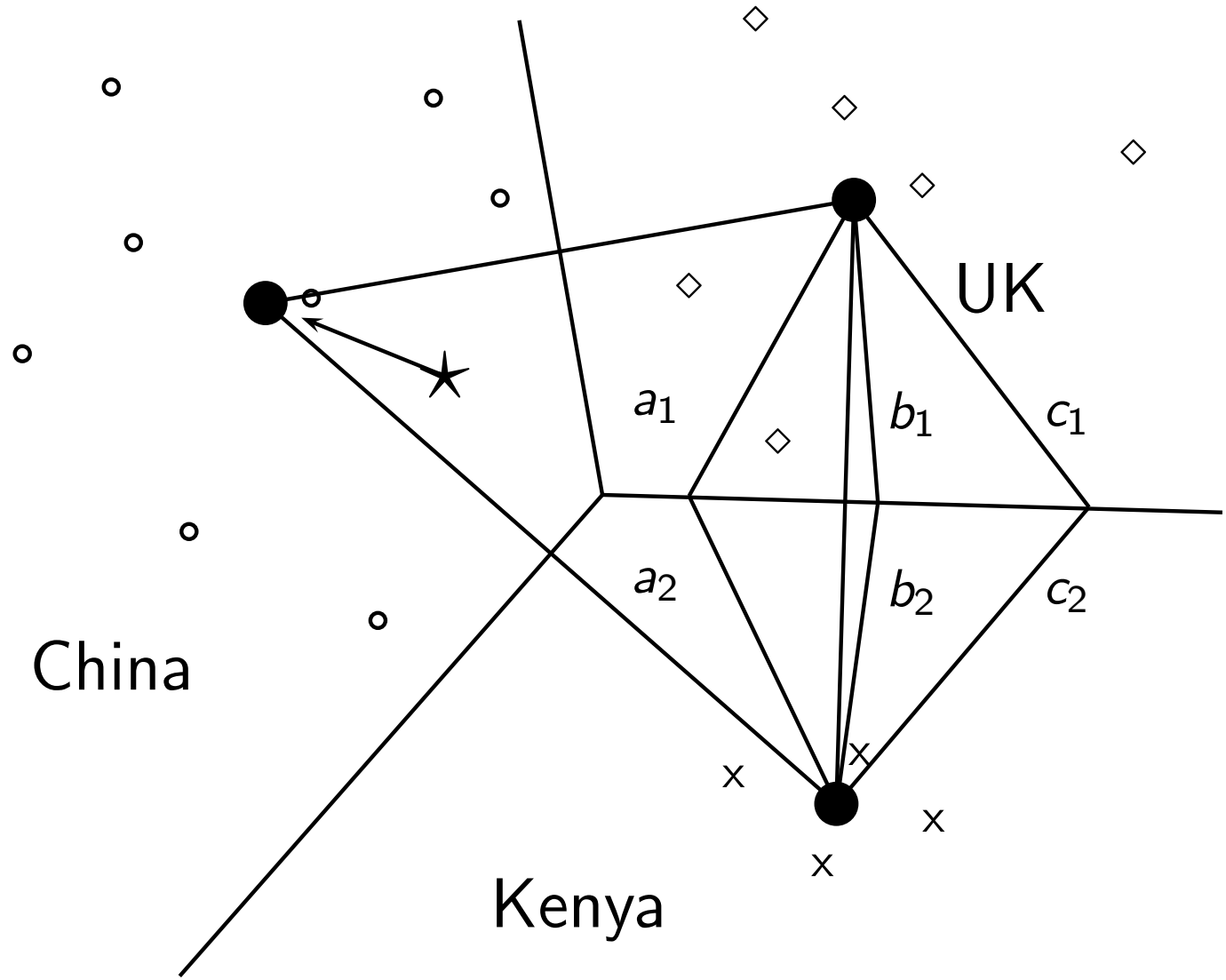
w	$P(w\|q_1)$	w	$P(w\|q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
		...	...

This is a one-state probabilistic finite-state automaton – a unigram language model – and the state emission distribution for its one state q_1. STOP is not a word, but a special symbol indicating that the automaton stops. frog said that toad likes frog STOP

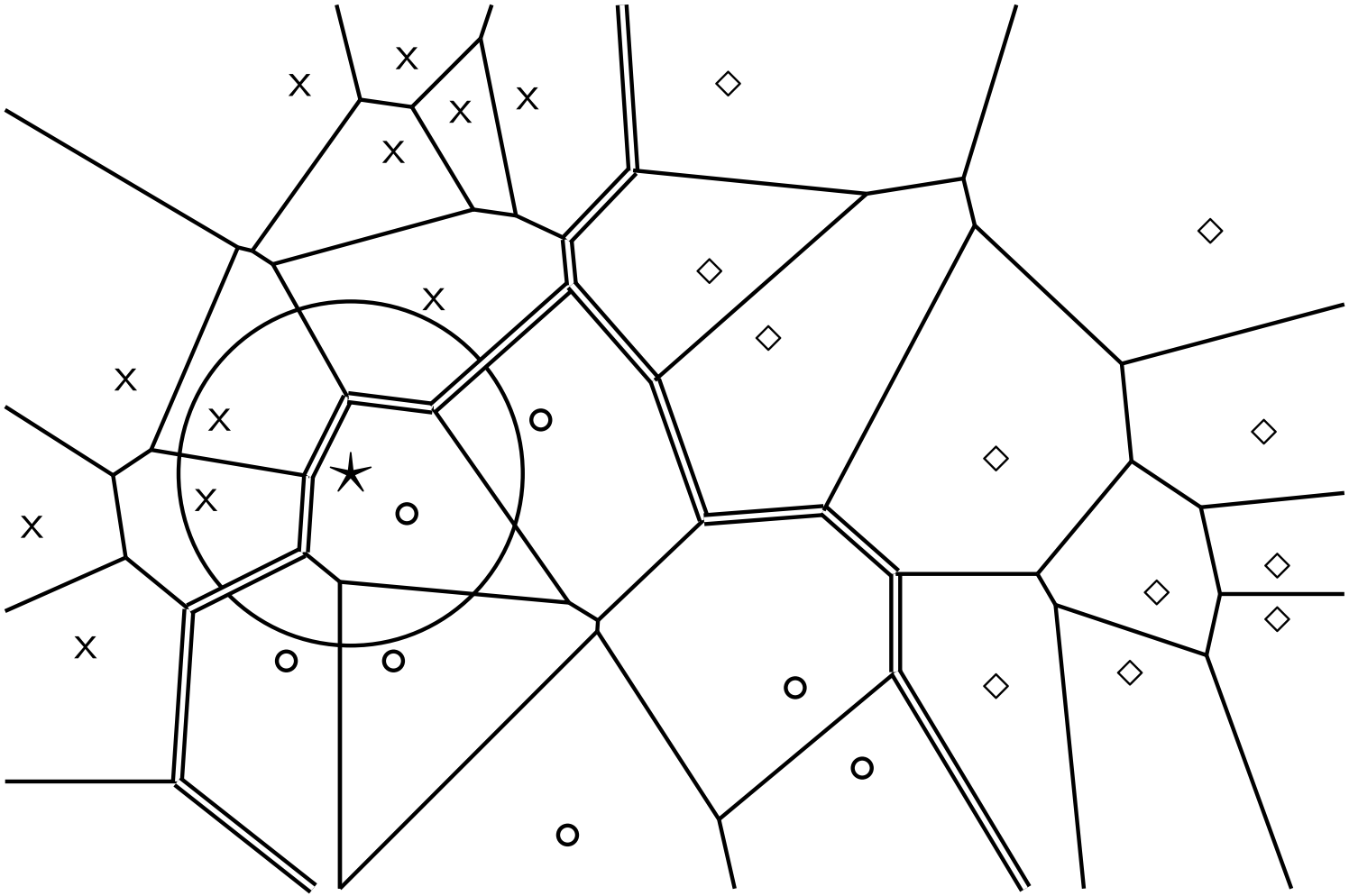
$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2$

$= 0.0000000000048$

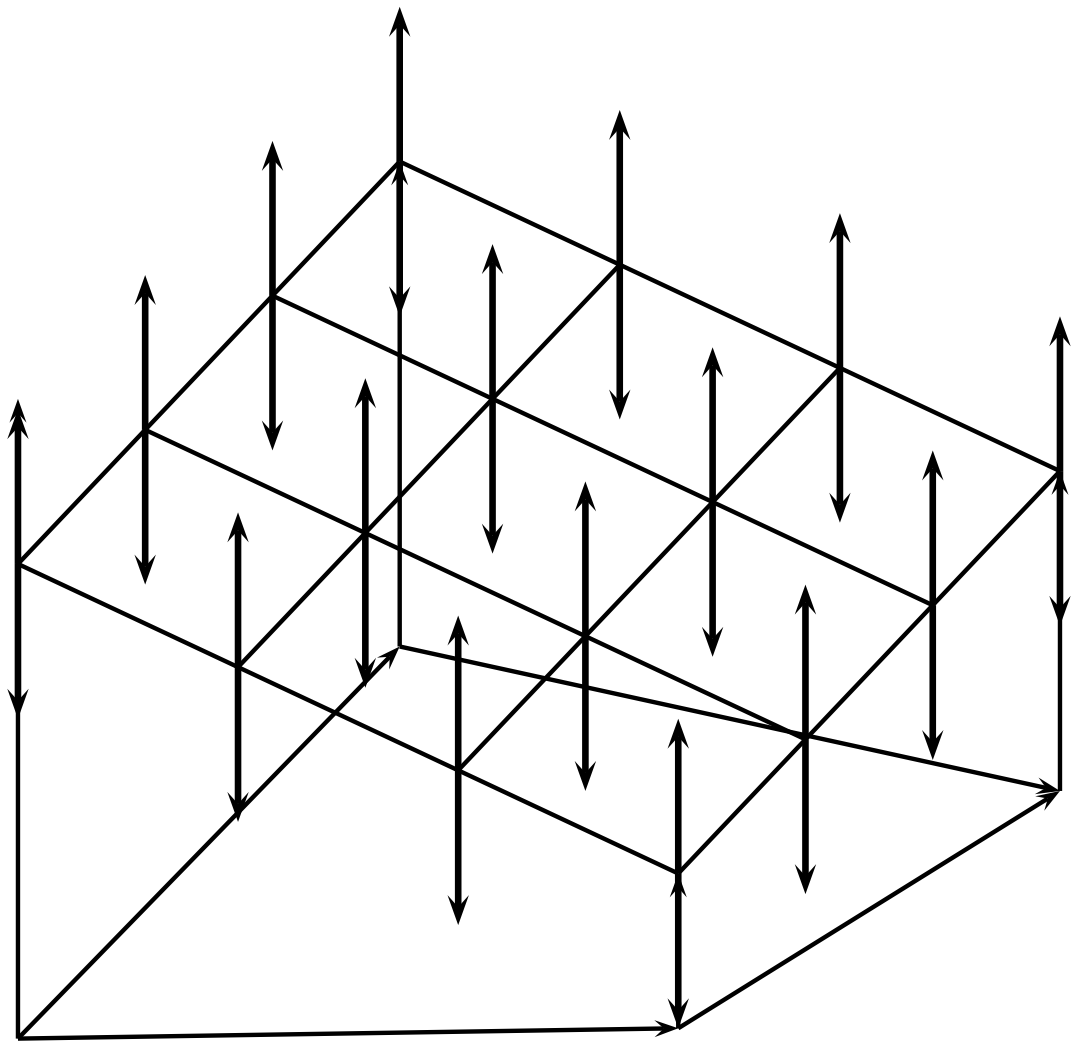
Rocchio illustrated : $a_1 = a_2, b_1 = b_2, c_1 = c_2$



kNN is based on Voronoi tessellation

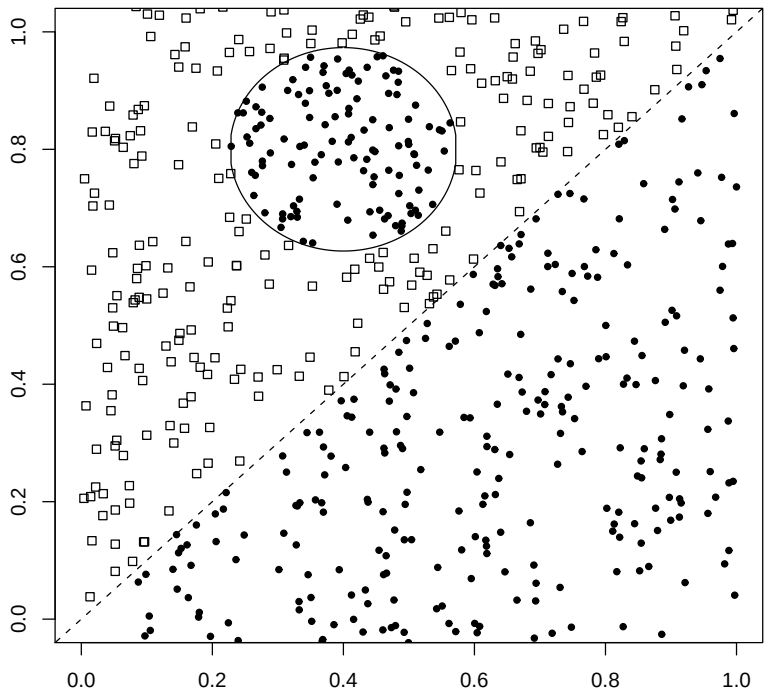


A linear classifier in 3D



- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$
- Example for a 3D linear classifier
- Points $(d_1\ d_2\ d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 \geq \theta$ are in the class c.
- Points $(d_1\ d_2\ d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 < \theta$ are in the complement class $\overline{c}$.

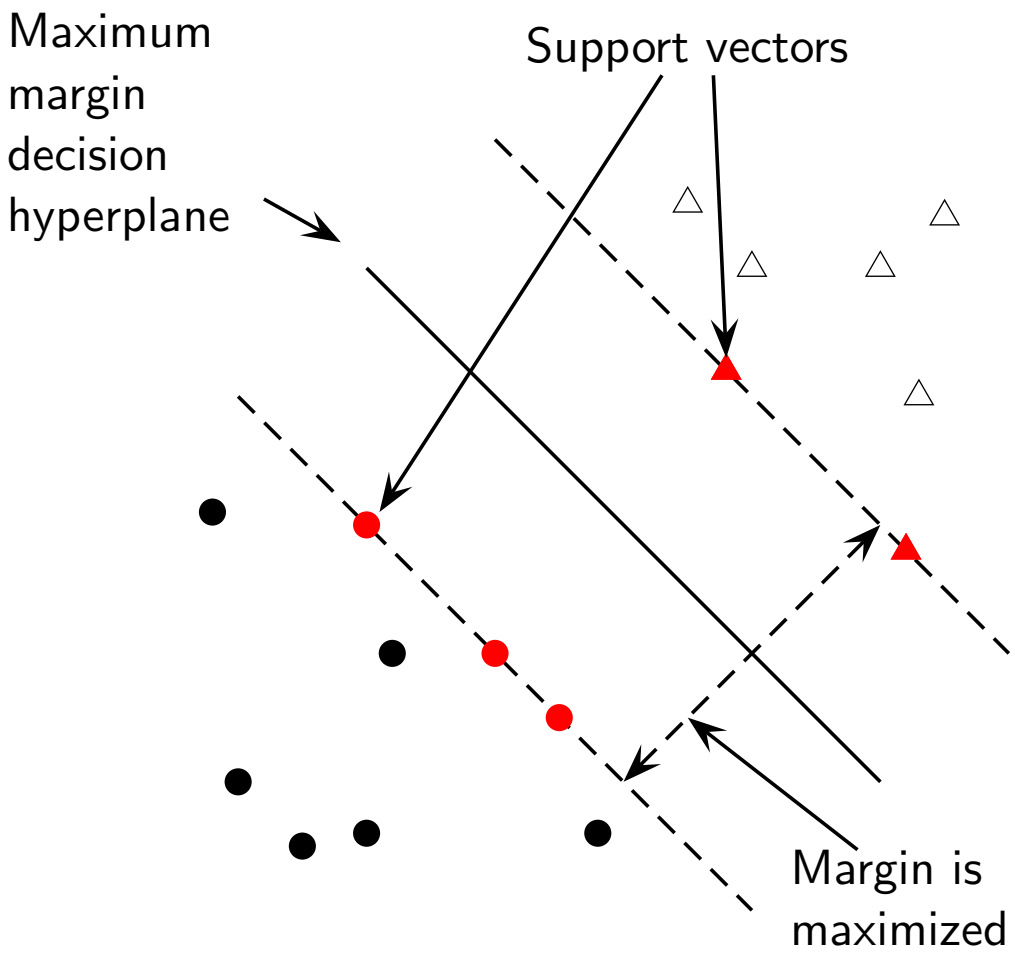
A nonlinear problem



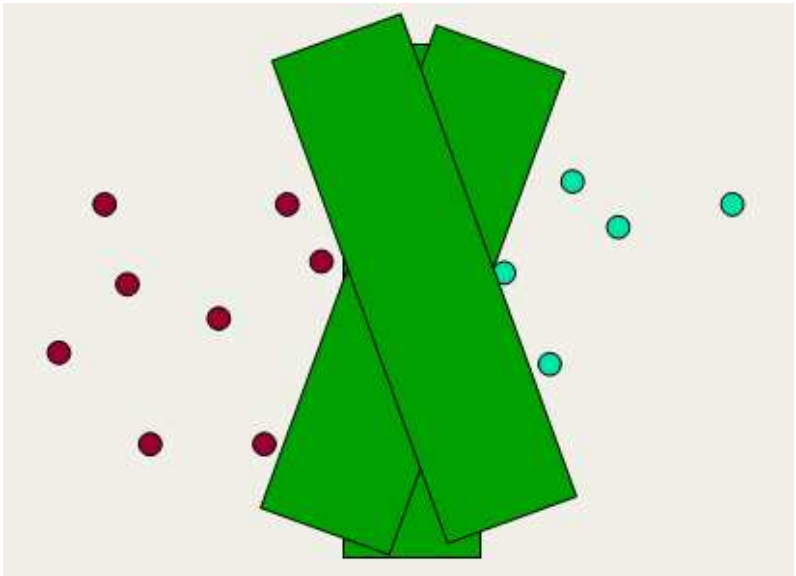
- Linear classifier like Rocchio does badly on this task.
- kNN will do well (assuming enough training data)

Why maximize the margin?

Points near the decision surface are uncertain classification decisions.
A classifier with a large margin makes no low certainty classification decisions (on the training set).
Gives classification safety margin with respect to errors and random variation



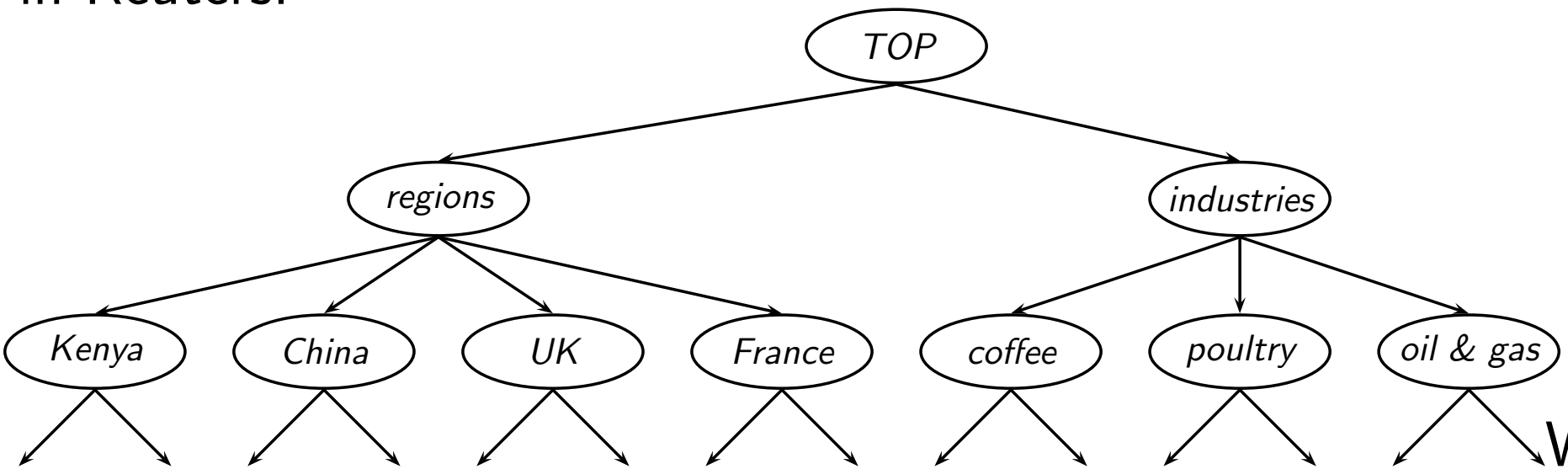
Why maximize the margin?



- SVM classification = large margin around decision boundary

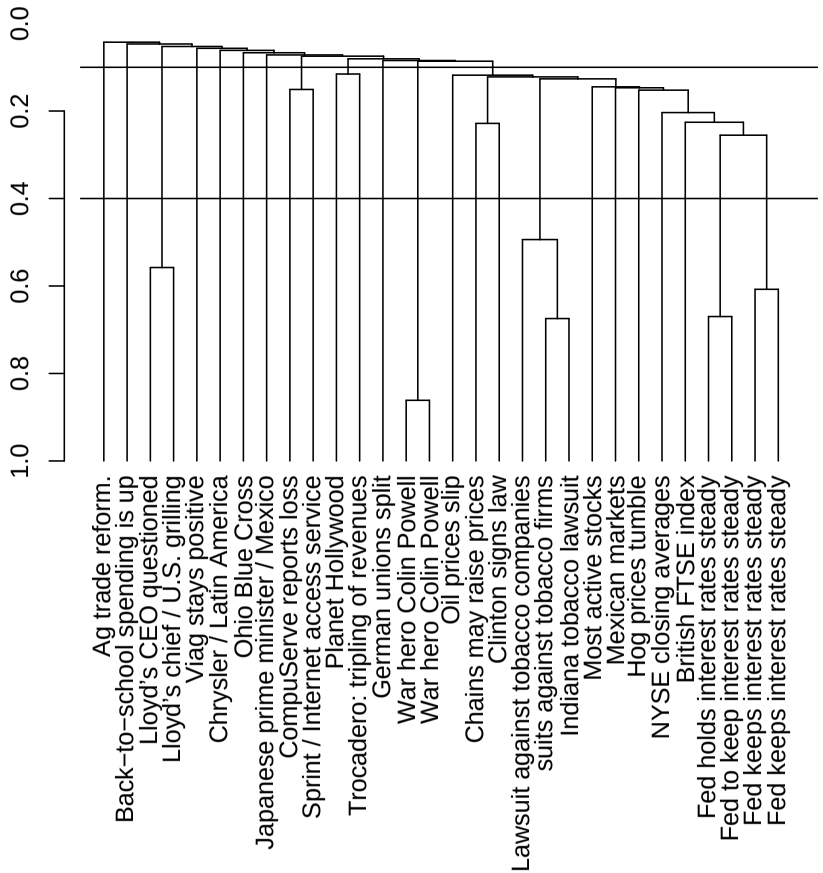
Hierarchical clustering

Our goal in hierarchical clustering is to create a hierarchy like the one we saw earlier in Reuters:



We want to create this hierarchy automatically. We can do this either top-down or bottom-up. The best known bottom-up method is hierarchical agglomerative clustering.

A dendrogram



- The history of mergers can be read off from bottom to top.
- The horizontal line of each merger tells us what the similarity of the merger was.
- We can cut the dendrogram at a particular point (e.g., at 0.1 or 0.4) to get a flat clustering. □

Single-link clustering