Tópicos em Recuperação de Informação[1]

Nivio Ziviani

[1]Conjunto de transparências elaborado por Nivio Ziviani e Patrícia Correia

Consulta Ordenada

Coordinate matching:

Técnica simples: conta o número de termos da consulta que aparece em cada documento.

- Quanto maior o número de termos da consulta em um documento, mais provável que ele seja relevante.

- Consulta se torna híbrida, intermediária entre a consulta conjuntiva AND e a consulta disjuntiva OR:
  - documento que contenha qualquer dos termos é resposta potencial.
  - preferência é para documentos que contenham todos ou a maioria dos termos.

- Toda informação necessária está no arquivo invertido.

Consulta Ordenada

d	**Documento** D_d
1	Pease porridge hot, pease porridge cold,
2	Pease porridge in the pot,
3	Nine days old.
4	In the pot cold, in the pot hot,
5	Pease porridge, pease porridge,
6	Eat the lot.

Coleção com 6 documentos e 10 termos.

$Q = \text{``}eat'' \Rightarrow D_6$

$Q = \text{``}hot\ porridge''$

Conjuntiva: D_1

Coordinate matching: $D_1 > D_2 = D_4 = D_5 > D_3 = D_6$

Similaridade do Produto Interno

Produto interno entre vetor da consulta e vetores de docs.

(a)	d	**Vetores de documentos** ($w_{d,t}$)									
		col	*day*	*eat*	*hot*	*lot*	*nin*	*old*	*pea*	*por*	*pot*
	1	1	0	0	1	0	0	0	1	1	0
	2	0	0	0	0	0	0	0	1	1	1
	3	0	1	0	0	0	1	1	0	0	0
	4	1	0	0	1	0	0	0	0	0	1
	5	0	0	0	0	0	0	0	1	1	0
	6	0	0	1	0	1	0	0	0	0	0
(b)	*eat*	0	0	1	0	0	0	0	0	0	0
	hot porridge	0	0	0	1	0	0	0	0	1	0

Similaridade entre consulta Q com documento D_d:

$$M(Q, D_d) = Q.D_d$$

onde,

$$X.Y = \sum_{i=1}^{n} x_i.y_i$$

Ex.: $M(\text{“}hot,\ porridge'',\ D_1) = 2$

Problemas

1. Não considera freqüência do termo.

   Ex.: $porridge$ aparece 2 vezes em D_1 e 1 vez em D_2
   $Q =$ "$porridge$": os 2 documentos são ordenados igualmente.

2. A fórmula não considera a raridade do termo.

   Ex.: eat aparece somente em D_6 e $porridge$ aparece em D_1, D_2 e D_5.

3. Longos documentos são favorecidos (talvez indevidamente pela diversidade do vocabulário, devido ao tamanho).

Soluções

1^o **Problema:**

Substituir bit por inteiro indicando o número de vezes que o termo aparece no documento.

Ex.: $M(\text{“}hot,\ porridge'',\ D_1)$

(0,0,0,1,0,0,0,0,1,0).(1,0,0,1,0,0,0,2,2,0) $= 3$

Esta idéia pode ser generalizada:

$w_{d,t}$: document-term weight e $w_{q,t}$: query-term weight

$$M(Q, D_d) = Q.D_d = \sum_{i=1}^{n} w_{q,t}.w_{d,t}$$

Quando t não ocorre em Q, $w_{q,t} = 0$

Logo,

$$M(Q, D_d) = \sum_{t \in Q} w_{q,t}.w_{d,t}$$

Soluções

2^o **Problema:**

Dar peso aos termos de acordo com o inverso da freqüência em documentos - idf (*inverse document frequency*).

$$w_t = \log_e \left(1 + \frac{N}{f_t} \right)$$

onde,

N: número de documentos na coleção.

f_t: número de documentos que contém temo t.

$$w_{d,t} = (1 + \log_e f_{d,t}).w_t$$

Conhecida como regra TF x IDF (*term frequency x inverse document frequency*)

Lei de Zipf:

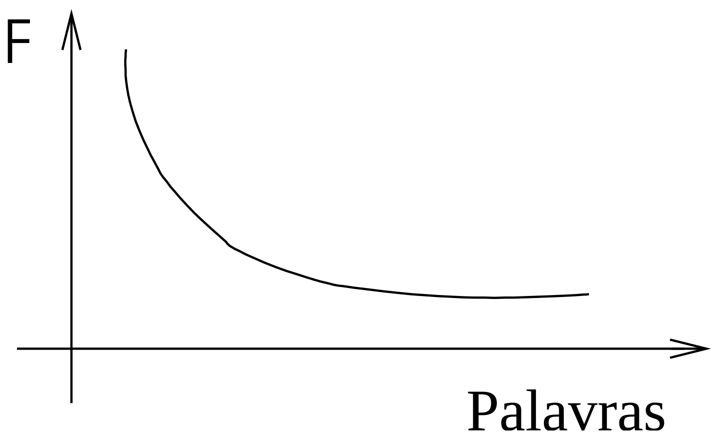
Freqüência do termo é inversamente proporcional ao seu *"rank"*.

Soluções

Zipf, 1949

Human Behavior and the Principle of Least Effort

Frequência da i-ésima palavra mais frequente é $1/i^{\Theta}$ vezes a da palavra mais frequente, $\Theta \geq 1$



Soluções

w_t: Peso do termo t

$w_t = \frac{1}{f_t}$

$w_t = log_e(1 + \frac{N}{f_t})$

$w_t = log_2\frac{N}{f_t}$ Persin, Zobel & Sacks_Davis: JASIS, 1996

- Logaritmo não deixa um termo com $f_t = 1$ ser duas vezes mais importante que um temo com $f_t = 2$

Soluções

$r_{d,t}$: *Relative term frequency*

- Existem muitas maneiras para calcular $r_{d,t}$

$r_{d,t} = 1 + log_e f_{d,t}$

$r_{d,t} = 1$

$r_{d,t} = f_{d,t}$

$r_{d,t} = log_2 f_{d,t}$ Persin, Zobel & Sacks_Davis: JASIS, 1996

- Crescimento menor quando $f_{d,t}$ cresce com o uso do logaritmo (crescimento monotônico)

Valores de documentos podem ser calculados como:

$W_{d,t} = r_{d,t}$ ou

$W_{d,t} = r_{d,t} \cdot w_t$ (TF $\times$ IDF)

$r_{d,t}$: crescimento monotônico
w_t: decrescimento monotônico

Soluções

- A direção de dois vetores (documento e consulta), ou mais precisamente a *diferença da direção de cada vetor*, é o que importa.
- Ângulo Θ entre dois vetores satisfaz:

$$X.Y = |X||Y| \cos \Theta$$

$X.Y$: produto interno

$|X| = \sqrt{\sum_{i=1}^{n} x_i^2}$: distância Euclidiana do vetor X

X: vetor n-dimensional $< x_i >$

$$\cos \Theta = \frac{X.Y}{|X||Y|}$$

Soluções

3^o **Problema:**

Documentos longos são favorecidos porque contêm mais termos e o valor do produto interno cresce.

Introduzir a normalização:

$$M(Q, D_d) = \frac{\sum_{t \in Q} w_{q,t} . w_{d,t}}{\mid D_d \mid}$$

onde,

$\mid D_d \mid = \sum_i f_{d,t}$ (comprimento do documento D_d)

Soluções

Implicações:

1. Normalização descrita anteriormente é a distribuição Euclidiana do documento – i.e., comprimento no espaço n-dimensional dos pesos documento-termo que descrevem o documento

2. Fórmula dá uma visualização clara do que a regra de ranking significa:

   - Documentos são pontos na região positiva do espaço n-dimensional, onde pequenos documentos estão mais próximos da origem e *consultas são raios a partir da origem*

Modelo de Espaço Vetorial

Documentos e consultas são representados por vetores em um espaço n-dimensional.

Da álgebra vetorial:

$$\cos\Theta = \frac{X.Y}{\mid X \mid . \mid Y \mid} = \frac{\sum_{i=1}^{n} x_i . y_i}{(\sqrt{\sum_{i=1}^{n} x_i^2})(\sqrt{\sum_{i=1}^{n} y_i^2})}$$

Normalização: distância Euclidiana do documento (comprimento no espaço n-dimensional do conjunto de pesos doc-termo).

Similaridade expressa pelo $\cos\Theta$: quanto maior o cosseno maior a similaridade.

$\cos\Theta = 1$ para $\Theta = 0$

$\cos\Theta = 0$ para $\Theta = 90^o$

Modelo de Espaço Vetorial

$$\cos(Q, D_d) = \frac{Q.D_d}{\mid Q \mid \mid D_d \mid} = \frac{\sum_{i=1}^{n} w_{q,t}.w_{d,t}}{W_q \; W_d}$$

onde,

$W_d = \sqrt{\sum_{i=1}^{n} w_{d,t}^2}$ e $W_q = \sqrt{\sum_{i=1}^{n} w_{q,t}^2}$

- Pode-se usar qualquer método *"term-weighting"*.

- Um bom método (simples): TF x IDF

$$\cos(Q, D_d) = \frac{1}{W_d \; W_q} \sum_{t \in Q \bigcap D_d} (1 + \log_e f_{d,t}). \log_e (1 + \frac{N}{f_t})$$

- Como W_q é constante para uma dada consulta ele pode ser desprezado sem alterar o *"ranking"*.

Modelo de Espaço Vetorial

d	Vetores de documentos ($w_{d,t}$)									$\mathbf{W_d}$
	col	*day*	*eat*	*hot*	*lot*	*nin*	*old*	*pea*	*por*	*pot*	
1	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.7	1.7	0.0	2.78
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.73
3	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.73
4	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.7	2.21
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.7	1.7	0.0	2.39
6	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.41
f_t	2	1	1	2	1	1	1	3	3	2	
w_t	1.39	1.95	1.95	1.39	1.95	1.95	1.95	1.10	1.10	1.39	

d	Consulta			
	eat	*porridge*	*hot porridge*	*eat nine day old porridge*
	$W_q = 1.95$	$W_q = 1.10$	$W_q = 1.77$	$W_q = 3.55$
1	0.00	0.61	0.66	0.19
2	0.00	0.58	0.36	0.18
3	0.00	0.00	0.00	0.63
4	0.00	0.00	0.36	0.00
5	0.00	0.71	0.44	0.22
6	0.71	0.00	0.00	0.39
Top	6	5	1	3

Medida do Cosseno

- Exige mais informação do que o processamento de consultas booleanas

- Decisões importantes sobre como estruturar a informação para fazer o *processamento do ranking eficiente*:

  1. Precisamos da $f_{d,t}$ no índice

  2. Como avaliar a medida do cosseno

     – Para grandes coleções é potencialmente caro
       * tempo CPU
       * largura de banda de disco
       * demanda por memória

     – Soluções
       * limitar o número de candidatos ao ranking
       * ordenar por $f_{d,t}$
       * representação eficiente de w_d

Implementação da Medida do Cosseno

Freqüência de t em d ($f_{d,t}$):

- $f_{d,t}$ tem que ser adicionado ao índice.

- $f_{d,t}$ é necessário para a maioria dos métodos de atribuição de peso aos termos (*term weight assignment*).

- Entrada para *"porridge"* no arquivo invertido:
  $<$3;[1,2,5]$>$ passa a ser $<$3;[(1,2)(2,1),5,2)]$>$

  onde,
  $f_{porridge} = 3$ (*porridge* ocorre em 3 documentos: 1,2,5)
  $f_{1,porridge} = 2$ (*porridge* ocorre 2 vezes no documento 1)

$$\Downarrow$$

$$< f_t; [(d_1, f_{1,t}), (d_2, f_{2,t}), ..., (d_{f_t}, f_{d_{f_t},t})] >$$

Implementação da Medida do Cosseno

Como resolver problema de espaço extra necessário para armazenar $f_{d,t}$?

Código unário:

1	0
2	10
3	110

Logo, a entrada para o temo t no arquivo invertido cresce de um número de bits igual ao número total de ocorrências de t na coleção.

$$F_t = \sum_{d=1}^{N} f_{d,t} \quad \text{(total de ocorrências de } t\text{)}$$

$$F = \sum_{i=1}^{n} F_t \quad \text{(total de palavras na coleção)}$$

Logo, o arquivo invertido cresce de F bits.

Implementação da Medida do Cosseno

Fato:

É possível indexar toda palavra do texto usando menos de 1 byte por apontador (mesmo usando $f_{d,t}$)

Elias-γ:

- γ nunca é muito pior que unário, mas pode ser muito melhor!
  - Para documentos pequenos ocupa 1 bit a mais do que o unário
  - Muito melhor quando existem muitos termos com $f_{d,t}$ alto

Implementação da Medida do Cosseno

Método	**Bits por apontador**			
	Bíblia	*GNUbib*	*Comact*	*TREC*
Unário	1.27	1.16	1.74	2.49
γ	1.38	1.23	1.88	2.13
δ	1.53	1.32	2.15	2.41
Observed frequency	1.27	1.15	1.72	2.02
Interpolative	0.76	0.45	1.41	1.77

- Coleções *Bible*, *GNUbib*, *Comact*: código unário é perto do ótimo. Isto ocorre porque a distribuição das probabilidades é tipo: $Pr[x] = (\frac{1}{2})^x$ (probabilidades são múltiplos de 1/2).

- Usando compressão, mesmo para índice aumentando de $f_{d,t}$, para TREC o índice ocupa 6.3% do texto.

- Elias-γ: melhor opção para coleções grandes em linguagem natural. Consome $1 + 2\log x$ bits (menor do que o unário para termos com $f_{d,t}$ alto).

Cálculo do Valor do Cosseno

$$\cos(Q, D_d) = \frac{1}{W_d\, W_q} \sum_{t \in Q \bigcap D_d} (1 + \log_e f_{d,t}).\log_e(1 + \frac{N}{f_t})$$

$W_q =$ constante

$W_d = \sqrt{\sum_{i=1}^{n} w_{d,t}^2}$

$w_{d,t} = (1 + \log_e f_{d,t}).w_t$

$w_t = \log_e(1 + \frac{N}{f_t})$

Estruturas de dados:

1. Arranjo para W_d (pode ser pré-computado).

2. Acumuladores indexados por d para acumular $\cos(Q, D_d)$ (qualquer doc d contendo algum termo de Q faz $A_d > 0$).

Cálculo do Valor do Cosseno

Para recuperar r documentos usando a medida do cosseno

1. $A \leftarrow \{\}$. A é o conjunto de acumuladores.
2. **Para cada** termo na consulta $t \in Q$ **faça**
   (a) Obtenha t.
   (b) Pesquise no vocabulário.
   (c) Armazene f_t e o endereço de I_t, a entrada do arquivo invertido para t.
   (d) $w_t \leftarrow 1 + \log_e (N/f_t)$.
   (e) Leia a entrada do arquivo invertido I_t.
   (f) **Para cada** par $(d, f_{d,t})$ em I_t **faça**
      i. **Se** $A_d \notin A$ **então**
         $A_d \leftarrow 0$,
         $A \leftarrow A + \{A_d\}$.
      ii. $A_d \leftarrow A_d + \log_e (1 + f_{d,t}) * w_t$
3. **Para cada** $A_d \in A$ **faça**
   $A_d \leftarrow A_d / W_d$.
   A_d agora é proporcional ao valor $cosseno(Q, D_d)$.
4. **Para** $1 \leq i \leq r$ **faça**
   (a) Escolha d tal que $A_d = max\{A\}$.
   (b) Procure o endereço do documento d.
   (c) Recupere o documento d e o apresente para o usuário.
   (d) $A \leftarrow A - \{A_d\}$.

Cálculo do Valor do Cosseno

Pontos importantes:

1. W_q e $f_{q,t}$ são ignorados
   W_q é constante e não afeta o *ranking*.
   $f_{q,t} = 1$ para palavras presentes.
   $f_{q,t} = 0$ para palavras ausentes.

2. Muita memória é necessária
   W_d: 4 bytes por documento.
   A_d: 4 bytes por documento.

3. $r \ll N$
   Passo 4 não precisa ordenar completamente A_d, basta extrair os r maiores valores.

Cálculo do Valor do Cosseno

Memória para W_d:

TREC 2: 740.000 documentos

1. Guardar W em disco e acessar seqüencialmente.

2. Guardar $f_{d,t}/W_d$ ao invés de $f_{d,t}$

   - Problema: se o arquivo invertido é mantido comprimido esta opção é muito cara!

   Logo, passa-se menos de 2 bits por $f_{d,t}$ para 16 ou 32 bits.

Cálculo do Valor do Cosseno

Memória para W_d:

3. Compromisso entre tudo na memória e tudo no disco

   - Armazenar pesos aproximados na memória que possam dirigir o acesso a disco para os valores exatos.

   - 6 a 8 bits de memória principal são suficientes para manter o número de acessos um pouco maior que r.

   - Suponha que b bits serão usados para codificar uma aproximação de x.

   - Suponha que limites inferiores e superiores para que x sejam conhecidos: $L \leq x \leq U$.

   - Usar um conjunto de *"buckets"* que crescem como uma seqüência geométrica:

$$B = \left(\frac{U+\epsilon}{L} \right)^{2^{-b}}$$

Cálculo do Valor do Cosseno

Memória para W_d:

3. (Continuação)

   - Código c para x:

$$f(x) = \lfloor \log_B(x/L) \rfloor = \left\lfloor \frac{\log x/L}{\log B} \right\rfloor$$

   - O intervalo de valores de x que correspondem a um código c:

$$g(c) \leq x \leq g(c+1)$$

   onde,

$$g(c) = L.B^c$$

Cálculo do Valor do Cosseno

Memória para W_d:

3. (Continuação)

   Ex.: para aproximar valores no intervalo $10 \leq x \leq 18$ por b = 2 bits
   $\epsilon = 0.1$
   $B = \left(\frac{18.1}{10}\right)^{2^{-2}} = (1.81)^{1/4} = 1.160$

   Logo, o código para 15.3 seria:

   $c = f(15.3) = \left\lfloor \log_{1.16} \frac{15.3}{10} \right\rfloor = \left\lfloor \frac{\log 15.3}{1.16} \right\rfloor = \lfloor 2.865 \rfloor = 2$

   Logo 10, ou $c = 2$, é o código que é expandido para o intervalo $13.456 \leq x < 15.610$.

Cálculo do Valor do Cosseno

Memória para W_d ($L = 10.0$, $U = 18.0$ e $b = 2$):

Código c	Valor do código	Intervalo corresp. para x	$x = g(c + 0.5)$
00	0	[10.00, 11.60)	10.77
01	1	[11.60, 13.46)	12.49
10	2	[13.46, 15.61)	14.49
11	3	[15.61, 18.01)	16.81

Cálculo do Valor do Cosseno

Como usar no *ranking*:

1. $\hat{W}_d$ para normalizar:

   $\hat{W}_d = g(c_d)$
   $c_d = f(W_d)$ (código de b bits usado para representar W_d)

   Esta aproximação pode alterar a ordem um pouco.

2. Usar pesos aproximados para recuperar W_d do disco e calcular o *ranking* exato:

   Dado c_d de um documento d e $g(c_d)$ é um limite inferior para W_d e fornece limite superior para o cosseno a ser calculado.

$$\Downarrow$$

Este limite pode ser usado para eliminar documentos!

Cálculo do Valor do Cosseno

Para determinar os r documentos de topo usando pesos aproximados

1. **Para** $1 \leq d \leq N$ **faça**
   $A \leftarrow A_d / g(c_d)$.
   A_d é o valor do $cosseno$ aproximado; $A_d \geq cosseno(Q, D_d)$.
2. $H \leftarrow r$ valores de topo de A_d.
3. **Para cada** $d \in H$ **faça**
   (a) Leia W_d do disco.
   (b) Armazene o endereço do documento d.
   (c) $A_d \leftarrow A_d * g(c_d)/W_d$.
   Documentos em H têm os valores do $cosseno$ exatos.
4. **Para** $1 \leq d \leq N$ **faça**
   (a) **Se** $A_d \notin H$ **e** $A_d > min\{H\}$ **então**
   i. Leia W_d do disco.
   ii. $A_d \leftarrow A_d * g(c_d)/W_d$.
   iii. **Se** $A_d > min\{H\}$ **então**
   $H \leftarrow H - \{min\{H\}\} + \{A_d\}$,
   Armazene o endereço do documento d.

   H agora contém os r valores do $cosseno$ exatos do topo.

Cálculo do Valor do Cosseno

A_d: valor para documento d

c_d: código aproximado para W_d

$g(c)$: função inversa para a aproximação

H:

- contém as r respostas
- inicializado com r valores aproximados do cosseno usando limite inferior em W_d
- ao final contém r maiores valores exatos

Cálculo do Valor do Cosseno

Passo 2: Set H $\leftarrow$ top r values of A_d

- Verifica se qualquer dos documentos que não estejam nos top r poderiam pertencer ao conjunto:
  - Compara o menor de H com o limite superior de A_d calculado previamente para d.
  - Se $A_d > min\{H\}$ então lê do disco valor exato (que é menor que valor aproximado).
  - Se valor exato ainda é maior do que $min\{H\}$ considere o documento d.

Ex.: TREC com arranjo de pesos aproximados por 8-bit, os top 25 podem ser identificados recuperando apenas 30 valores exatos.

Tratamento dos Acumuladores A_d

Heurísticas baseadas no fato de que entradas do arquivo invertido são processadas na ordem crescente de f_t (ou decrescente de w_t), como nas consultas booleanas.

Heurística: *ranking* será uma aproximação da regra do cosseno.

Regra simples:

- Limite "a priori" no número de acumuladores > 0 permitidos, parando de adicionar novos valores quando limite é atingido.

- Termos de peso alto são considerados e termos de peso baixo são desconsiderados.

Tratamento dos Acumuladores A_d

Regra simples:

E os termos restantes?

1. *Quit*

   Ignora contribuição dos termos não processados (mesmo se eles são para documentos que já tenham acumuladores processados).

2. *Continue*

   Continua o processamento de entradas do arquivo invertido:

   - documentos ainda sem acumuladores são ignorados.
   - documentos para os quais acumuladores tenham sido criados continuam a ter contribuições para o seu cosseno acrescidas.

Tratamento dos Acumuladores A_d

Regra simples:

1. *Quit*

   - Processamento rápido: entradas não processadas são as mais longas.
   - Resultados pobres de *ranking*.

2. *Continue*

   - *Retrieval performance* comparável ao método exato para o cosseno.

3. Experimentos TREC com conjunto de consultas padrão:

   - 5000 acumuladores são suficientes para extrair os 1000 top documentos.

Ordenação

- Como evitar ordenação?

$$N = 1.000.000$$

$$N \log N = 20 \text{ milhões}$$

$$r \ll N$$

$$\Downarrow$$

Uso de um *heap*

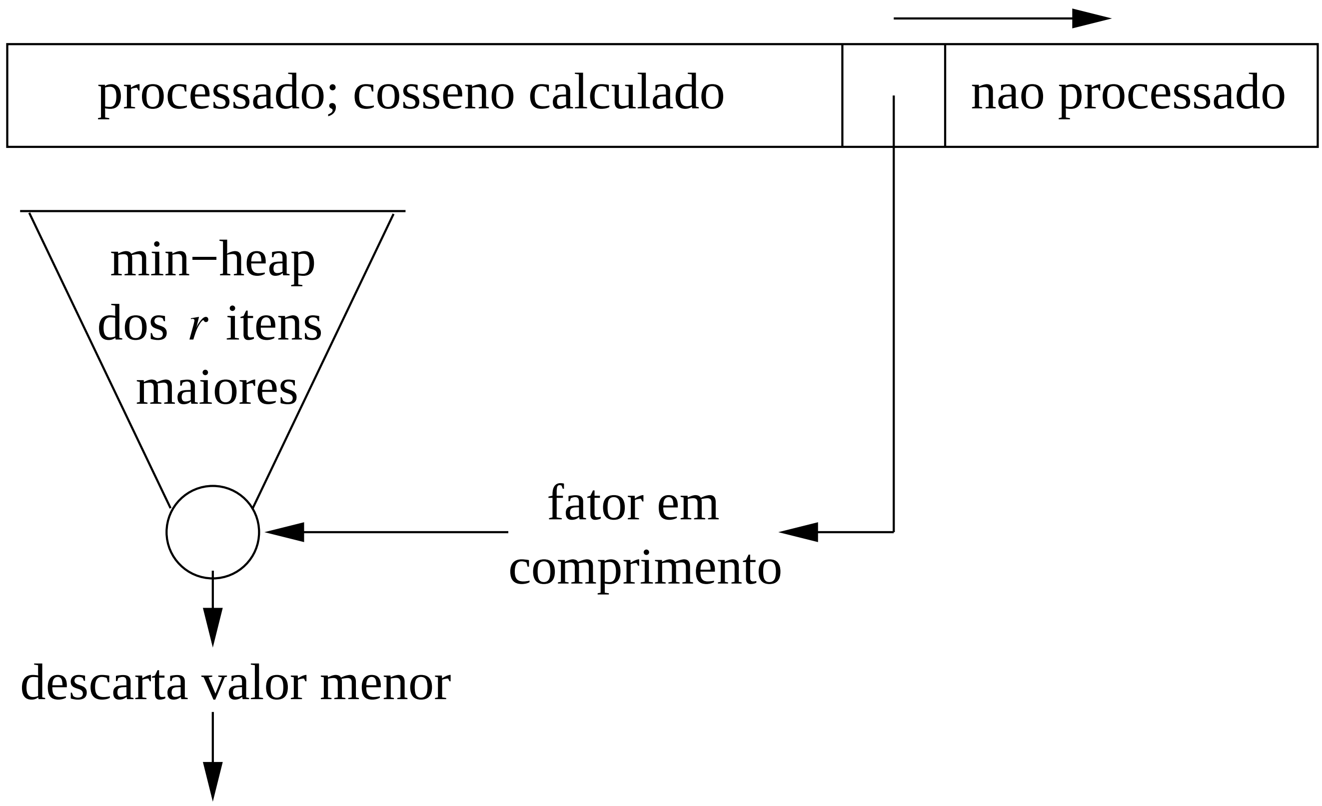
- Construção $\approx 2N$

  - r extrações $\approx 2r \log N$

- Qual o problema de usar um *heap* com N elementos?

  - espaço para armazenar N endereços de documentos.

Ordenação



Solução: armazenar no *heap* r valores (*min-heap*).

Algoritmo:

- inicia com os primeiros r acumuladores (com endereço dos documentos).
  Custo: $2r$

- lê novo acumulador: se maior que o menor valor do *heap* $\Rightarrow$ substituir.

Ordenação

Para selecionar os r valores do cosseno de topo

1. $H \leftarrow \{\}$. H é o *heap*.
2. **Para** $1 \leq d \leq r$ **faça**
   (a) $A_d \leftarrow A_d/W_d$.
   (b) Armazene o endereço do documento d.
   (c) $H \leftarrow H + \{A_d\}$.
3. Construa H no *heap*.
4. **Para** $r + 1 \leq d \leq N$ **faça**
   (a) $A_d \leftarrow A_d/W_d$.
   (b) **Se** $A_d > min\{H\}$ **então**
      i. $H \leftarrow H - min\{H\} + \{A_d\}$.
      ii. "Peneire" H.
      iii. Armazene o endereço do documento d.

   H agora contém os r valores do $cosseno$ exatos do topo.
5. **Para** $1 \leq i \leq r$ **faça**
   (a) Selecione d tal que $A_d = max\{H\}$.
   (b) Recupere o documento d e o apresente para o usuário.
   (c) $H \leftarrow H - \{A_d\}$.

Processamento de Consultas

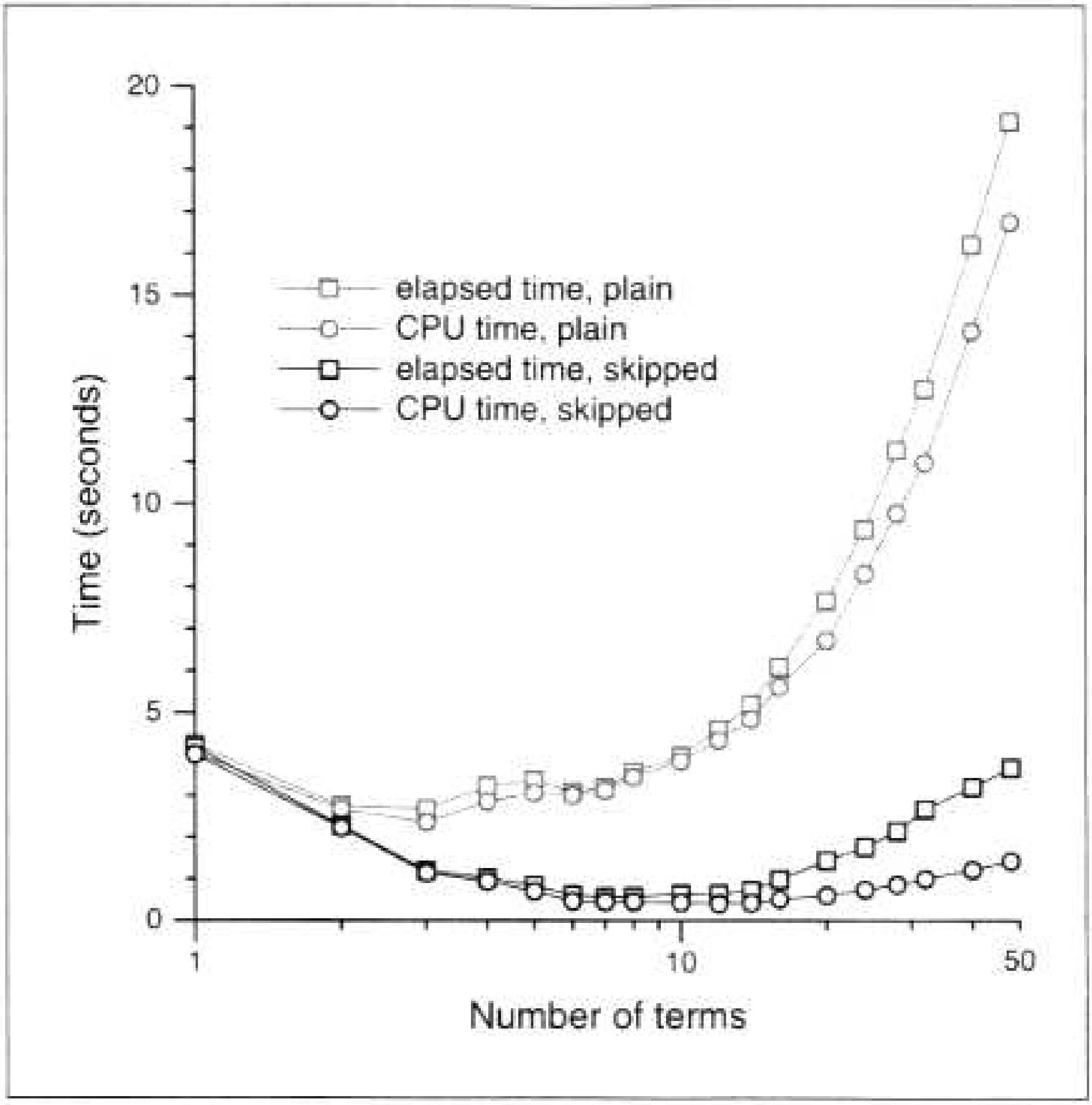


Figure 9.10 Time required by mg for Boolean queries.

Processamento de Consultas

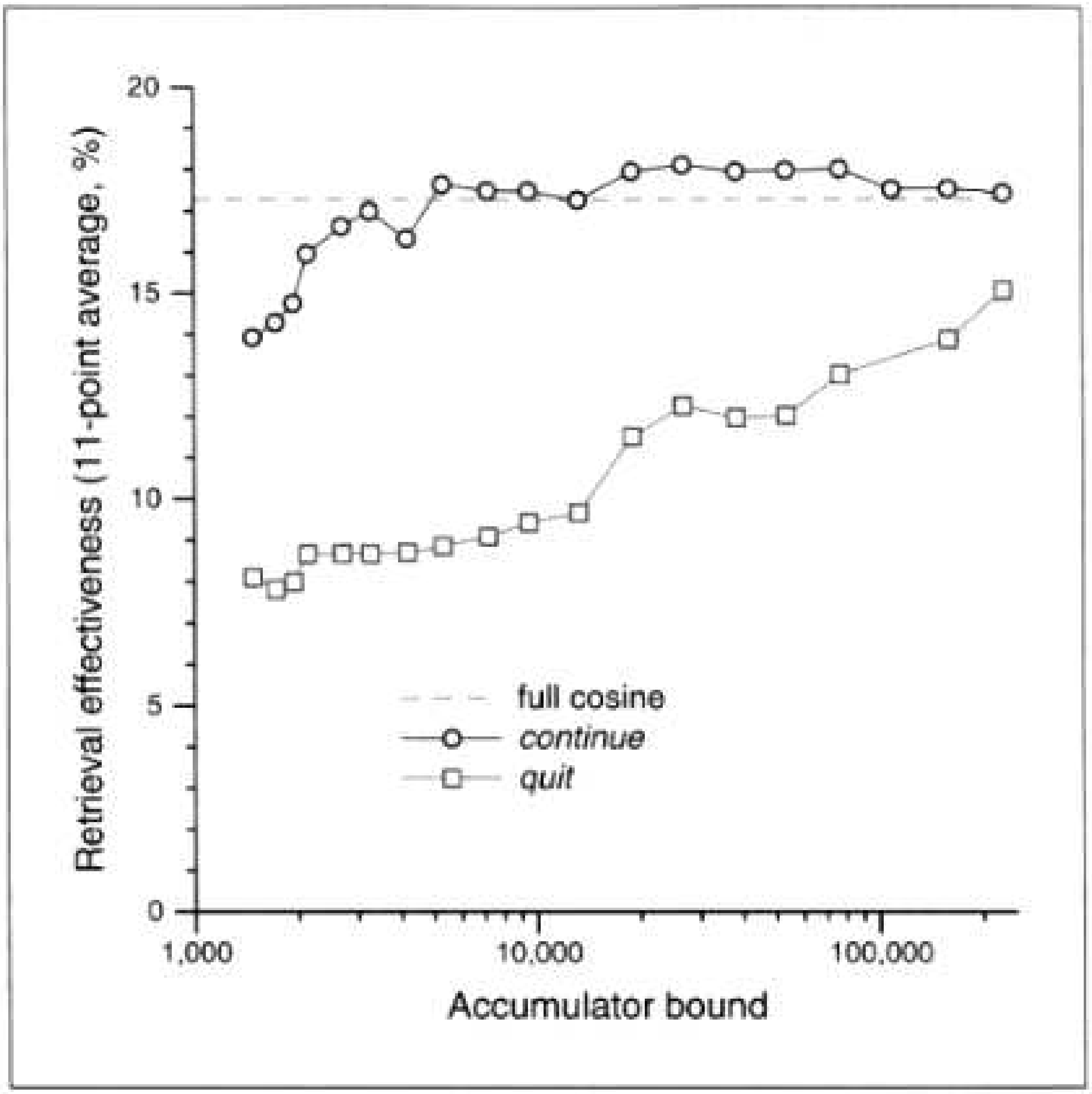


Figure 9.11 Ranking effectiveness: *quit* and *continue*.

Processamento de Consultas

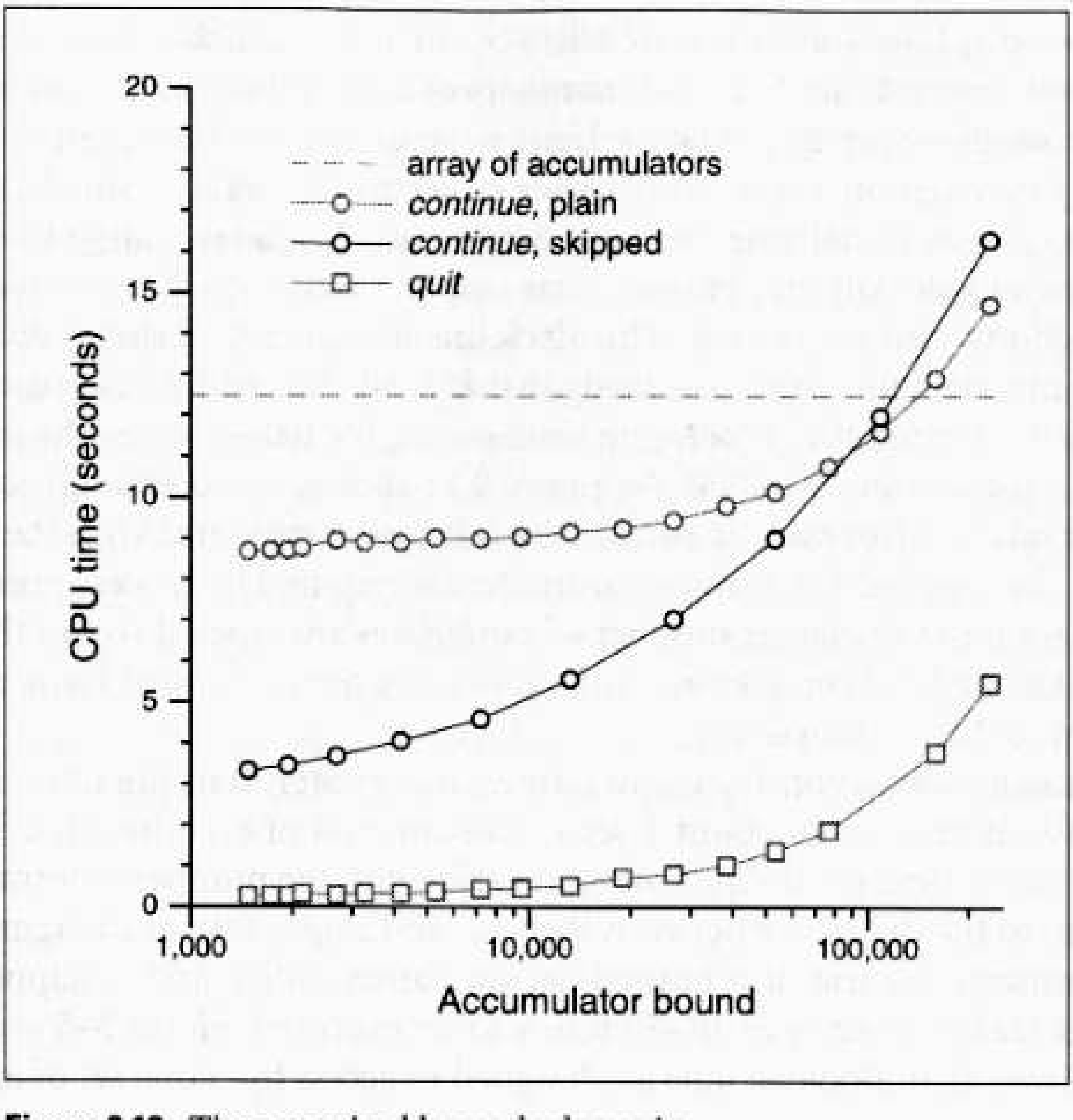


Figure 9.12 Time required by ranked queries.

Medida do Cosseno

Algoritmo Básico:

1. Crie e inicialize uma estrutura de acumuladores (A).
2. **Para cada** termo t na consulta **faça**.
   (a) Recupere lista invertida do termo t do disco.
   (b) **Para cada** par $[d, f_{d,t}]$ na lista invertida t **faça**:
   i. **Se** $A_d \notin A$ **então**
   $A_d = 0$,
   $A = A + \{A_d\}$
   ii. $A_d = A_d + w_{q,t} * w_{d,t}$ $(sim_{q,d,t})$
3. Divida cada acumulador A_d por W_d.
4. Recupere os r maiores valores para os acumuladores e retorne os documentos correspondestes.

Medida do Cosseno

$C_{q,d} = \frac{\sum_t sim_{q,d,t}}{W_d}$

$sim_{q,d,t} = w_{q,t} \times w_{d,t}$
similaridade parcial entre q e d em relação ao termo t

$w_{x,t}$: peso de t na consulta ou no documento

$W_d = \sqrt{\sum_t w_{d,t}^2}$
W_d: comprimento do documento d (pré-computado)

$w_{d,t} = \log_2 f_{d,t} \times \log_2 \frac{N}{f_t}$
Persin, Zobel & Sacks_Davis: JASIS, 1996

$w_{d,t} = (1 + \log_e f_{d,t}) \times \log_e (1 + \frac{N}{f_t})$
WMB, 1999

Medida do Cosseno

Estruturas de Dados para A_d:

- Aloca acumulador para para documento d no início
- Adiciona dinamicamente um acumulador para cada documento d quando $sim_{q,d,t} > 0$

Medida do Cosseno

Custo de avaliar a medida do cosseno pode ser muito alto:

- consultas podem conter muitos termos
- uma medida de similaridade é atribuida a cada documento contendo um dos termos da consulta

Solução:

- técnicas para filtrar documentos durante a ordenação
- acumulador é incrementado somente se a combinação $tf \times idf$ é grande o suficiente para causar alteração na ordem final

Medida do Cosseno

Duas heurísticas adicionais para limitar o número de acumuladores:

1. Estratégia de corte que processa listas invertidas na ordem w_t decrescente, permitindo criar acumuladores enquanto $w_t > max_{t \in Q} \frac{w_t}{3}$
   - Avaliação do restante dos termos não cria mais acumuladores, mas podem contribuir quando já houver A_d criado
2. Processa uma lista invertida apenas se w_t do termo mais todos os outros termos restantes é tal que o documento ranqueado na posição $r+1$ poderia entrar nos r documentos do topo
   - Garante os top r documentos mas não necessariamente na mesma ordem da medida exata do cosseno

Medida do Cosseno

Como avaliar:

1. Pontencialmente caro para grandes coleções

   - tempo CPU
   - largura de banda de disco
   - demanda por memória

2. Soluções

   - limitar o número de documentos candidatos ao ranking
   - ordenar por $f_{d,t}$
   - representação eficiente dos W_d

Objetivo:

- procurar não usar mais memória e tempo de CPU necessários ao processamento da consulta booleana

Medida do Cosseno - Persin

Custos Principais:

- Memória
  - armazena as medidas de similaridade
  - A_d: um acumulador por documento
- Acesso a disco
  - transferência das listas invertidas do disco para a memória
- Tempo CPU
  - para processar informação do índice

Medida do Cosseno - Persin

Limitação do número de candidatos usando Persin, Zobel & Sacks_Davis:

- Lista invertida ordenada por $f_{d,t}$

  $< 5; (1, 2), (2, 2), (3, 5), (4, 1), (5, 2) >$

  $< 5; (3, 5), (1, 2), (2, 2), (5, 2), (4, 1) >$

  outra representação:

  $< 5; (5, 1 : 3), (2, 3 : 1, 2, 5), (1, 1 : 4) >$

  permite usar diferenças nos números dos documentos.

Medida do Cosseno - Persin

Como usar lista invertida ordenada por $f_{d,t}$:

- Normalmente idf é previamente calculado e usado para ordenar os termos da consulta de forma que termos mais raros são processados primeiro
- Entretanto, $f_{d,t}$ também é importante. Logo, uma boa solução é processar todas as listas em paralelo, um bloco de cada vez:
  - primeiro bloco da lista invertida de cada termo da consulta é lido e o produto $(1 + \log_e f_{d,t}) \times (\log_e 1 + \frac{N}{f_t})$ é calculado para cada bloco
  - todos os documentos de cada bloco recebem a mesma contribuição no acumulador (mesmo $f_{d,t}$)
  - processar na ordem decrescente de produto (maiores contribuições primeiro)

Medida do Cosseno - Persin

Como usar lista invertida ordenada por $f_{d,t}$:

Vantagens:

1. Mais precisa
   - cada bloco é processado no momento certo
   - se limite no número de acumuladores é usado, processamento pode usar *quit* ou *continue* quando limite é atingido, e agora sabemos que contribuições desprezadas são ainda menores
2. Menos processamento
   - um item de qualquer lista com $f_{d,t} = 1$ dificilmente será processado
3. Menos transferência de disco
   - listas longas raramente são lidas na sua totalidade
4. Ranking melhor do que *quit* e *continue* com lista invertida ordenada pelo número do documento

Medida do Cosseno - Persin

Como usar lista invertida ordenada por $f_{d,t}$:

Desvantagens:

1. Difícil processar consultas booleanas

Medida do Cosseno - Persin

Como filtar documentos:

Transição gradual da inclusão para a omissão de documentos

1. termos da consulta são ordenados por idf decrescente (como no algoritmo básico)
2. antes de processar cada termo t calcula dois parâmetros: S_{ins} e S_{add}, $S_{add} \leq S_{ins}$

   - S_{add} permite ignorar itens da lista invertida
   - S_{ins} permite ignorar alguns documentos

- Propostas anteriores decidiam processar ou descartar listas invertidas inteiras
- Persin permite decidir processar ou rejeitar documentos individuais

Medida do Cosseno - Persin

Algoritmo:

1. Crie e inicialize uma estrutura de acumuladores (A).
2. Ordene termos da consulta por idf decrescente.
3. $S_{max} = 0$
4. **Para cada** termo t na consulta **faça**.
   (a) Compute os valores para os filtros f_{ins} e f_{add}.
   (b) Recupera lista invertida do disco.
   (c) **Para cada** par $[d, f_{d,t}]$ na lista invertida **faça**.
   i. **Se** $fd, t > f_{ins}$ **então**
   Crie A_d se necessário
   $A_d = A_d + w_{q,t}.w_{d,t}\ (sim_{q,d,t})$
   ii. **Se não**, **Se** $fd, t > f_{add}$ e A_d existe **então**
   $A_d = A_d + w_{q,t}.w_{d,t}\ (sim_{q,d,t})$
   iii. **Se não**, Desvie para o próximo termo da consulta
   iv. $S_{max} = max(S_{max}, A_d)$.
5. Divida cada acumulador A_d por W_d.
6. Recupere os r maiores valores para os acumuladores
   e retorne os documentos correspondestes.

Medida do Cosseno - Persin

Como obter f_{ins} e f_{add}

- São função da similaridade parcial acumulada do documento mais relevante no momento (S_{max})

  $S_{ins} = C_{ins} \times S_{max}, \qquad S_{add} = C_{add} \times S_{max}$
  onde, $0 \leq C_{add} \leq C_{ins}$, não constantes

- $sim_{q,d,t} \leq w_{q,t} \times w_{d,t} = f_{d,t} \times idf \times f_{q,t} \times idf$

  ou,
  $$\frac{sim_{q,d,t}}{f_{q,t} \times idf^2} \leq f_{d,t}$$

  (processa elemento como condição sobre $f_{d,t}$)

  logo, $$f_{ins} = \frac{C_{ins} \times S_{max}}{f_{q,t} \times idf^2}, \quad f_{add} = \frac{C_{add} \times S_{max}}{f_{q,t} \times idf^2}$$

  (Valores de f_{ins} e f_{add} são constantes durante o processamento de uma lista invertida)

Medida do Cosseno - Persin

Observações sobre f_{ins} e f_{add}

- permitem transição suave da aceitação para rejeição de elementos nas listas invertidas
- fica progressivamente mais difícil adicionar ou atualizar os acumuladores
- para primeiros termos o valor de S_{max} é pequeno e o valor do idf é alto
- na medida que S_{max} cresce e idf diminui f_{ins} e f_{add} crescem até que, no limite, todos $f_{d,t} < f_{add}$ (e processar a lista a partir deste ponto não promove mudanças significativas nos acumuladores)

Medida do Cosseno - Persin

Nova versão:

1. Cálculo de f_{ins} e f_{add}, na versão original, ocorre uma vez para cada termo t e permanece igual durante o processamento da lista invertida do termo t
2. Na nova versão, o cálculo de f_{ins} e f_{add} ocorre toda vez que S_{max} é alterado.

Medida do Cosseno - Persin

Algoritmo nova versão:

1. Crie e inicialize uma estrutura de acumuladores (A).
2. $S_{max} = 0$
3. **Para cada** termo t na consulta **faça**.
   (a) Compute os valores para os filtros f_{ins} e f_{add}.
   (b) Recupere primeiro bloco de cada lista do disco.
   (c) Calcule $sim_{q,d_s,t}$ e Insere $sim_{q,d_s,t}$ em H.
4. **Enquanto** houver blocos em H **faça**.
   (a) Extrai bloco de H e insere próximo bloco da lista.
   (b) Calcule $sim_{q,d_s,t}$.
      i. **Se** $fd,t > f_{ins}$ **então**
         Crie A_d se necessário
         $A_d = A_d + w_{q,t}.w_{d,t}\ (sim_{q,d,t})$
      ii. **Se não**, **Se** $fd,t > f_{add}$ e A_d existe **então**
         $A_d = A_d + w_{q,t}.w_{d,t}\ (sim_{q,d,t})$
      iii. **Se** $A_d > S_{max}$ **então**
         $S_{max} = max(S_{max}, A_d)$.
         Compute os valores para os filtros f_{ins} e f_{add}.
5. Divida cada acumulador A_d por W_d.
6. Recupere os r maiores valores para os acumuladores e retorne os documentos correspondentes.

Medida do Cosseno - Persin

Exemplo:

$$
\begin{array}{lll}
t_1 & \rightarrow & < 4; (4, 2 : 1, 3), (2, 1 : 2), (1, 1 : 5) > \\
t_2 & \rightarrow & < 1; (2 : 1, 5) > \\
t_3 & \rightarrow & < 2; (8, 1 : 7), (4, 1 : 6) >
\end{array}
$$

$$w_{d,t} = tf \times idf = \log_2 f_{d,t} \times \log_2 \frac{N}{f_t}$$

$N = 8$

$idf_{t_1} = \log_2 \frac{8}{4} = 1$
$idf_{t_2} = \log_2 \frac{8}{1} = 3$
$idf_{t_3} = \log_2 \frac{8}{2} = 2$

$tf \times idf$:

$Bloco_{1,t_1} = \log_2 4 \times 1 = 2$
$Bloco_{1,t_2} = \log_2 2 \times 3 = 3$
$Bloco_{1,t_3} = \log_2 8 \times 2 = 6$
$Bloco_{2,t_1} = \log_2 2 \times 1 = 1$
$Bloco_{2,t_3} = \log_2 4 \times 2 = 4$
$Bloco_{3,t_1} = \log_2 1 \times 2 = 0$

Medida do Cosseno - Persin

Exemplo:

Ordem de processamento dos blocos:

1. Persin Original: (t_2, t_3 e t_1)
   (2,1:5), (8,1:7), (4,1:6), (4,2:1,3), (2,1:2), (1,1:5)

2. Nova versão:
   (8,1:7), (4,1:6), (2,1:5), (4,2:1,3), (2,1:5)