GSI024
Organização e Recuperação de Informação
Ranking: avaliação e implementação

Marcelo K. A.

Faculdade de Computação - UFU

8 de Outubro de 2015

Resumo

- A importância de ranking: estudos com usuários no Google

Resumo

- A importância de ranking: estudos com usuários no Google

Resumo

- A importância de ranking: estudos com usuários no Google
- Normalização de comprimento: normalização com pivot

Resumo

- A importância de ranking: estudos com usuários no Google
- Normalização de comprimento: normalização com pivot
- Implementação de ranking

Resumo

- A importância de ranking: estudos com usuários no Google
- Normalização de comprimento: normalização com pivot
- Implementação de ranking
- O sistema de busca completo

Por que ranking é tão importante?

- Última aulas: problemas com recuperação sem ranking

Por que ranking é tão importante?

- Última aulas: problemas com recuperação sem ranking
  - Usuário querem olhar apenas alguns resultados: não milhares

Por que ranking é tão importante?

- Última aulas: problemas com recuperação sem ranking
  - Usuário querem olhar apenas alguns resultados: não milhares
  - É muito difícil escrever consultas para produzir poucos resultados

Por que ranking é tão importante?

- Última aulas: problemas com recuperação sem ranking
  - Usuário querem olhar apenas alguns resultados: não milhares
  - É muito difícil escrever consultas para produzir poucos resultados
  - $\rightarrow$ Ranking é importante porque reduz um grande número de resultados para um número pequeno de resultados

Por que ranking é tão importante?

- Última aulas: problemas com recuperação sem ranking
  - Usuário querem olhar apenas alguns resultados: não milhares
  - É muito difícil escrever consultas para produzir poucos resultados
  - $\rightarrow$ Ranking é importante porque reduz um grande número de resultados para um número pequeno de resultados
- A seguir: mais dados sobre "usuários olham poucos resultados"

Por que ranking é tão importante?

- Última aulas: problemas com recuperação sem ranking
  - Usuário querem olhar apenas alguns resultados: não milhares
  - É muito difícil escrever consultas para produzir poucos resultados
  - $\rightarrow$ Ranking é importante porque reduz um grande número de resultados para um número pequeno de resultados
- A seguir: mais dados sobre "usuários olham poucos resultados"
- De fato, na maioria dos casos, examinam 1, 2, ou 3 resultados

Investigação empírica do efeito de ranking

- Como podemos medir a importância do ranking?

Investigação empírica do efeito de ranking

- Como podemos medir a importância do ranking?
- Pode-se usar em um ambiente controlado:

Investigação empírica do efeito de ranking

- Como podemos medir a importância do ranking?
- Pode-se usar em um ambiente controlado:
  - Filmagem

Investigação empírica do efeito de ranking

- Como podemos medir a importância do ranking?
- Pode-se usar em um ambiente controlado:
  - Filmagem
  - Pedir para usuário "pense em voz alta"

Investigação empírica do efeito de ranking

- Como podemos medir a importância do ranking?
- Pode-se usar em um ambiente controlado:
  - Filmagem
  - Pedir para usuário "pense em voz alta"
  - Entrevista

Investigação empírica do efeito de ranking

- Como podemos medir a importância do ranking?
- Pode-se usar em um ambiente controlado:
  - Filmagem
  - Pedir para usuário "pense em voz alta"
  - Entrevista
  - Seguir movimento ocular (eye-tracking)

Investigação empírica do efeito de ranking

- Como podemos medir a importância do ranking?
- Pode-se usar em um ambiente controlado:
  - Filmagem
  - Pedir para usuário "pense em voz alta"
  - Entrevista
  - Seguir movimento ocular (eye-tracking)
  - Medir tempos de ações

Investigação empírica do efeito de ranking

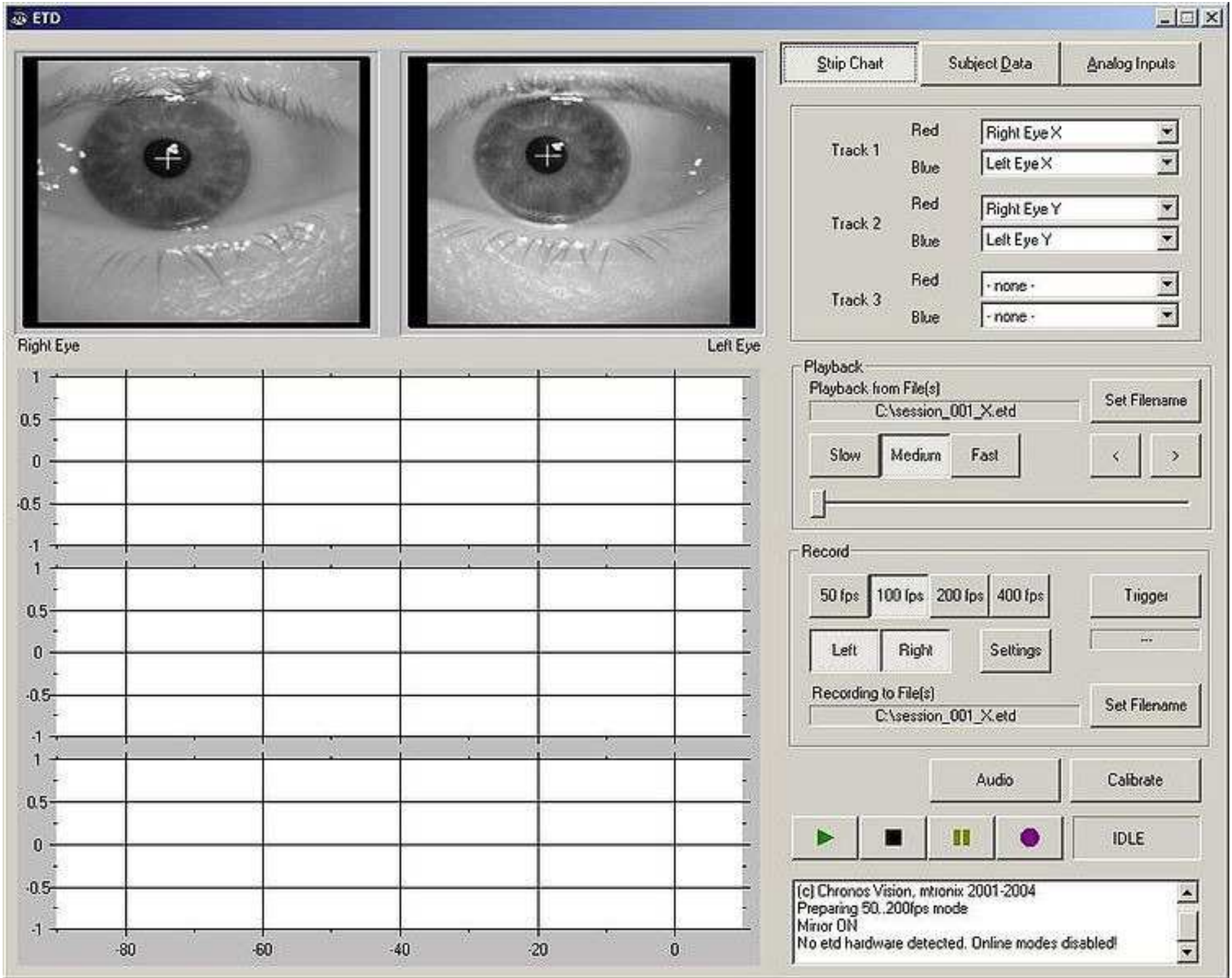
- Como podemos medir a importância do ranking?
- Pode-se usar em um ambiente controlado:
  - Filmagem
  - Pedir para usuário "pense em voz alta"
  - Entrevista
  - Seguir movimento ocular (eye-tracking)
  - Medir tempos de ações
  - Contabilizar cliques

Investigação empírica do efeito de ranking

- Como podemos medir a importância do ranking?
- Pode-se usar em um ambiente controlado:
  - Filmagem
  - Pedir para usuário "pense em voz alta"
  - Entrevista
  - Seguir movimento ocular (eye-tracking)
  - Medir tempos de ações
  - Contabilizar cliques
- Os slides a seguir são de uma apresentação de Dan Russell

Investigação empírica do efeito de ranking

- Como podemos medir a importância do ranking?
- Pode-se usar em um ambiente controlado:
  - Filmagem
  - Pedir para usuário "pense em voz alta"
  - Entrevista
  - Seguir movimento ocular (eye-tracking)
  - Medir tempos de ações
  - Contabilizar cliques
- Os slides a seguir são de uma apresentação de Dan Russell
- Dan Russell é o "Líder de tecnologia para qualidade de busca" no Google


ETD
Right Eye
Left Eye
Strip Chart
Subject Data
Analog Inputs
Track 1
Red
Right Eye X
Blue
Left Eye X
Track 2
Red
Right Eye Y
Blue
Left Eye Y
Track 3
Red
- none -
Blue
- none -
Playback
Playback from File(s)
C:\session_001_X.etd
Set Filename
Slow
Medium
Fast
<
>
Record
50 fps
100 fps
200 fps
400 fps
Trigger
Left
Right
Settings
Recording to File(s)
C:\session_001_X.etd
Set Filename
Audio
Calibrate
IDLE
(c) Chronos Vision, mtronix 2001-2004
Preparing 50..200fps mode
Mirror ON
No etd hardware detected. Online modes disabled!
1
0.5
0
-0.5
-1
-80
-60
-40
-20
0

So.. Did you notice the FTD official site?

To be honest, I didn't even look at that.

At first I saw "from $20" and $20 is what I was looking for.

To be honest, 1800-flowers is what I'm familiar with and why I went there next even though I kind of assumed they wouldn't have $20 flowers

And you knew they were expensive?

I knew they were expensive but I thought "hey, maybe they've got some flowers for under $20 here…"

But you didn't notice the FTD?

No I didn't, actually… that's really funny.

Interview video

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.

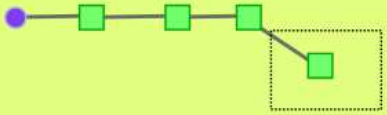
Google

Kinds of behaviors we see in the data

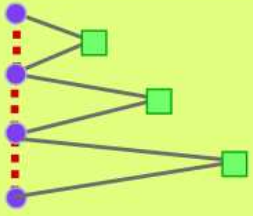
Short / Nav

Topic exploration

Topic switch

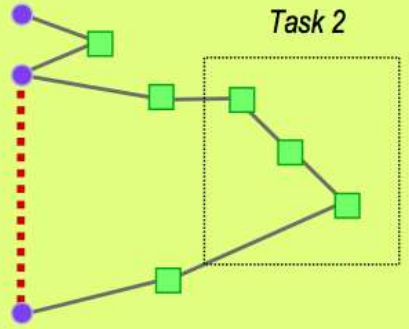


Methodical results exploration

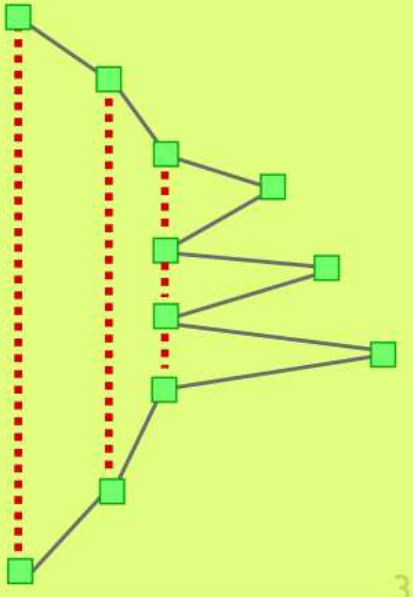


Query reform

Multitasking

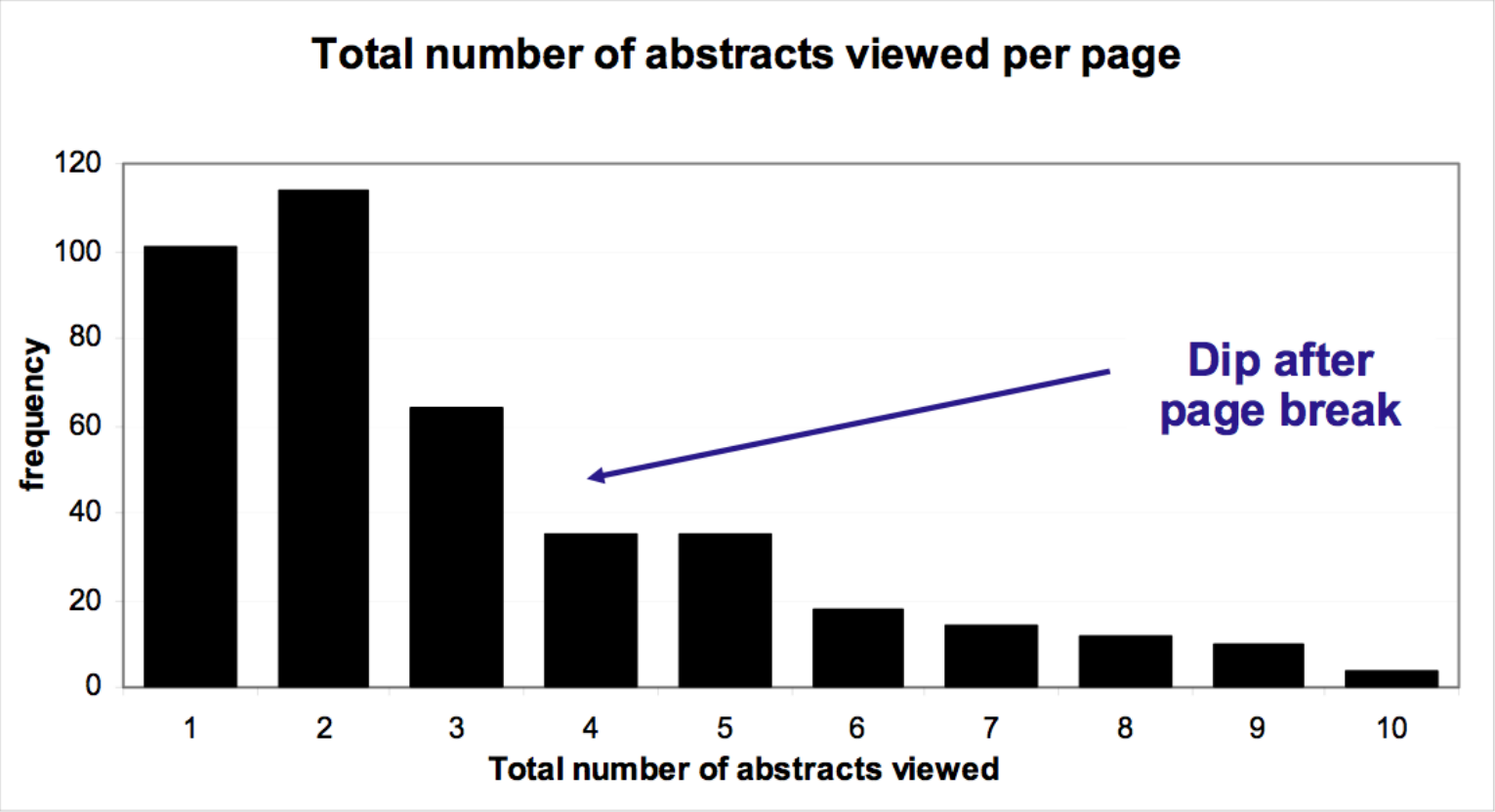


Stacking behavior



Google

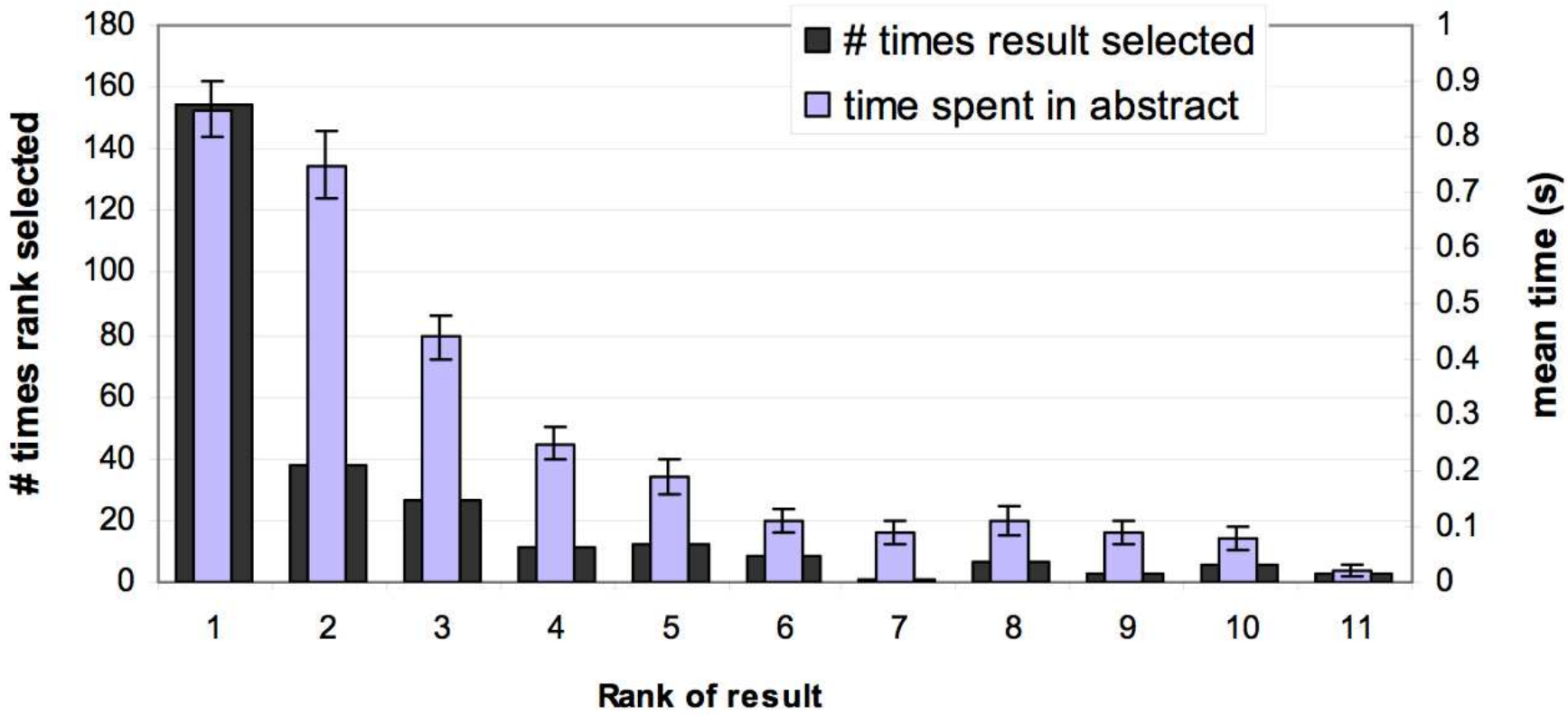
How many links do users view?



Mean: 3.07 Median/Mode: 2.00

Google

Looking vs. Clicking

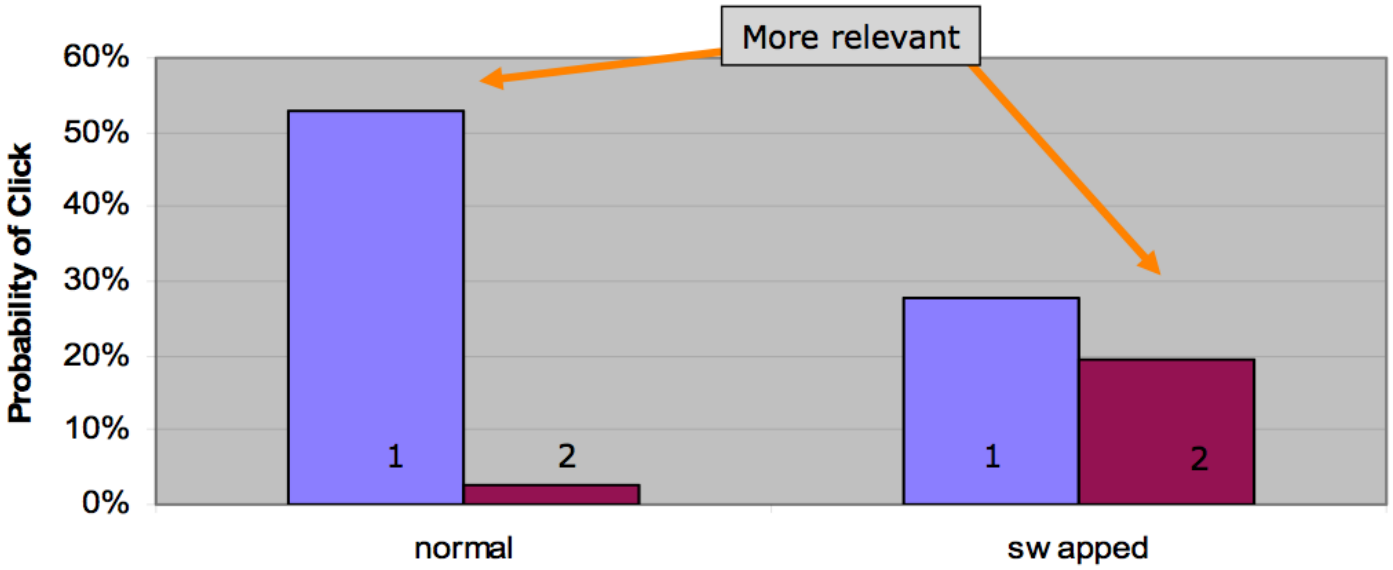


- Users view results one and two more often / thoroughly
- Users click most frequently on result one

Google

Presentation bias – reversed results

- Order of presentation influences where users look **AND** where they click



Google

Importância de ranking

- **Lendo resumos:** usuário tendem a ler resumos de páginas melhor posicionadas no ranking (1,2,3,4)

Importância de ranking

- **Lendo resumos:** usuário tendem a ler resumos de páginas melhor posicionadas no ranking (1,2,3,4)
- **Cliques:** distribuição é ainda mais tendenciosa para cliques

Importância de ranking

- **Lendo resumos:** usuário tendem a ler resumos de páginas melhor posicionadas no ranking (1,2,3,4)
- **Cliques:** distribuição é ainda mais tendenciosa para cliques
- Em metade dos casos, usuários clicam na página no topo do ranking

Importância de ranking

- **Lendo resumos:** usuário tendem a ler resumos de páginas melhor posicionadas no ranking (1,2,3,4)
- **Cliques:** distribuição é ainda mais tendenciosa para cliques
- Em metade dos casos, usuários clicam na página no topo do ranking
- Mesmo se a página melhor posicionada não for relevante, 30% dos usuários clicam nela

Importância de ranking

- **Lendo resumos:** usuário tendem a ler resumos de páginas melhor posicionadas no ranking (1,2,3,4)
- **Cliques:** distribuição é ainda mais tendenciosa para cliques
- Em metade dos casos, usuários clicam na página no topo do ranking
- Mesmo se a página melhor posicionada não for relevante, 30% dos usuários clicam nela
- $\rightarrow$ Fazer um bom ranking é muito importante

Importância de ranking

- **Lendo resumos:** usuário tendem a ler resumos de páginas melhor posicionadas no ranking (1,2,3,4)
- **Cliques:** distribuição é ainda mais tendenciosa para cliques
- Em metade dos casos, usuários clicam na página no topo do ranking
- Mesmo se a página melhor posicionada não for relevante, 30% dos usuários clicam nela
- $\rightarrow$ Fazer um bom ranking é muito importante
- $\rightarrow$ Acertar a página no topo do ranking é ainda mais importante

Importância de ranking

- **Lendo resumos:** usuário tendem a ler resumos de páginas melhor posicionadas no ranking (1,2,3,4)
- **Cliques:** distribuição é ainda mais tendenciosa para cliques
- Em metade dos casos, usuários clicam na página no topo do ranking
- Mesmo se a página melhor posicionada não for relevante, 30% dos usuários clicam nela
- $\rightarrow$ Fazer um bom ranking é muito importante
- $\rightarrow$ Acertar a página no topo do ranking é ainda mais importante

Importância de ranking

- **Lendo resumos:** usuário tendem a ler resumos de páginas melhor posicionadas no ranking (1,2,3,4)
- **Cliques:** distribuição é ainda mais tendenciosa para cliques
- Em metade dos casos, usuários clicam na página no topo do ranking
- Mesmo se a página melhor posicionada não for relevante, 30% dos usuários clicam nela
- $\rightarrow$ Fazer um bom ranking é muito importante
- $\rightarrow$ Acertar a página no topo do ranking é ainda mais importante

Mais sobre similaridade cosseno

- Vimos que usar distância euclidiana não é boa ideia
- Usamos similaridade pelo cosseno dos vetores
- Porém, há dificuldades quando documentos têm tamanho variado: usar normalização

Exercício: um problema para normalização cosseno

- Consulta q: "regras anti-doping Beijing 2008 olimpíadas"

Exercício: um problema para normalização cosseno

- Consulta q: "regras anti-doping Beijing 2008 olimpíadas"
- Compare os três seguintes documentos

Exercício: um problema para normalização cosseno

- Consulta q: "regras anti-doping Beijing 2008 olimpíadas"
- Compare os três seguintes documentos
  - d_1: um documento curto em regras de anti-doping na Olímpiadas de 2008

Exercício: um problema para normalização cosseno

- Consulta q: "regras anti-doping Beijing 2008 olimpíadas"
- Compare os três seguintes documentos
  - d_1: um documento curto em regras de anti-doping na Olímpiadas de 2008
  - d_2: um documento longo que consiste de uma cópia de d_1 e outras 5 notícias sobre Olimpíadas e anti-doping

Exercício: um problema para normalização cosseno

- Consulta q: "regras anti-doping Beijing 2008 olimpíadas"
- Compare os três seguintes documentos
  - d_1: um documento curto em regras de anti-doping na Olímpiadas de 2008
  - d_2: um documento longo que consiste de uma cópia de d_1 e outras 5 notícias sobre Olimpíadas e anti-doping
  - d_3: um documento curto nas regras de anti-doping nas Olimpíadas de Atenas 2004

Exercício: um problema para normalização cosseno

- Consulta q: "regras anti-doping Beijing 2008 olimpíadas"
- Compare os três seguintes documentos
  - d_1: um documento curto em regras de anti-doping na Olímpiadas de 2008
  - d_2: um documento longo que consiste de uma cópia de d_1 e outras 5 notícias sobre Olimpíadas e anti-doping
  - d_3: um documento curto nas regras de anti-doping nas Olimpíadas de Atenas 2004
- Qual ranking nós esperamos no modelo de espaço vetorial?

Exercício: um problema para normalização cosseno

- Consulta q: "regras anti-doping Beijing 2008 olimpíadas"
- Compare os três seguintes documentos
  - d_1: um documento curto em regras de anti-doping na Olímpiadas de 2008
  - d_2: um documento longo que consiste de uma cópia de d_1 e outras 5 notícias sobre Olimpíadas e anti-doping
  - d_3: um documento curto nas regras de anti-doping nas Olimpíadas de Atenas 2004
- Qual ranking nós esperamos no modelo de espaço vetorial?
- É provável que d_2 seja posicionado abaixo de d_3 ...

Exercício: um problema para normalização cosseno

- Consulta q: "regras anti-doping Beijing 2008 olimpíadas"
- Compare os três seguintes documentos
  - d_1: um documento curto em regras de anti-doping na Olímpiadas de 2008
  - d_2: um documento longo que consiste de uma cópia de d_1 e outras 5 notícias sobre Olimpíadas e anti-doping
  - d_3: um documento curto nas regras de anti-doping nas Olimpíadas de Atenas 2004
- Qual ranking nós esperamos no modelo de espaço vetorial?
- É provável que d_2 seja posicionado abaixo de d_3 . . .
- . . . mas d_2 é mais relevante que d_3.

Exercício: um problema para normalização cosseno

- Consulta q: "regras anti-doping Beijing 2008 olimpíadas"
- Compare os três seguintes documentos
  - d_1: um documento curto em regras de anti-doping na Olímpiadas de 2008
  - d_2: um documento longo que consiste de uma cópia de d_1 e outras 5 notícias sobre Olimpíadas e anti-doping
  - d_3: um documento curto nas regras de anti-doping nas Olimpíadas de Atenas 2004
- Qual ranking nós esperamos no modelo de espaço vetorial?
- É provável que d_2 seja posicionado abaixo de d_3 . . .
- . . . mas d_2 é mais relevante que d_3.
- O que podemos fazemos sobre isso?

Normalização com pivot

- Normalização cosseno produz pesos que são **altos para documentos curtos** e **baixos para documentos longos** (em média).

Normalização com pivot

- Normalização cosseno produz pesos que são **altos para documentos curtos** e **baixos para documentos longos** (em média).
- Ajustar normalização cosseno por ajuste linear : "mudando" a normalização média em relação ao **pivot**

Normalização com pivot

- Normalização cosseno produz pesos que são **altos para documentos curtos** e **baixos para documentos longos** (em média).
- Ajustar normalização cosseno por ajuste linear : "mudando" a normalização média em relação ao **pivot**
- Efeito: similaridades de documentos curtos com busca dimimui; similaridades de documentos longos com consulta aumenta.

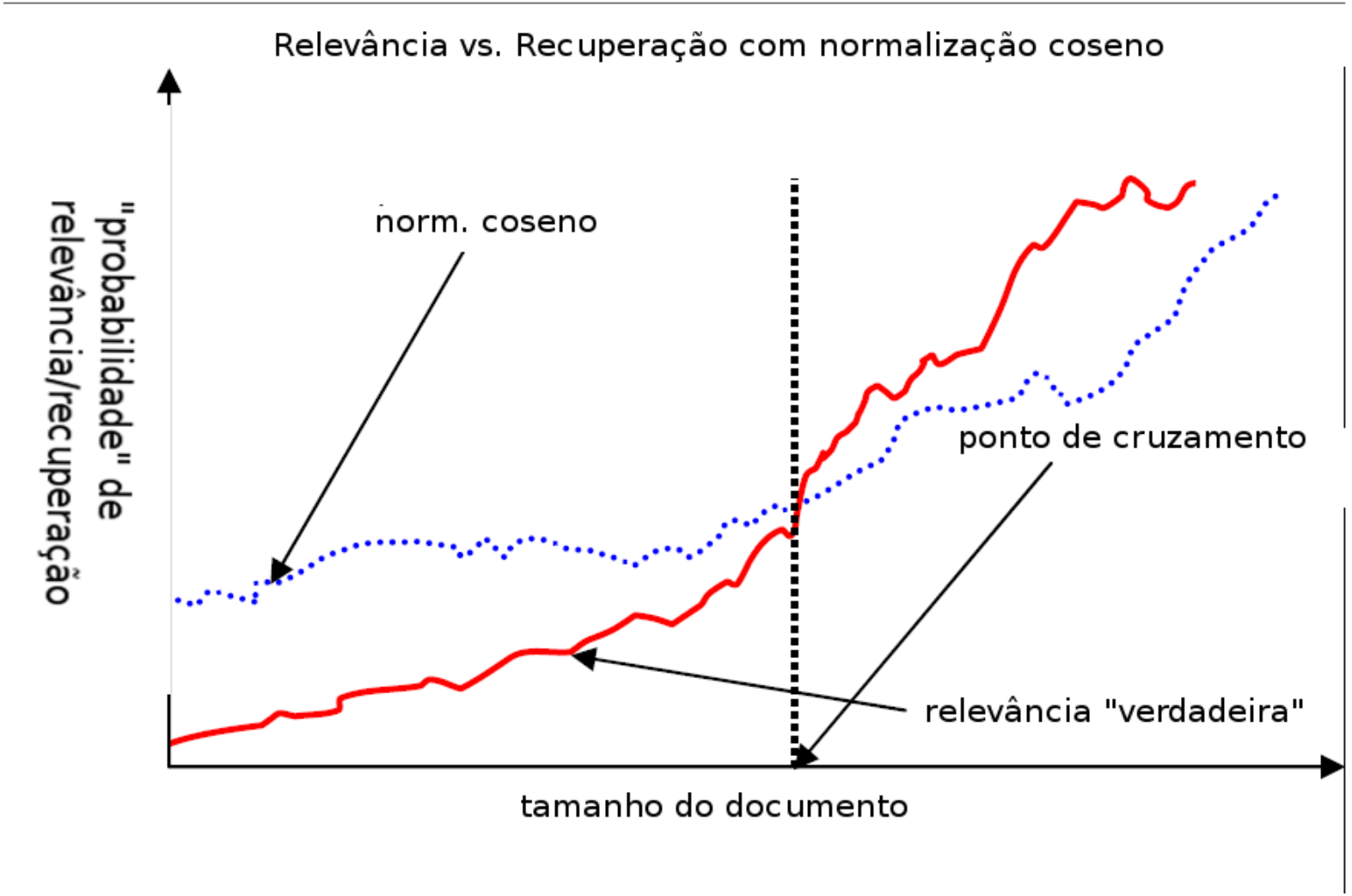
Normalização com pivot

- Normalização cosseno produz pesos que são **altos para documentos curtos** e **baixos para documentos longos** (em média).
- Ajustar normalização cosseno por ajuste linear : "mudando" a normalização média em relação ao **pivot**
- Efeito: similaridades de documentos curtos com busca dimimui; similaridades de documentos longos com consulta aumenta.
- Isto remove a vantagem injusta que documentos curtos tem.

Normalização com pivot

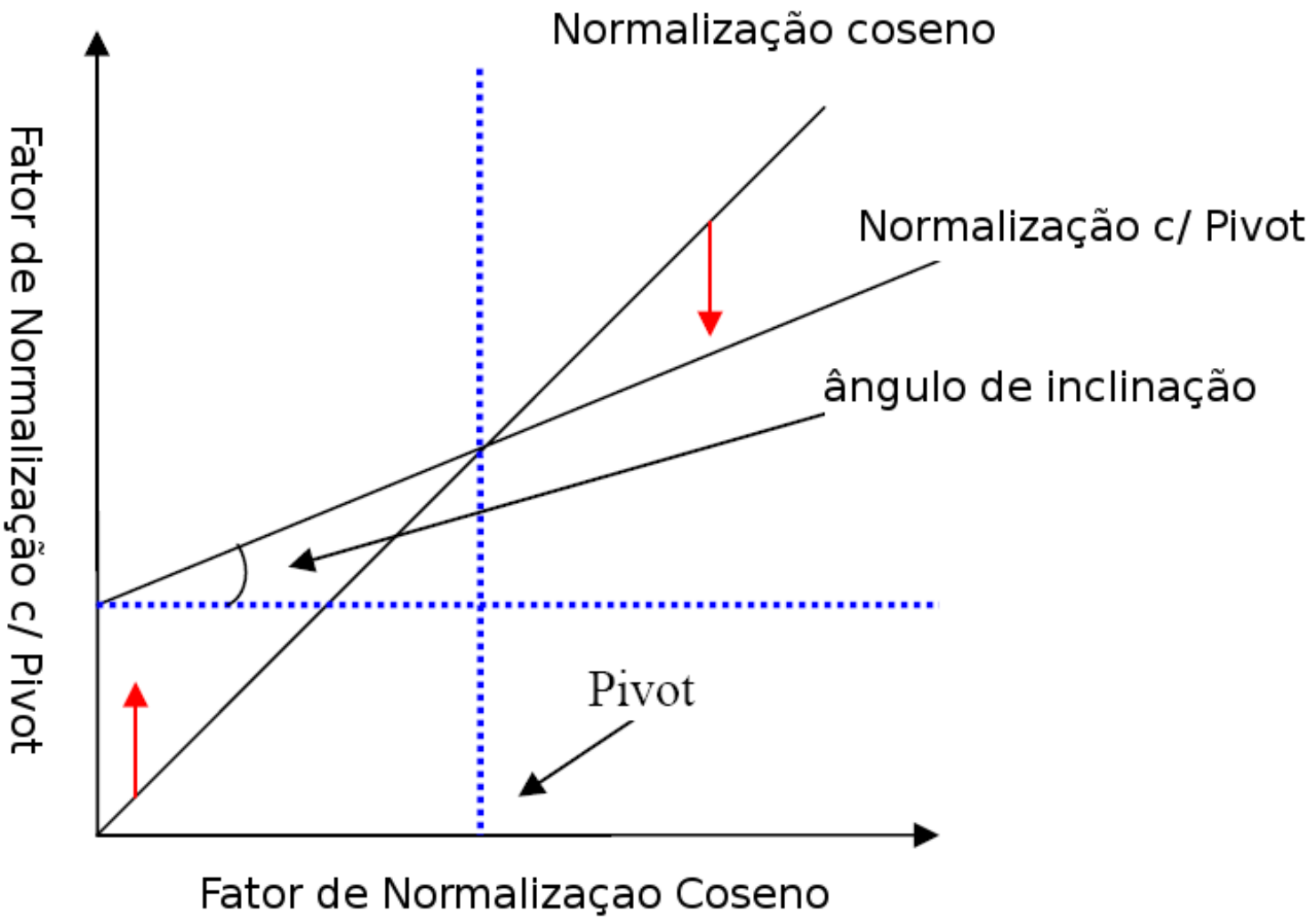
- Normalização cosseno produz pesos que são **altos para documentos curtos** e **baixos para documentos longos** (em média).
- Ajustar normalização cosseno por ajuste linear : "mudando" a normalização média em relação ao **pivot**
- Efeito: similaridades de documentos curtos com busca dimimui; similaridades de documentos longos com consulta aumenta.
- Isto remove a vantagem injusta que documentos curtos tem.
- Note que pontação normalizada com "pivot" não são mais limitadas entre 0 e 1.

Relevância predita e sua probabilidade verdadeira



fonte:
Lillian Lee

Normalização com pivot



fonte:
Lillian Lee

Normalização com pivot: experimentos de Amit Singhal

$$\text{Termo de normalização} = a|V(d)| + (1-a)pivot$$

a: tangente do ângulo de inclinação entre curva "verdadeira" e curva com norm. cosseno
$|V(d)|$: tamanho euclidiano do documento
pivot: valor da normalização cosseno no ponto de intersecção

Cosine	Pivoted Cosine Normalization				
	Slope				
	0.60	0.65	0.70	**0.75**	0.80
6,526	6,342	6,458	6,574	**6,629**	6,671
0.2840	0.3024	0.3097	0.3144	**0.3171**	0.3162
Improvement	+ 6.5%	+ 9.0%	+10.7%	**+11.7%**	+11.3%

Figura: Experimentos de Amit Singhal: chefe do time de ranking do Google

Como encontrar o pivot?

- Ordenar documentos pelo tamanho (bytes)

Como encontrar o pivot?

- Ordenar documentos pelo tamanho (bytes)
- Montar "baldes" de mesmo número de documentos de tamanho similar

Como encontrar o pivot?

- Ordenar documentos pelo tamanho (bytes)
- Montar "baldes" de mesmo número de documentos de tamanho similar
- Usar informações de usuários para identificar pares de "consulta-documento relevante"

Como encontrar o pivot?

- Ordenar documentos pelo tamanho (bytes)
- Montar "baldes" de mesmo número de documentos de tamanho similar
- Usar informações de usuários para identificar pares de "consulta-documento relevante"
- Usar pares "consulta-documento relevante" e contar número de documentos relevantes em cada "balde"

Como encontrar o pivot?

- Ordenar documentos pelo tamanho (bytes)
- Montar "baldes" de mesmo número de documentos de tamanho similar
- Usar informações de usuários para identificar pares de "consulta-documento relevante"
- Usar pares "consulta-documento relevante" e contar número de documentos relevantes em cada "balde"
- Obter $P(\text{balde}(i)|Rel)$ probabilidade de relevância de documentos do balde i pela divisão de contagens de documentos relevantes do balde i pelo total de todos os baldes

Como encontrar o pivot?

- Ordenar documentos pelo tamanho (bytes)
- Montar "baldes" de mesmo número de documentos de tamanho similar
- Usar informações de usuários para identificar pares de "consulta-documento relevante"
- Usar pares "consulta-documento relevante" e contar número de documentos relevantes em cada "balde"
- Obter $P(\text{balde}(i)|Rel)$ probabilidade de relevância de documentos do balde i pela divisão de contagens de documentos relevantes do balde i pelo total de todos os baldes
- Obter $P(\text{balde}(i)|Rec)$ probabilidade de recuperação de documentos do balde i contando o número de vezes que seus documentos aparecem nos 1000 resultados de recuperação

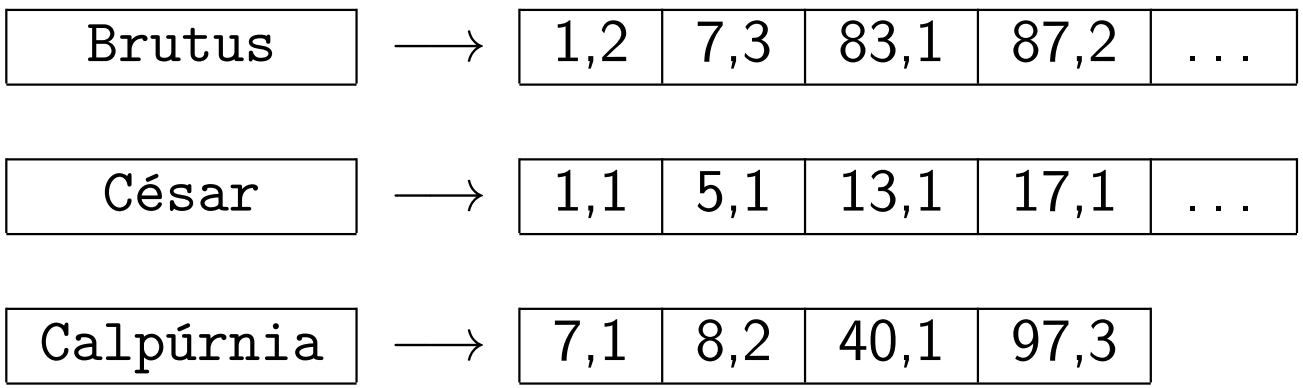
Como encontrar o pivot?

- Ordenar documentos pelo tamanho (bytes)
- Montar "baldes" de mesmo número de documentos de tamanho similar
- Usar informações de usuários para identificar pares de "consulta-documento relevante"
- Usar pares "consulta-documento relevante" e contar número de documentos relevantes em cada "balde"
- Obter $P(\text{balde}(i)|Rel)$ probabilidade de relevância de documentos do balde i pela divisão de contagens de documentos relevantes do balde i pelo total de todos os baldes
- Obter $P(\text{balde}(i)|Rec)$ probabilidade de recuperação de documentos do balde i contando o número de vezes que seus documentos aparecem nos 1000 resultados de recuperação
- O pivot é representado pelo tamanho em que $P(\text{balde}(i)|Rec) \approx P(\text{balde}(i)|Rel)$

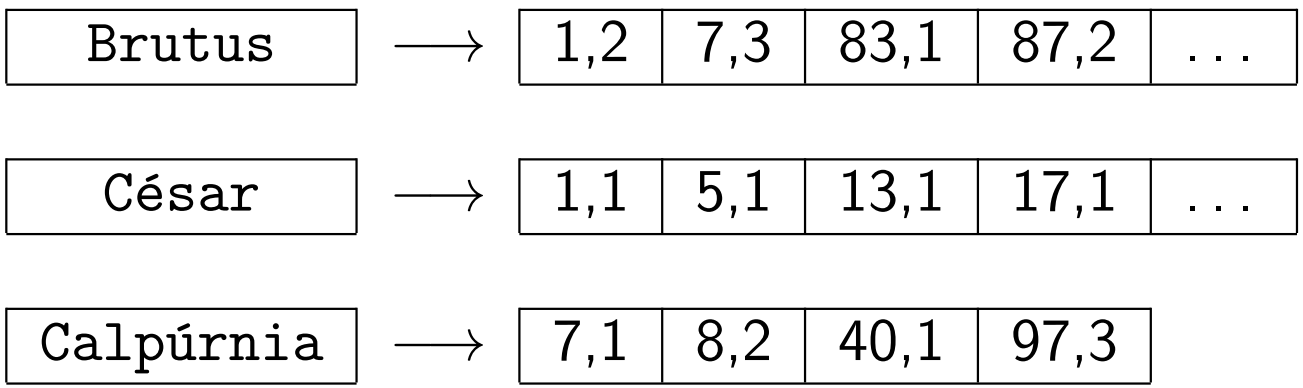
Implementação de ranking

- O que muda no índice invertido?
- Como obter os K documentos mais relevantes?
- Como reduzir o custo de fazer ranking?
- Alternativas de construção de ranking

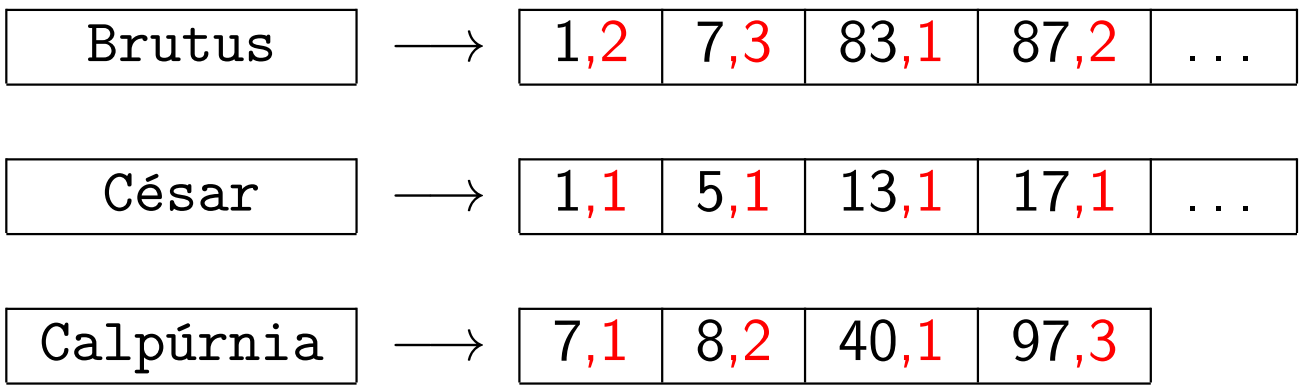
Também precisamos da frequência de termos no índice



Também precisamos da frequência de termos no índice



Também precisamos da frequência de termos no índice



frequência de termos

Frequência de termos no índice invertido

- Em cada lista de referências, armazenar também $\mathrm{tf}_{t,d}$ junto ao docID d

Frequência de termos no índice invertido

- Em cada lista de referências, armazenar também $\mathrm{tf}_{t,d}$ junto ao docID d
- Usar frequência inteira e não valor log ...

Frequência de termos no índice invertido

- Em cada lista de referências, armazenar também $\mathrm{tf}_{t,d}$ junto ao docID d
- Usar frequência inteira e não valor log . . .
- . . . porque valores reais são difíceis de comprimir.

Como calculamos os top k documentos no ranking?

- Em muitas aplicações, não precisamos de um ranking completo

Como calculamos os top k documentos no ranking?

- Em muitas aplicações, não precisamos de um ranking completo
- Precisamos somente dos k melhores para um k pequeno (e.g., $k = 100$).

Como calculamos os top k documentos no ranking?

- Em muitas aplicações, não precisamos de um ranking completo
- Precisamos somente dos k melhores para um k pequeno (e.g., $k = 100$).
- Se não precisamos do ranking completo, existe jeito eficiente de calcular os top k?

Como calculamos os top k documentos no ranking?

- Em muitas aplicações, não precisamos de um ranking completo
- Precisamos somente dos k melhores para um k pequeno (e.g., $k = 100$).
- Se não precisamos do ranking completo, existe jeito eficiente de calcular os top k?
- Método simples ineficiente:

Como calculamos os top k documentos no ranking?

- Em muitas aplicações, não precisamos de um ranking completo
- Precisamos somente dos k melhores para um k pequeno (e.g., $k = 100$).
- Se não precisamos do ranking completo, existe jeito eficiente de calcular os top k?
- Método simples ineficiente:
  - Calcular pontuação para todos os documentos

Como calculamos os top k documentos no ranking?

- Em muitas aplicações, não precisamos de um ranking completo
- Precisamos somente dos k melhores para um k pequeno (e.g., $k = 100$).
- Se não precisamos do ranking completo, existe jeito eficiente de calcular os top k?
- Método simples ineficiente:
  - Calcular pontuação para todos os documentos
  - Ordenar

Como calculamos os top k documentos no ranking?

- Em muitas aplicações, não precisamos de um ranking completo
- Precisamos somente dos k melhores para um k pequeno (e.g., $k = 100$).
- Se não precisamos do ranking completo, existe jeito eficiente de calcular os top k?
- Método simples ineficiente:
  - Calcular pontuação para todos os documentos
  - Ordenar
  - Retornar os k com maior pontuação

Como calculamos os top k documentos no ranking?

- Em muitas aplicações, não precisamos de um ranking completo
- Precisamos somente dos k melhores para um k pequeno (e.g., $k = 100$).
- Se não precisamos do ranking completo, existe jeito eficiente de calcular os top k?
- Método simples ineficiente:
  - Calcular pontuação para todos os documentos
  - Ordenar
  - Retornar os k com maior pontuação
- Alternativa: min heap

Min heap para selecionar os k melhores resultados de N

- Uma min heap binária é uma árvore binária em que cada nó tem valor menor que os valores de nós-filhos

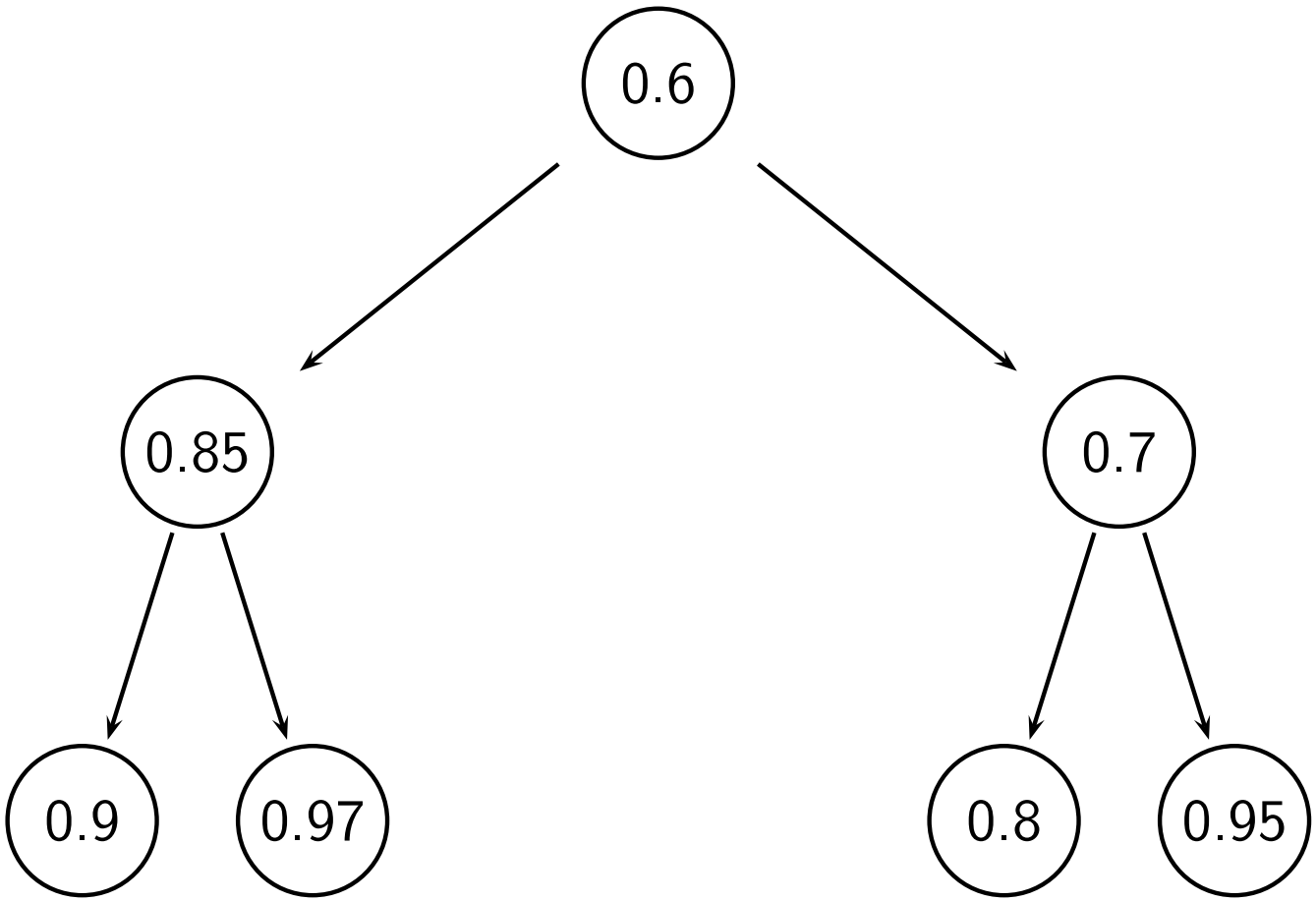
Min heap para selecionar os k melhores resultados de N

- Uma min heap binária é uma árvore binária em que cada nó tem valor menor que os valores de nós-filhos
- Custa $O(N \log k)$ operações para construir (onde N é o número de documentos) ...

Min heap para selecionar os k melhores resultados de N

- Uma min heap binária é uma árvore binária em que cada nó tem valor menor que os valores de nós-filhos
- Custa $O(N \log k)$ operações para construir (onde N é o número de documentos) . . .
- . . . e para ler os k melhores custa $O(k \log k)$

Árvore min heap binária



Escolhendo k melhores documentos em $O(N \log k)$

- Objetivo: manter os melhores k documentos vistos até o momento

Escolhendo k melhores documentos em $O(N \log k)$

- Objetivo: manter os melhores k documentos vistos até o momento
- Usar uma min heap binária

Escolhendo k melhores documentos em $O(N \log k)$

- Objetivo: manter os melhores k documentos vistos até o momento
- Usar uma min heap binária
- Processar um novo documento d' com pontuação s':

Escolhendo k melhores documentos em $O(N \log k)$

- Objetivo: manter os melhores k documentos vistos até o momento
- Usar uma min heap binária
- Processar um novo documento d' com pontuação s':
  - Pegar mínimo atual h_m da heap ($O(1)$)

Escolhendo k melhores documentos em $O(N \log k)$

- Objetivo: manter os melhores k documentos vistos até o momento
- Usar uma min heap binária
- Processar um novo documento d' com pontuação s':
  - Pegar mínimo atual h_m da heap ($O(1)$)
  - Se $s' \leq h_m$, pula para próximo documento

Escolhendo k melhores documentos em $O(N \log k)$

- Objetivo: manter os melhores k documentos vistos até o momento
- Usar uma min heap binária
- Processar um novo documento d' com pontuação s':
  - Pegar mínimo atual h_m da heap ($O(1)$)
  - Se $s' \leq h_m$, pula para próximo documento
  - Se $s' > h_m$ heap-remover-raiz ($O(\log k)$)

Escolhendo k melhores documentos em $O(N \log k)$

- Objetivo: manter os melhores k documentos vistos até o momento
- Usar uma min heap binária
- Processar um novo documento d' com pontuação s':
  - Pegar mínimo atual h_m da heap ($O(1)$)
  - Se $s' \leq h_m$, pula para próximo documento
  - Se $s' > h_m$ heap-remover-raiz ($O(\log k)$)
  - Heap-inserir d'/s' ($O(\log k)$)

Computação mais eficiente dos k melhores?

- Ranking tem complexidade de tempo $O(N)$ onde N é o número de documentos.

Computação mais eficiente dos k melhores?

- Ranking tem complexidade de tempo $O(N)$ onde N é o número de documentos.
- Otimizações reduzem o fator constante mas são ainda $O(N)$, $N > 10^{10}$

Computação mais eficiente dos k melhores?

- Ranking tem complexidade de tempo $O(N)$ onde N é o número de documentos.
- Otimizações reduzem o fator constante mas são ainda $O(N)$, $N > 10^{10}$
- Existem algoritmos sublineares?

Computação mais eficiente dos k melhores?

- Ranking tem complexidade de tempo $O(N)$ onde N é o número de documentos.
- Otimizações reduzem o fator constante mas são ainda $O(N)$, $N > 10^{10}$
- Existem algoritmos sublineares?
- Estamos tentando resolver o problema de k vizinhos mais próximos para o vetor de consulta

Computação mais eficiente dos k melhores?

- Ranking tem complexidade de tempo $O(N)$ onde N é o número de documentos.
- Otimizações reduzem o fator constante mas são ainda $O(N)$, $N > 10^{10}$
- Existem algoritmos sublineares?
- Estamos tentando resolver o problema de k vizinhos mais próximos para o vetor de consulta
- De modo geral não há soluções sublineares

Computação mais eficiente dos k melhores: heurísticas

- Ideia 1: Reordenar listas de referências

Computação mais eficiente dos k melhores: heurísticas

- Ideia 1: Reordenar listas de referências
  - Ao invés de ordenar de acordo com docID ...

Computação mais eficiente dos k melhores: heurísticas

- Ideia 1: Reordenar listas de referências
  - Ao invés de ordenar de acordo com docID ...
  - ... ordenar de acordo com uma medida de "relevância esperada"

Computação mais eficiente dos k melhores: heurísticas

- Ideia 1: Reordenar listas de referências
  - Ao invés de ordenar de acordo com docID ...
  - ... ordenar de acordo com uma medida de "relevância esperada"
    - Exemplo: PageRank (veremos em aulas futuras)

Computação mais eficiente dos k melhores: heurísticas

- Ideia 1: Reordenar listas de referências
  - Ao invés de ordenar de acordo com docID ...
  - ... ordenar de acordo com uma medida de "relevância esperada"
    - Exemplo: PageRank (veremos em aulas futuras)
- Ideia 2: heurísticas para podar o espaço de busca

Computação mais eficiente dos k melhores: heurísticas

- Ideia 1: Reordenar listas de referências
  - Ao invés de ordenar de acordo com docID ...
  - ... ordenar de acordo com uma medida de "relevância esperada"
    - Exemplo: PageRank (veremos em aulas futuras)
- Ideia 2: heurísticas para podar o espaço de busca
  - Corretude não é garantida ...

Computação mais eficiente dos k melhores: heurísticas

- Ideia 1: Reordenar listas de referências
  - Ao invés de ordenar de acordo com docID ...
  - ... ordenar de acordo com uma medida de "relevância esperada"
    - Exemplo: PageRank (veremos em aulas futuras)
- Ideia 2: heurísticas para podar o espaço de busca
  - Corretude não é garantida ...
  - ... mas raramente falha.

Computação mais eficiente dos k melhores: heurísticas

- Ideia 1: Reordenar listas de referências
  - Ao invés de ordenar de acordo com docID ...
  - ... ordenar de acordo com uma medida de "relevância esperada"
    - Exemplo: PageRank (veremos em aulas futuras)
- Ideia 2: heurísticas para podar o espaço de busca
  - Corretude não é garantida ...
  - ... mas raramente falha.
  - Na prática, próximo de tempo constante

Ordenação de listas de referências sem usar docID

- Até agora: listas de referências foram ordenadas de acordo com docID

Ordenação de listas de referências sem usar docID

- Até agora: listas de referências foram ordenadas de acordo com docID
- Alternativa: uma medida independente da consulta de avaliação da página

Ordenação de listas de referências sem usar docID

- Até agora: listas de referências foram ordenadas de acordo com docID
- Alternativa: uma medida independente da consulta de avaliação da página
- Exemplo: **PageRank** $g(d)$ de uma página d, uma medida de quantas páginas "boas" linkam d

Ordenação de listas de referências sem usar docID

- Até agora: listas de referências foram ordenadas de acordo com docID
- Alternativa: uma medida independente da consulta de avaliação da página
- Exemplo: **PageRank** $g(d)$ de uma página d, uma medida de quantas páginas "boas" linkam d
- Ordenar documentos em listas de referências de acordo com PageRank: $g(d_1) > g(d_2) > g(d_3) > \ldots$

Ordenação de listas de referências sem usar docID

- Até agora: listas de referências foram ordenadas de acordo com docID
- Alternativa: uma medida independente da consulta de avaliação da página
- Exemplo: **PageRank** $g(d)$ de uma página d, uma medida de quantas páginas "boas" linkam d
- Ordenar documentos em listas de referências de acordo com PageRank: $g(d_1) > g(d_2) > g(d_3) > \ldots$
- Definir uma pontuação composta de um documento d em relação a uma consulta q:

$$\text{pontuacao-total}(q, d) = g(d) + \cos(q, d)$$

Ordenação de listas de referências sem usar docID

- Até agora: listas de referências foram ordenadas de acordo com docID
- Alternativa: uma medida independente da consulta de avaliação da página
- Exemplo: **PageRank** $g(d)$ de uma página d, uma medida de quantas páginas "boas" linkam d
- Ordenar documentos em listas de referências de acordo com PageRank: $g(d_1) > g(d_2) > g(d_3) > \ldots$
- Definir uma pontuação composta de um documento d em relação a uma consulta q:

$$\text{pontuacao-total}(q, d) = g(d) + \cos(q, d)$$

- Este esquema permite interrupção ao encontrar os melhores k documentos

Ordenação sem usar docID da lista de referências (2)

- Ordenar documentos em listas de referências de acordo com PageRank: $g(d_1) > g(d_2) > g(d_3) > \ldots$
- Definir pontuação composta de um documento:

$$\text{pontuacao-total}(q, d) = g(d) + \cos(q, d)$$

- Suponha: (i) $g \rightarrow [0, 1]$; (ii) $g(d) < 0.1$ para o documento d que estamos processando; (iii) a menor pontuação entre os k documentos mais relevantes que encontramos até agora é 1.2

Ordenação sem usar docID da lista de referências (2)

- Ordenar documentos em listas de referências de acordo com PageRank: $g(d_1) > g(d_2) > g(d_3) > \ldots$
- Definir pontuação composta de um documento:

$$\text{pontuacao-total}(q, d) = g(d) + \cos(q, d)$$

- Suponha: (i) $g \rightarrow [0, 1]$; (ii) $g(d) < 0.1$ para o documento d que estamos processando; (iii) a menor pontuação entre os k documentos mais relevantes que encontramos até agora é 1.2

Ordenação sem usar docID da lista de referências (2)

- Ordenar documentos em listas de referências de acordo com PageRank: $g(d_1) > g(d_2) > g(d_3) > \ldots$
- Definir pontuação composta de um documento:

$$\text{pontuacao-total}(q, d) = g(d) + \cos(q, d)$$

- Suponha: (i) $g \to [0, 1]$; (ii) $g(d) < 0.1$ para o documento d que estamos processando; (iii) a menor pontuação entre os k documentos mais relevantes que encontramos até agora é 1.2
- Então toda a pontuação seguinte será < 1.1.

Ordenação sem usar docID da lista de referências (2)

- Ordenar documentos em listas de referências de acordo com PageRank: $g(d_1) > g(d_2) > g(d_3) > \ldots$
- Definir pontuação composta de um documento:

$$\text{pontuacao-total}(q, d) = g(d) + \cos(q, d)$$

- Suponha: (i) $g \rightarrow [0, 1]$; (ii) $g(d) < 0.1$ para o documento d que estamos processando; (iii) a menor pontuação entre os k documentos mais relevantes que encontramos até agora é 1.2
- Então toda a pontuação seguinte será < 1.1.
- Assim, já encontramos os k melhores documentos e podemos interromper o processamento do resto das referências do índice invertido

Ordenação sem usar docID da lista de referências (2)

- Ordenar documentos em listas de referências de acordo com PageRank: $g(d_1) > g(d_2) > g(d_3) > \ldots$
- Definir pontuação composta de um documento:

$$\text{pontuacao-total}(q, d) = g(d) + \cos(q, d)$$

- Suponha: (i) $g \rightarrow [0, 1]$; (ii) $g(d) < 0.1$ para o documento d que estamos processando; (iii) a menor pontuação entre os k documentos mais relevantes que encontramos até agora é 1.2
- Então toda a pontuação seguinte será < 1.1.
- Assim, já encontramos os k melhores documentos e podemos interromper o processamento do resto das referências do índice invertido

Processamento de um documento por vez

- Ordenação por docID e por PageRank impõem ordenação consistente dos documentos nas listas de referências do índice invertido

Processamento de um documento por vez

- Ordenação por docID e por PageRank impõem ordenação consistente dos documentos nas listas de referências do índice invertido
- O cálculo de cossenos nesse esquema é realizado **um documento por vez**.

Processamento de um documento por vez

- Ordenação por docID e por PageRank impõem ordenação consistente dos documentos nas listas de referências do índice invertido
- O cálculo de cossenos nesse esquema é realizado **um documento por vez**.
- Nós completamos a computação da pontuação da similaridade consulta-documento do documento d_i antes de começar a calcular a similaridade de d_{i+1}

Processamento de um documento por vez

- Ordenação por docID e por PageRank impõem ordenação consistente dos documentos nas listas de referências do índice invertido
- O cálculo de cossenos nesse esquema é realizado **um documento por vez**.
- Nós completamos a computação da pontuação da similaridade consulta-documento do documento d_i antes de começar a calcular a similaridade de d_{i+1}
- Alternativa: processamento de **um termo por vez**

Heurísticas: lista de referências ordenada por pesos

- Processamento de **um termo por vez**

Heurísticas: lista de referências ordenada por pesos

- Processamento de **um termo por vez**
  - Ideia: não processar termos que contribuem pouco com a pontuação final

Heurísticas: lista de referências ordenada por pesos

- Processamento de **um termo por vez**
  - Ideia: não processar termos que contribuem pouco com a pontuação final
  - Ordenar termos de acordo com idf

Heurísticas: lista de referências ordenada por pesos

- Processamento de **um termo por vez**
  - Ideia: não processar termos que contribuem pouco com a pontuação final
  - Ordenar termos de acordo com idf
  - Ordenar documentos na lista de referências de acordo com tf

Heurísticas: lista de referências ordenada por pesos

- Processamento de **um termo por vez**
  - Ideia: não processar termos que contribuem pouco com a pontuação final
  - Ordenar termos de acordo com idf
  - Ordenar documentos na lista de referências de acordo com tf
  - Os k melhores documentos são prováveis de aparecer no começo dessas listas ordenadas

Heurísticas: lista de referências ordenada por pesos

- Processamento de **um termo por vez**
  - Ideia: não processar termos que contribuem pouco com a pontuação final
  - Ordenar termos de acordo com idf
  - Ordenar documentos na lista de referências de acordo com tf
  - Os k melhores documentos são prováveis de aparecer no começo dessas listas ordenadas
  - $\rightarrow$ Interrupção precoce do processamento da lista de referências quando for improvável de mudar os k melhores

Processamento de um termo por vez

- Caso mais simples: processar completamente a lista de referências do primeiro termo da consulta

Processamento de um termo por vez

- Caso mais simples: processar completamente a lista de referências do primeiro termo da consulta
- Criar um acumulador para cada docID que encontrar

Processamento de um termo por vez

- Caso mais simples: processar completamente a lista de referências do primeiro termo da consulta
- Criar um acumulador para cada docID que encontrar
- Então processar completamente a lista de referências do segundo termo da consulta

Processamento de um termo por vez

- Caso mais simples: processar completamente a lista de referências do primeiro termo da consulta
- Criar um acumulador para cada docID que encontrar
- Então processar completamente a lista de referências do segundo termo da consulta
- . . . e assim por diante

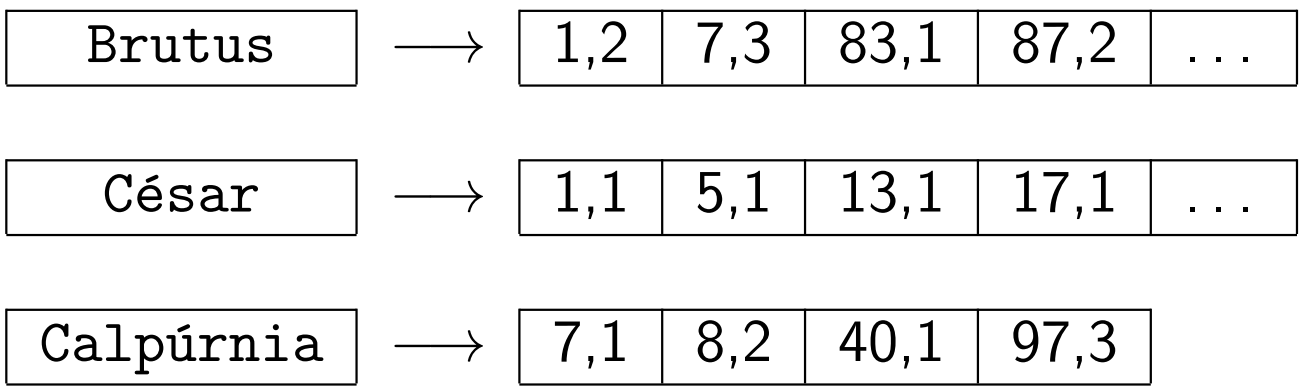
Processamento de um termo por vez

PontuacaoCosseno(q)

```
1  float Acumuladores[N] = 0
2  float Comprimento[N]
3  for each termo consulta t
4  do calcular w_{t,q} e acessar lista de referências para t
5      for each par(d, tf_{t,d}) na lista de referências
6      do Acumuladores[d] += w_{t,d} × w_{t,q}
7  Ler o array Comprimento
8  for each d
9  do Acumuladores[d] = Acumuladores[d]/Comprimento[d]
10 return Top k componentes de Acumuladores[]
```

Exemplo: acumuladores

- Para a web (40 bilhões de documentos) um array de acumuladores em memória é inviável
  - Ou seja: não criar acumuladores para docs que não contém nenhum dos termos de consulta



- Para consulta: [Brutus César]
- Somente usa acumuladores para 1, 5, 7, 13, 17, 83, 87
- Não precisa de acumuladores para 3, 8 etc.

Forçando o uso de busca conjuntiva

- Podemos usar busca conjuntiva (estilo booleano): considerar somente documentos (e criar acumuladores) se todos termos ocorrerem

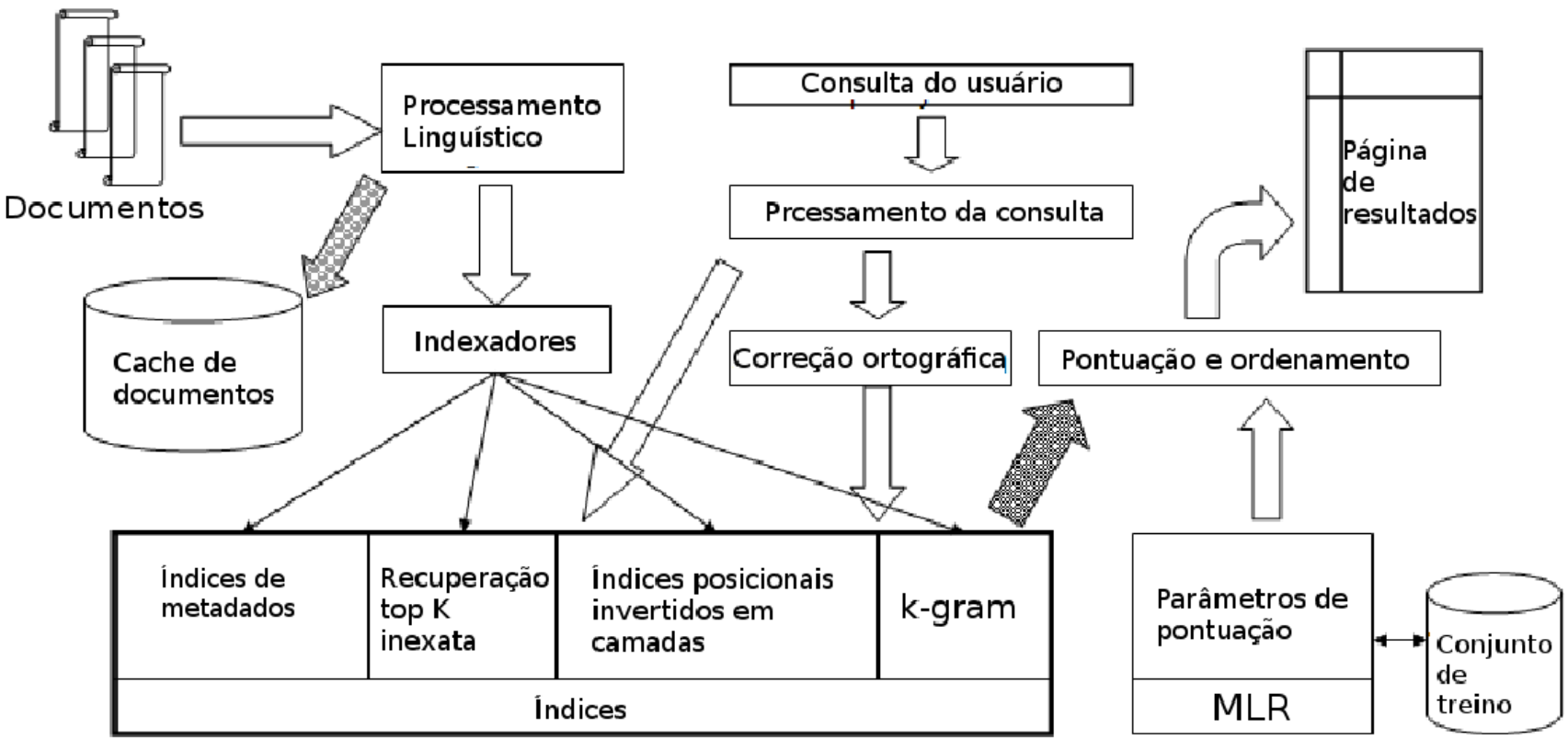
Forçando o uso de busca conjuntiva

- Podemos usar busca conjuntiva (estilo booleano): considerar somente documentos (e criar acumuladores) se todos termos ocorrerem
- Exemplo: apenas um acumulador para [Brutus César] no exemplo anterior ...

Forçando o uso de busca conjuntiva

- Podemos usar busca conjuntiva (estilo booleano): considerar somente documentos (e criar acumuladores) se todos termos ocorrerem
- Exemplo: apenas um acumulador para [Brutus César] no exemplo anterior ...
- ... porque somente d_1 contém ambas as palavras.

Um sistema de busca completo



Índices em camadas

- Ideia básica:

Índices em camadas

- Ideia básica:
  - Criar várias camadas de índices, correspondendo à importância de termos de indexação

Índices em camadas

- Ideia básica:
  - Criar várias camadas de índices, correspondendo à importância de termos de indexação
  - Durante processamento de consultas, inicias com índice de camada mais alta

Índices em camadas

- Ideia básica:
  - Criar várias camadas de índices, correspondendo à importância de termos de indexação
  - Durante processamento de consultas, inicias com índice de camada mais alta
  - Se o índice da camada mais alta retorna pela menos k resultados (e.g., $k = 100$): interrompe e retorna resultados ao usuário

Índices em camadas

- Ideia básica:
  - Criar várias camadas de índices, correspondendo à importância de termos de indexação
  - Durante processamento de consultas, inicias com índice de camada mais alta
  - Se o índice da camada mais alta retorna pela menos k resultados (e.g., $k = 100$): interrompe e retorna resultados ao usuário
  - Se encontrou-se somente $< k$ resultados: repetir para índice na próxima camada

Índices em camadas

- Ideia básica:
  - Criar várias camadas de índices, correspondendo à importância de termos de indexação
  - Durante processamento de consultas, inicias com índice de camada mais alta
  - Se o índice da camada mais alta retorna pela menos k resultados (e.g., $k = 100$): interrompe e retorna resultados ao usuário
  - Se encontrou-se somente $< k$ resultados: repetir para índice na próxima camada
- Exemplo: sistema de duas camadas

Índices em camadas

- Ideia básica:
  - Criar várias camadas de índices, correspondendo à importância de termos de indexação
  - Durante processamento de consultas, inicias com índice de camada mais alta
  - Se o índice da camada mais alta retorna pela menos k resultados (e.g., $k = 100$): interrompe e retorna resultados ao usuário
  - Se encontrou-se somente $< k$ resultados: repetir para índice na próxima camada
- Exemplo: sistema de duas camadas
  - Tier 1: Índice de todos os títulos

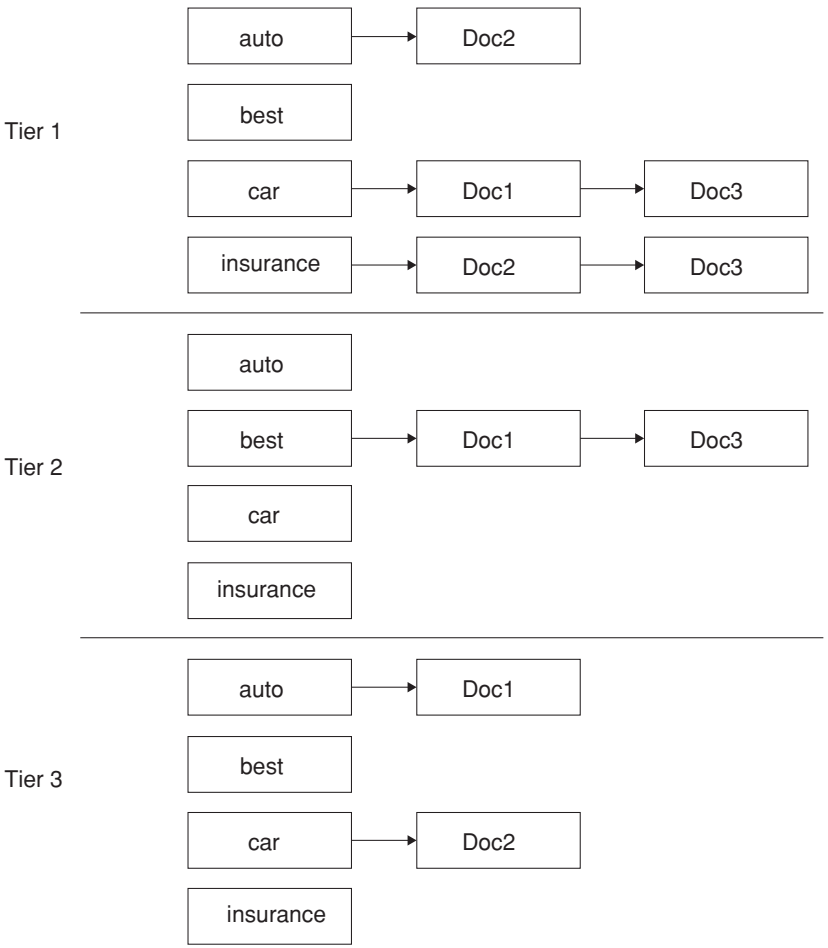
Índices em camadas

- Ideia básica:
  - Criar várias camadas de índices, correspondendo à importância de termos de indexação
  - Durante processamento de consultas, inicias com índice de camada mais alta
  - Se o índice da camada mais alta retorna pela menos k resultados (e.g., $k = 100$): interrompe e retorna resultados ao usuário
  - Se encontrou-se somente $< k$ resultados: repetir para índice na próxima camada
- Exemplo: sistema de duas camadas
  - Tier 1: Índice de todos os títulos
  - Tier 2: Índice do resto dos documentos

Índices em camadas

- Ideia básica:
  - Criar várias camadas de índices, correspondendo à importância de termos de indexação
  - Durante processamento de consultas, inicias com índice de camada mais alta
  - Se o índice da camada mais alta retorna pela menos k resultados (e.g., $k = 100$): interrompe e retorna resultados ao usuário
  - Se encontrou-se somente $< k$ resultados: repetir para índice na próxima camada
- Exemplo: sistema de duas camadas
  - Tier 1: Índice de todos os títulos
  - Tier 2: Índice do resto dos documentos
  - Páginas contendo palavras de busca no título são melhores que páginas com palavras no corpo do texto

Índices em camadas



Índices em camadas

- O uso de índices em camada é dito uma das razões da qualidade de busca do Google ter sido inicialmente (2000/2001) bem melhor que a dos concorrentes

Índices em camadas

- O uso de índices em camada é dito uma das razões da qualidade de busca do Google ter sido inicialmente (2000/2001) bem melhor que a dos concorrentes
- em conjunto com PageRank, uso de texto âncora e restrições de proximidade

Componentes de um sistema

- Pré-processamento de documentos (linguístico e outros)
- Índices de posicionais
- Índices em camadas
- Correção ortográfica
- Índices k-gram para consultas com caracter curinga e correção ortográfica
- Processamento de consultas
- Pontuação de documentos
- Processamento de um termo por vez

Componentes adicionais

- Cache de documentos: geração de "resumos dinâmicos"
- Índices de zonas: separação de índices para diferentes zonas: corpo do documento, texto grifado, texto de âncora (em link), texto em campos de meta-dados
- Funções de ranking obtidas a partir de aprendizado de máquina
- Ranking de proximidade (e.g., documentos de ranking nos quais os termos de consulta ocorrem em maior frequência em uma "janela local" que em outros documentos cujos termos estão mais esparsados)
- Análise sintática de consultas

Um sistema de busca completo