Janeiro 2013

Resumo dos apontamentos de Armazenamento, Indexação e Recuperação da Informação

Bruno Cruz  n° MEC 41537
Ricardo Marques  n° MEC 63265
Rui Mendes  n° MEC 67012
Índice

AULA 1 – INTRODUÇÃO AOS MÉTODOS E MODELOS DE RI ........................................................................... 4
AULA 2 – PESQUISA BOOLEANA, ÍNDICE INVERTIDO .................................................................................. 16
AULA 3 – PROCESSAMENTO E ANÁLISE DE TEXTO .................................................................................. 24
  TOKENS E TERMOS ..................................................................................................................................... 26
  CONSULTA DE FRASES E ÍNDICES POSICIONAIS .................................................................................. 34
AULA 4 – CONSTRUÇÃO DO ÍNDICE ........................................................................................................... 39
  ESTRUTURA DE DADOS ............................................................................................................................. 44
  JUNÇÃO DE POSTINGS MAIS RÁPIDA: IGNORAR APONTADORES/IGNORAR LISTAS ................................ 48
  CONSTRUÇÃO DO ÍNDICE ......................................................................................................................... 51
AULA 4.1 – ESTATÍSTICAS TEXTUAIS .......................................................................................................... 63
AULA 5 – CLASSIFICAÇÃO (RANKING) E MODELO DE ESPAÇO VECTORIAL .................................................. 69
AULA 6 – AVALIAÇÃO .................................................................................................................................... 82
  APRESENTAÇÃO DE RESULTADOS ........................................................................................................ 95
AULA 7 – RECUPERAÇÃO TOLERANTE (TOLERANT RETRIEVAL) ................................................................. 98
  SPelling CORRECTION .............................................................................................................................. 101
  FONÉTICA (SOUNDEX) .............................................................................................................................. 107
AULA 8 – CLASSIFICAÇÃO E CRIAÇÃO DE RESULTADOS ........................................................................ 109
  Lucene ......................................................................................................................................................... 122
Alguns termos que são considerados nos apontamentos.
Alguns tiveram tradução, outros não.

<table>
<thead>
<tr>
<th>Inglês</th>
<th>Considerado no resumo</th>
</tr>
</thead>
<tbody>
<tr>
<td>IR (Information Retrieval)</td>
<td>RI (Recuperação da Informação)</td>
</tr>
<tr>
<td>Recall</td>
<td>Retorno</td>
</tr>
<tr>
<td>Precision</td>
<td>Precisão</td>
</tr>
<tr>
<td>Document Frequency</td>
<td>Frequência de Documentos</td>
</tr>
<tr>
<td>Term Frequency</td>
<td>Frequência do termo</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Avaliação</td>
</tr>
<tr>
<td>Skip pointer</td>
<td>Apontador de salto</td>
</tr>
<tr>
<td>Tree</td>
<td>Árvore</td>
</tr>
<tr>
<td>Posting</td>
<td>Posting</td>
</tr>
<tr>
<td>Query</td>
<td>Query (consulta)</td>
</tr>
<tr>
<td>Retrieval</td>
<td>Recuperação</td>
</tr>
<tr>
<td>Ranking</td>
<td>Classificação</td>
</tr>
<tr>
<td>Indexing</td>
<td>Indexação</td>
</tr>
<tr>
<td>Searching</td>
<td>Pesquisa</td>
</tr>
<tr>
<td>Stopword</td>
<td>Stopword</td>
</tr>
<tr>
<td>Lemmatization</td>
<td>Lemmatization</td>
</tr>
<tr>
<td>Thesauri</td>
<td>Dicionário de sinónimos</td>
</tr>
<tr>
<td>Parsers</td>
<td>Analisadores</td>
</tr>
<tr>
<td>Inverters</td>
<td>Inversores</td>
</tr>
<tr>
<td>accuracy</td>
<td>Precisão</td>
</tr>
</tbody>
</table>
Recuperação e Pesquisa e Informação

✓ A pesquisa na Web é uma atividade diária para muitas pessoas em todo o mundo.
✓ A pesquisa e a comunicação são as atividades mais populares no que diz respeito à utilização do computador.
✓ Aplicações que envolvem pesquisa estão em toda parte.
✓ O campo da ciência da computação que é está mais ligado I&D (Investigação e Desenvolvimento) pesquisa é Recuperação de Informação (RI).

Recuperação da Informação

✓ “A recuperação da informação é um campo que envolve a estrutura, análise, organização, armazenamento, pesquisa e recuperação da informação.” (Salton, 1968)
✓ “Recuperação de Informação (RI) consiste em encontrar material (geralmente documentos) de natureza destruturada (geralmente texto) que satisfaça uma dada condições de uma quantidade enorme de coleções (geralmente armazenadas em computadores).” (Manning, et al, 2008)

Texto Destruturado vs. Dados Estruturados (base de dados) em 1996

![Diagrama de barras mostrando o volume de dados e mercado cap na comparação entre dados estruturados e destruturados em 1996.](image-url)
Texto Destruturado vs. Dados Estruturados (database) em 2009

Típica tarefa de RI

Fornecido

- Um corpo de documento com linguagem textual natural.
- Uma query na forma de texto (string).

Pesquisa

- O ranking de um conjunto de documentos que sejam relevantes para a query.

Sistema RI
**Pesquisa Web**

- Diferenças:
  - É necessário rastrear *(spidering)* a Web para elaborar o corpus de documentos.
  - É possível explorar a informação colocada na estrutura do HTML (XML).
  - Os documentos podem variar de uma maneira incontrolável.
  - É possível explorar a estrutura de ligações (links) da Web.

**Sistema de Pesquisa Web**

![Diagrama do Sistema de Pesquisa Web]

**Definição de documento**

**Exemplos:**

- Páginas web, e-mail, livros, notícias, artigos académicos, mensagens de texto, ficheiros do Word™, ficheiros do PowerPoint ™, PDF, mensagens em fórum, patentes, sessões de mensagens instantâneas, etc.
- Propriedades comuns.
- Texto com conteúdo significativo.
Alguma estrutura (por exemplo, título, autor, data de trabalhos; assunto, remetente de destino, destino do e-mail)

**Documentos vs. Registos de Base de Dados**

- Registos de base de dados (ou tuples – listas ordenadas de elementos – em base de dados relacionais) são geralmente compostas por campos bem definidos (ou atributos)
  - por exemplo, registos bancários com números de contas, saldos, nomes, endereços, números de segurança social, datas de nascimento, etc.
- Há facilidade de comparação entre a semântica de campos bem definidos com consultas, a fim de encontrar correspondências.
- O texto torna-se mais difícil.

**Exemplo de uma query da base de dados:**

- Encontrar registos com saldo> US $ 50.000 em agências localizadas em Amherst, MA.
- Correspondência facilmente encontrada por comparação com os valores dos campos dos registos.

**Exemplo de consulta num motor de pesquisa:**

- Escândalos bancários em massa no Ocidente
- Este texto deve ser em relação ao texto de notícias inteiras

**Comparando Texto**

- Comparando o texto da consulta com o texto do documento e determinar o que é uma boa correspondência é a questão central de recuperação de informação.
- A correspondência exata de palavras poderá não ser suficiente:
  - Muitas maneiras diferentes de escrever a mesma coisa (sinónimos) na "Linguagem natural" como o Inglês
**Dimensões da RI**

✓ RI é mais do que apenas texto, e mais do que apenas Pesquisa Web, embora estes sejam essenciais.
✓ Existem Pessoas que trabalham na área de RI com diferentes objetivos, diferentes tipos de aplicações de pesquisa e diferentes tarefas.

**Outros meios de comunicação**

✓ Novas aplicações envolvem cada vez mais as novas formas de comunicação:
  - por exemplo, vídeo, fotos, música, discurso,
✓ Tal como o texto, o conteúdo é difícil de descrever e comparar
  - O texto pode ser usado para representá-los (por exemplo: etiquetas – tags)
✓ As técnicas de RI para pesquisa e avaliação são apropriadas.

**Dimensões da RI**

<table>
<thead>
<tr>
<th>Content</th>
<th>Applications</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Web search</td>
<td>Ad hoc search</td>
</tr>
<tr>
<td>Images</td>
<td>Vertical search</td>
<td>Filtering</td>
</tr>
<tr>
<td>Video</td>
<td>Enterprise search</td>
<td>Classification</td>
</tr>
<tr>
<td>Scanned docs</td>
<td>Desktop search</td>
<td>Question answering</td>
</tr>
<tr>
<td>Audio</td>
<td>Forum search</td>
<td></td>
</tr>
<tr>
<td>Music</td>
<td>P2P search</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Literature search</td>
<td></td>
</tr>
</tbody>
</table>
Arquitetura do Sistema de RI

Componentes do Sistema de RI

- **Operações de Texto** (Text Operations) criam um índice de palavras (tokens).
  - Remoção de Stopwords
  - Stemming
- A **Indexação** (Indexing) constrói um índice invertido da palavra que aponta para documentos.
- A **Pesquisa** (Searching) encontra documentos que contenham o token da query no índice invertido.
- Na **Avaliação** (Ranking) são avaliados (e ordenados) todos os documentos encontrados de acordo com a relevância métrica.
- A **Interface do Utilizador** (User Interface) gere a interação com o utilizador:
  - Consulta pretendida (input) e saída de documentos (output).
  - Revisão da relevância.
  - Visualização dos resultados.
- **Operações de consulta** (Query Operations) transformam a consulta com vista a melhorar a recuperação:
  - Expansão da consulta usando um dicionário de sinônimos.
  - Transformação da consulta usando feedback de relevância.
Outras tarefas relacionadas com RI

- Categorização automática de documentos
- Filtragem de Informações (filtragem de spam)
- Redirecionamento de Informações
- Agrupamento automático de documentos
- A recomendação de informações ou produtos
- Extração de Informação
- Integração da Informação
- Responder a perguntas

Questões relevantes na RI

Relevância

Definição simples (e simplista): Um documento relevante contém a informação que uma pessoa pesquisou quando submeteu a query no motor de pesquisa.

- Muitos fatores influenciam a decisão dum a pessoa decidir sobre o que é relevante ou não: por exemplo, a tarefa, o contexto, a novidade, estilo.
- Relevância temática (mesmo tema) vs Relevância do Utilizador (tudo o resto).
- Modelos de recuperação definem uma vista sobre a relevância.
- Algoritmos de classificação utilizados em motores de pesquisa são baseados em modelos de recuperação.
- A maioria dos modelos descreve propriedades estatísticas de texto em vez de propriedades relacionadas com a linguística.
  - Isto é, contando recursos de texto simples, como palavras em vez de separar e analisar as expressões.
  - As características linguísticas podem fazer parte dum modelo estatístico.
Avaliação

✓ Consiste em procedimentos experimentais e medidas para comparação dos resultados provenientes do sistema com as expectativas do utilizador.
  o O início foi com experiências de Cranfield na década de 60.
✓ Métodos de avaliação de RI são utilizados já em muitos campos.
✓ Tipicamente utilizam coleções de teste de documentos, consultas e conclusões de relevância.
  o As mais comuns são coleções TREC.
✓ **Retorno** (Recall) e **Precisão** (Precision) são dois exemplos de medidas de eficácia.

Utilizadores e necessidades de informação

✓ A avaliação da pesquisa é baseada no utilizador
✓ As palavra-chaves da query são frequentemente descrições “pobres” da verdadeira necessidade da informação
✓ A interação e contexto são importantes para a compreensão do que o utilizador pretende
✓ Consultas técnicas de melhoria (refinamento), como a expansão da consulta, sugestão de consulta, feedback de relevância tendem a melhorar o ranking dos documentos devolvidos.

Motores de pesquisa e RI

✓ Um motor de pesquisa é uma aplicação prática de técnicas de recuperação de informação para grande escala de coleções de texto
✓ Os mais conhecidos são os da Web.
✓ Existem outros motores de pesquisa de código aberto que são importantes para a pesquisa e desenvolvimento (por exemplo, Lucene, Lemur / Indri, Galago)
Questões dos motores de pesquisa

Desempenho (Performance)

✔ Medir e melhorar a eficiência da pesquisa
  o Por exemplo, reduzindo o tempo de resposta, aumentando a eficácia da consulta, aumentando a velocidade de indexação

✔ Os Índices são estruturas de dados destinadas a melhorar a eficiência da pesquisa
  o Concepção e implementação são questões importantes para os motores de pesquisa.

Os dados dinâmicos (Dynamic Data)

✔ A "coleção" para a maioria das aplicações reais está em constante mudança, em termos de atualizações, adições, exclusões
  • por exemplo, páginas da web

✔ Adquirir ou "rastrear" os documentos é uma tarefa importante
  • As medidas típicas são de cobertura – quanto foi indexada; e atualização (“frescura”) – como recentemente foi indexada.

Escalabilidade (Scalability)

✔ O objetivo é fazer com que tudo funcione com milhões de utilizadores todos os dias, e muitos terabytes de documentos.

✔ Utilizar processamento distribuído é essencial.

Adaptabilidade (Adaptability)

✔ Alterar e afinar componentes dos motores de pesquisa, tais como:
  o Algoritmo de classificação
  o Estratégia de indexação
  o Interface para diferentes aplicações.
Spam

Para a pesquisa da Web, o spam em todas as suas formas, é um dos principais problemas:

- Afeta a eficiência dos motores de pesquisa e, mais grave, a eficácia dos resultados
- Existem muitos tipos de spam, por exemplo, o termo Spam ou spamdexing, spam link, "otimização".
- Trata-se de um novo subcampo chamado “adversário” da RI, uma vez que os spammers são "adversários", com objetivos diferentes.

Áreas relacionadas com a RI

- Gestão de Base de dados
- Biblioteca e Ciência da Informação
- Inteligência Artificial
- Processamento de Linguagem Natural
- Máquinas “inteligentes”

Gestão de Base de dados

- Focada em dados estruturados armazenados em tabelas relacionais em vez de texto livre.
- Focada em processamento eficiente de consultas bem definidas numa linguagem formal (SQL).
- Semântica mais clara para ambos os dados e consultas.
- Movimento recente em direção a dados semiestruturados (XML) coloca a GBD mais próxima de RI.

Biblioteca e Ciência da Informação

- Focada nos aspectos de utilizadores humanos de recuperação da informação (interação humano-computador, interface de utilizador, visualização).
- Interessa-se com a categorização eficaz do conhecimento humano.
- Preocupa-se com a análise de citações e bibliometria (estrutura de informação).
- Trabalhos recentes sobre bibliotecas digitais colocam este tema mais próximo de RI.
Inteligência Artificial

✓ Focada na representação do conhecimento, raciocínio e ação inteligente.
✓ Formalismos para a representação de conhecimento e consultas:
  • Lógica de predicados de primeira ordem
  • Bayesian Networks (Modelo estatístico ou Modelo gráfico probabilístico)
✓ Um trabalho recente em ontologias web e agentes de informação inteligentes coloca este termo mais perto de RI.

Processamento de Linguagem Natural

✓ Focada na análise sintática, semântica e pragmática do texto de linguagem natural e do discurso.
✓ A capacidade de analisar a sintaxe (estrutura das frases) e a semântica poderia permitir a recuperação com base no significado em vez de palavras-chave.

PLN – Direções RI

✓ Métodos para determinar o sentido de uma palavra ambígua com base no contexto (desambiguação do sentido da palavra).
✓ Métodos para identificar informações específicas num documento (extração de informações).
✓ Métodos para responder a questões específicas NL que o documento incorpora.

Máquinas “inteligentes”

✓ Direcionada para o desenvolvimento de sistemas computacionais que otimiza o seu desempenho através de experiência.
✓ Classificação automática de exemplos baseados em aprendizagem de conceitos a partir de exemplos de etiquetas de treino (aprendizagem supervisionada).
✓ Métodos automáticos para agrupar exemplos não rotulados em grupos com sentido (sem aprendizagem supervisionada).
Máquinas “inteligentes” – Direções RI

✓ Categorização do Texto
✓ Classificação hierárquica automática (Yahoo).
✓ Filtragem adaptada / encaminhamento / recomendação.
✓ Filtragem automática de Spam.
✓ Agrupamento de Texto
✓ Agrupamento de resultados da consulta RI.
✓ Formação automática de hierarquias (Yahoo).
✓ Aprender para a Extração de Informações
✓ Text Mining

Palavras-chave do RI

Consulta (query) – a representação daquilo que o utilizador procura - pode ser uma lista de palavras ou frases.

Documento (document) – uma entidade informações que o utilizador quer recuperar.

Coleção (collection) – um conjunto de documentos.

Índice (index) – a representação de informação que faz com que facilita a consulta.

Termo (term) – palavra ou conceito que aparece num documento ou numa consulta.

Outros termos importantes

✓ Classificação (classification)
✓ Cluster
✓ Semelhança (similarity)
✓ Extração de Informação
✓ Frequência do termo (term frequency)
✓ Frequência de documentos inversa
✓ Precisão (precision)
✓ Retorno (recall)
✓ Ficheiro invertido
✓ Expansão de consulta (query expansion)
✓ Relevância (relevance)
✓ Feedback relevante
✓ Stemming
✓ Stopword
✓ Vector Model Space
✓ Peso (weighting)
✓ TREC / TIPSTER / MUC
Problema

✔ Quais as peças de Shakespeare que contêm as palavras **Brutos AND Caeser** mas **NOT Calpurnia**?

Solução

✔ Pode-se fazer a pesquisa por todas as peças que contenham as palavras **Brutos AND Caeser** e depois descartar/retirar as linhas que contenham **Calpurnia**.

Esta solução não é a resposta porque:

- é lenta quando a coleção de informação é muito grande;
- **NOT Calpurnia** não é trivial;
- outras operações( encontrar palavra **Romans** perto de **countrymen**) não é fazível.

✔ Recuperação classificada (melhores documentos para retornar);

Incidência de Termo-documento

<table>
<thead>
<tr>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>worser</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1 if play contains word, 0 otherwise
**Vectores de incidência**

- Vector de 0/1 para cada termo;
- Para respondermos à query: **Brutos AND Caeser mas NOT Calpurnia**
  - Pega nos vetores de **Brutos, Caesar e Calpurnia** considerando o **AND**;
  - O resultado será:
    - $110100 \text{ AND } 110111 \text{ AND } 101111 = 100100$;

**Respostas**

- *Antony and Cleopatra, Act III, Scene ii*
  
  *Agrippa [Aside to DOMITIUS ENOBARBUS]:* Why, Enobarbus,
  
  When Antony found Julius *Caesar* dead,
  
  He cried almost to roaring; and he wept
  
  When at Philippi he found *Brutus* slain.

- *Hamlet, Act III, Scene ii*
  
  *Lord Polonius: I did enact Julius *Caesar* I was killed i' the
  Capitol; *Brutus* killed me.*

**Coleções maiores**

- Considerando N = 1 milhão de documentos e cada um com 1000 palavras, com uma média de 6 bytes/palavra incluindo espaços e pontuação perfaz 6GB de dados nos documentos.
- Supondo que há M = 500k de termos distintos nesta coleção de dados não se poderia construir a matriz:
  - 500k x 1M matrix tem meio trilhão de ‘0’s e 1’s;
  - mas não tem mais de 1 bilhão de 1’s (matrix bastante dispersa)

**Qual a melhor representação?**

- A melhor representação seria apenas guardar as posições com ‘1’
Índice invertido

✓ Para cada termo t, temos de guardar a lista de todos os documentos que contêm t.
  o Identificar cada docID, que será o serial number do documento.
✓ É necessário ter em atenção que não pode ser usado um valor fixo para os arrays.

É necessário ter listas de tamanho variável:
  o Armazenar constantemente no disco é normal e melhor;
  o Em memória, pode ser usado listas ligadas ou arrays de tamanho variável.

Construção de um índice invertido
Passos para a criação do índice

✓ Sequência de pares (token modificado, DocumentID)

Ordenar (sort)

✓ Ordenar por termos e depois por docID. (passo fundamental na indexação)
Dicionário e Postings: Passos de Indexação

- Entradas de múltiplos termos num único documento são fundidas.
- Em seguida são divididas para o dicionário e para os postings.
- Depois desta operação a informação é guardada nas listas com a informação da frequência do termo no documento (Doc. Frequency).

Como é que armazenamos essas informações?

Later in the course:
- How do we index efficiently?
- How much storage do we need?
Processamento de queries: **AND**

- Considerando o processamento da query:

  **Brutus AND Caesar**
  - Localizar e devolver os resultados de **Brutus** no dicionário (devolvendo os postings);
  - Localizar e devolver os resultados de **Caesar** no dicionário (devolvendo os postings);
  - Fundir os dois resultados

  ![Diagrama de junção de postings]

**Exemplo de um algoritmo de junção de 2 postings**

\[
\text{INTERSECT}(p_1, p_2)
\]

1. \textit{answer} ← \texttt{\{}\texttt{\}
2. \textbf{while} \; p_1 \neq \texttt{NIL} \; \text{and} \; p_2 \neq \texttt{NIL}
3. \quad \textbf{do if} \; \text{docID}(p_1) = \text{docID}(p_2)
4. \quad \quad \textbf{then} \; \text{ADD}((\textit{answer}, \text{docID}(p_1))
5. \quad \quad \quad p_1 ← \text{next}(p_1)
6. \quad \quad \quad p_2 ← \text{next}(p_2)
7. \quad \quad \textbf{else if} \; \text{docID}(p_1) < \text{docID}(p_2)
8. \quad \quad \quad \textbf{then} \; p_1 ← \text{next}(p_1)
9. \quad \quad \quad \textbf{else} \; p_2 ← \text{next}(p_2)
10. \quad \textbf{return} \; \textit{answer}

**Juntar os resultados (merge)**

- Correr as duas listas devolvidas em simultâneo.

  ![Diagrama de juntar resultados]

- Se os tamanhos das listas são X e Y, o tempo de fusão é O(x+y).

**Nota:** É crucial ordenar por docID.
Queries booleanas

Correspondência exata

✓ O modelo de recuperação booleana permite fazer uma consulta que contém uma expressão booleana.
  ○ Este tipo de queries usa AND, OR e NOT para ligar termos da query
    ▪ Verifica cada documento como um conjunto de palavras
    ▪ É preciso (exato), ou seja, ou encontra a condição ou não.

✓ Este é um dos modelos mais simples de IR.
✓ É um sistema comercial de que existe há 3 décadas e muitos sistemas de pesquisa ainda o usam, tais como o email, catálogos de livrarias, Mac OS Spotlight.

Juntar resultados (merge)

✓ E como resolveríamos a fórmula arbitrária booleana?

(Brutus OR Caesar) AND NOT (Antony OR Cleopatra)

✓ Neste caso não é possível fundir os resultados nem em tempo linear e, para isso, seria necessário otimizar a query.

Otimização da query

✓ Qual a melhor ordem para se processar a query?
✓ Considerando que a query é em AND de n termos.
✓ Para cada um dos n termos, obtêm-se os resultados (postings) e então AND em todos.

Query: Brutus AND Calpurnia AND Caesar
Exemplo de otimização da query

Para se processar de forma a se obter uma frequência cada vez maior é necessário começar por um conjunto menor de resultados e iravançando de conjunto em conjunto.

**Nota:** Esta é a razão pela qual é guardada a frequência no documento.

Otimização mais geral

- Consideremos, por exemplo, a fórmula booleana:

\[(madding \text{ OR } crowd) \text{ AND } (ignoble \text{ OR } strife)\]

- O primeiro passo será obter a frequência de documentos para todos os termos
- Em seguida, será necessário estimar o tamanho de cada OR pela soma da sua frequência de documentos.
- Neste caso, será necessário processar pelos tamanhos em ordem crescente de OR.

Execute the query as (Calpurnia AND Brutus) AND Caesar.
AULA 3 – Processamento e Análise de Texto

Agenda deste capítulo

✓ Pré-processamento para o vocabulário de termos:
  - Documentos
  - Tokenização
  - Stopping
  - Lemmatization
  - Stemming
  - Handling frases

Fluxo do processo da indexação básica
Analisar um documento

✓ Qual o formato do documento?
  o pdf / Word / Excel / html?
✓ Em que linguagem está?
✓ O conjunto de caracteres que utiliza?

São questões que levantam problema de classificação, que iremos ver mais adiante.

Mas estas tarefas são, muitas vezes, feitas heurísticamente (método não comprovado cientificamente).

Complicações: Formato / linguagem

✓ Documentos a ser indexados podem incluir documentos de diversos idiomas
  o Um simples índice pode conter termos de várias línguas.
✓ Às vezes, um documento ou os seus componentes podem conter vários idiomas / formatos
  o e-mail francês com um anexo Alemão, por exemplo.

O que é um documento unitário?

✓ Um ficheiro?
✓ Um e-mail? (Talvez um dos muitos numa caixa de entrada)
✓ Um e-mail com 5 anexos?
✓ Um grupo de ficheiros (PPT ou LaTeX como páginas HTML)
Tokens e Termos

Tokenização (Tokenizing)

✓ Criar palavras através da sequência de caracteres
✓ Relativamente complexo em Inglês, pode ser mais difícil noutras línguas
✓ Os primeiros sistemas de RI consistiam em:
  - qualquer sequência de caracteres alfanuméricos de comprimento 3 ou mais
  - os termos terminavam por um espaço em branco ou outro caracter especial
  - transformando maiúsculas em minúsculas

Exemplo:

“Bigcorp's 2007 bi-annual report showed profits rose 10%.”

Transforma-se em:

“bigcorp 2007 annual report showed profits rose”

✓ Muito simples para aplicações de pesquisa ou mesmo experiências de larga escala.
✓ Porquê? Perdemos muita informação.

Conclusão:

✓ Pequenas decisões neste processo podem ter um grande impacto na eficácia de algumas pesquisas.

Problemas na Tokenização

✓ Pequenas palavras podem ser importantes em algumas consultas, geralmente em combinações
  • xp, ma, pm, ben e king, el paso, master p, gm, j lo, world war II
✓ Ambas as formas hifenizadas e não-hifenizadas de muitas palavras são comuns
  • Por vezes, o hífen não é necessário
    • e-Bay, wal-mart, active-x, cd-rom, t-shirts
✓ Noutros casos, os hífens devem ser consideradas como parte da palavra, ou um separador da palavra
  • Winston-Salem, Mazda RX-7, e-cards, pre-diabetes, T-Mobile, de spanish-speaking
Os caracteres especiais são uma parte importante de tags, URLs, código em documentos.

- Palavras em letras maiúsculas podem ter significado diferente das palavras em minúsculas
  - Bush, Apple
- Apóstrofes podem fazer parte da palavra, de uma parte de um possessivo, ou apenas um erro
  - rosie o'donnell, can't, don't, 80's, 1890's, men's straw hats, master's degree, england's ten largest cities, shriner's
- Os números podem ser importantes, incluindo casas decimais
  - nokia 3250, top 10 courses, united 93, quicktime 6.5 pro, 92.3 the beat, 288358
- Períodos de tempo podem ser identificados por números, abreviaturas, URLs, final de frases, e outras situações
  - I.B.M., Ph.D., cs.umass.edu, F.E.A.R.

**Nota:** os passos que são efetuados no processo de Tokenização das pesquisas (queries) devem ser os passos para o processo de indexação de documentos.

### Processo de Tokenização

- O primeiro passo é utilizar uma ferramenta de análise para identificar partes apropriadas do documento para “tokenizar”.
- Adiar decisões complexas a outros componentes:
  - Palavra é qualquer sequência de caracteres alfanuméricos, terminadas por um caracter de espaço em branco ou especial, com tudo convertido em minúsculas.
  - tudo indexado.
  - Exemplo: 92.3 → 92 3 mas procura encontrar documentos com 92 e 3 adjacente.
  - Incorporar algumas regras para reduzir a dependência de transformação de componentes na consulta.
- Notemos que se trata de um processo diferente de tokenização simples utilizado anteriormente.
- Exemplos de regras usadas com TREC:
  - Apóstrofes ignorados nas palavras
    - o’connor → oconnor
    - bob’s → bobs
  - Pontos em abreviações ignorados
    - I.B.M. → ibm
    - Ph.D. → ph d
**Stopping**

- Tipos de palavras (determinantes, preposições) têm pouco significado em si mesmo
- Frequência elevada de ocorrências
- É o tratamento das stopwords (ou seja remove-as)
  - Reduzir o espaço do índice, melhorar o tempo de resposta, melhorar a eficácia
- Pode ser importante em combinações
  - Por exemplo, "ser ou não ser"
- A lista de Stopwords pode ser criada a partir de palavras com frequência elevada ou com base numa lista padrão
- As listas são personalizadas para aplicações, domínios e até mesmo para partes de documentos
  - Por exemplo, "clique" é uma boa stopword para texto sobre hiperligações
- A melhor política é indexar todas as palavras em documentos, tomar decisões sobre quais palavras usar no momento da consulta.

**Normalização para termos**

- É necessário "normalizar" palavras no texto indexado, bem como às palavras na consulta (query).
  - Queremos corresponder U.S.A. a USA

**Resultado**

- **Terminos**: um termo é um tipo de palavra (normalizado), que é uma entrada no dicionário do sistema de RI

- Existe uma maior equivalência em relação à classe de termos, por exemplo,
  - Excluir pontos para formar um termo
    - U.S.A., USA → USA
  - Excluir hífens para formar um termo
    - anti-discriminatory, antidiscriminatory → antidiscriminatory
Normalização: outros idiomas

- Sotaques: e.g., French résumé vs. resume.
- Tremas: e.g., German: Tuebingen vs. Tübingen
  - Deverá ser equivalente
- Critério mais importante:
  - Como é que os utilizadores costumam escrever nas suas consultas para essas palavras?
- Mesmo em idiomas que têm sotaques “standardizados”, os utilizadores muitas vezes não os digitam.
  - Muitas vezes, a melhor forma de normalizar um termo acentuado é:
    - Tuebingen, Tübingen, Tubingen → Tubingen

Casos especiais (Case folding)

- Reduzir todas as letras para minúsculas
  - Exceção: letras maiúsculas no meio da frase?
    - General Motors
    - Fed vs. fed
    - SAIL vs. sail
- Muitas vezes, o melhor é reduzir tudo para minúsculas, pois os utilizadores vão usar minúsculas, independentemente da capitalização "correta" ...
- O exemplo do Google:
  - Query: C.A.T.
Dicionário de Sinónimos e fonética (Thesauri and soundex)

✓ E como lidamos com sinónimos e homónimos?
  o Por exemplo, classes equivalentes construídas manualmente
    ▪ carro = automóvel color = colour
  o Podemos reescrever para formar termos com a mesma classe
    ▪ Quando o documento tiver o termo automóvel, indexa-se como
      carro-automóvel e vice-versa.
  o Ou podemos expandir a query
    ▪ Quando a consulta contém automóvel, procurar também em carro

✓ E sobre erros de ortografia?
  o Uma abordagem é a fonética, que cria classes de palavras equivalentes com base em
    heurísticas fonéticas

  Nota: este tema será especificado mais adiante.

Tematização (Lemmatization)

✓ Reduz as variantes da palavra na sua forma base
  o Por exemplo,
    ▪ am, are, is → be
    ▪ car, cars, car’s, cars’ → car
    ▪ the boy’s cars are different colors → the boy car be different color

✓ Tematização: implica reduzir "adequadamente" o dicionário de termos à sua forma “base”.

Stemming

✓ Muitas variações morfológicas das palavras
  o Conjugações (plurais, tempos)
  o Derivacional (fazendo verbos substantivos etc.)

✓ Na maioria dos casos, estes têm os mesmos significados ou muito semelhante

✓ Stemmers tentam reduzir as variações morfológicas das palavras a um termo comum
  o geralmente envolve a remoção de sufixos

✓ Pode ser feito no momento da indexação ou como parte de processamento de consultas (como stopwords)
Existem basicamente dois tipos:

- Baseado no Dicionário: utiliza listas de palavras relacionadas
- Algorítmica: usa programas para determinar palavras relacionadas

Algorithmic stemmers

- sufixo-s: remove as terminações, ou seja, 's', assumindo que se trata do plural
  - p.e., cats → cat, lakes → lake, wiis → wii
  - Muitos falsos negativos: supplies → supplie (deveria ser supply)
  - Alguns falsos positivos: ups → up (conceito diferente)

**Algoritmo de Porter**

- Algoritmo comum de stemming: Inglês
  - Os resultados sugerem que é pelo menos tão bom como outras opções do stemming
- Convenções + 5 fases de reduções
  - As fases são aplicadas sequencialmente
  - Cada fase é constituída por um conjunto de comandos
  - convenção: seguindo as regras num processo composto, deve-se selecionar o que contém maior sufixo.

**Regras gerais no Porter**

- ational → ate (p.e., rational → rate)
- tional → tion (p.e., conventional → convention)
- sses → ss (p.e., guesses → guess)
- ies → i (p.e., dictionaries → dictionari)

- Peso das regras sensíveis a palavras (m>1) EMENT →
  - replacement → replac
  - cement → cement
Outros algoritmos de stemming

✔ Existem outros stemmers, p.e., Lovins stemmer
  o http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm
  o Single-pass, longest suffix removal (about 250 rules)
✔ Análise morfológica completa – com benefícios mais modestos para a recuperação

O processo de stemming ou outras normalizações ajudam?

  o Ingês: resultados confusos. Ajuda no retorno (recall) de alguns resultados mas diminui a
    precisão de outros.
    • P.e., operative (dentistry) ⇒ oper
  o Definitivamente úteis para Espanhol. Alemão, Finlandês, ...
    • 30% Performance ganha no caso do idioma Finlandês!

Exemplos

Sample text: 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

Lovins stemmer: such an analyz can reve featur th ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog tranpar and acces to interpre

Porter stemmer: such an analysis 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

Paice stemmer: 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
Krovetz Stemmer

✓ Algoritmo de Dicionário híbrido
✓ A palavra é pesquisada no dicionário
  ○ Caso esteja presente, é colocada sozinha ou substituída pela "exceção"
  ○ Caso não esteja presente, há uma verificação de sufixos na palavra para que possam ser removidos
  ○ Após a remoção, o dicionário é verificado novamente
✓ Produz palavras e não derivadas (stems)
✓ Eficácia comparável
✓ Menor taxa de falsos positivos, mas um pouco maior a ocorrência de falsos negativos

Comparação de processos Stemmer

**Original text:**
Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.

**Porter stemmer:**
document describ market strategi carri compani agricultur chemic report predict market share chemic report market statist agrochem pesticid herbicid fungicid insecticid fertil predict sale market share stimul demand price cut volum sale

**Krovetz stemmer:**
document describe marketing strategy carry company agriculture chemical report prediction market share chemical report market statistic agrochem pesticide herbicide fungicide insecticide fertilizer predict sale stimulate demand price cut volume sale
Consulta de frases e índices posicionais

✓ Pretende responder a perguntas tais como o exemplo de pesquisa "Stanford University", mas considerando a query como sendo uma frase,
✓ Para isso, ele já não é suficiente para armazenar apenas <term: docs> entradas
  o Mais entradas no dicionário, ou
  o A estrutura da lista de envio deve ser expandida.

Frases

✓ A questão de processamento de texto - como é que as frases são reconhecidas?
✓ Três abordagens possíveis:
  o Utilização de palavras n-grams (n-words).
  o Identificar frases sintáticas usando uma tag POS (Part-of-speech).
  o Armazenamento das posições da palavra nos índices e operadores de proximidade nas queries

Solução 1: Índices Biword (Biword indexes)

✓ Índice de cada par consecutivo de termos no texto funcionando como uma frase
✓ Por exemplo, o texto "Amigos, romanos, compatriotas" gerará os biwords
  o amigos romanos
  o romanos compatriotas
✓ Cada um desses biwords é agora uma palavra do dicionário
✓ Duas "palavras-frase" de processamento de consultas é agora imediata.

Queries de frases longas

✓ Frases mais longas são processadas tal como se faz com wildcards:
  o stanford university palo alto pode ser separado na pesquisa booleana dos biwords:
    stanford university AND university palo AND palo alto
Sem os documentos, não podemos verificar que o resultado dos mesmos engloba a consulta booleana da frase acima.

POS Tagging

POS taggers utiliza modelos estatísticos de texto para descobrir formas (tags) sintáticas de palavras

- Exemplo de tags:
  - NN (nome singular), NNS (nome plural), VB (verbo), VBD (verb, past tense), VBN (verb, past participle), IN (preposition), JJ (adjective), CC (conjunction, e.g., “and”, “or”), PRP (pronoun), and MD (modal auxiliary, e.g., “can”, “will”).

As frases podem assim serem definidas como simples grupos de substantivos, por exemplo.

Exemplo de POS Tagging

Original text:
Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.

Brill tagger:
Solução 2: biwords extensíveis

✔ Analisar o texto indexado e executar tags POS (POS Tagging).
✔ Colocar os termos em Nomes (N) e artigos/proposições (X).
✔ Chamar qualquer conjunto de termos (string) que tem a forma NX*N como uma biword extensível.
✔ Cada biword extensível é agora parte de um termo no dicionário.
✔ Exemplo: catcher in the rye

N  X  X  N

✔ Processamento da Query: analisar os N’s e X’s
  o Segmentar a query para biwords aprimoradas
  o Vai verificar no index: catcher rye

Problemas para índices biword

✔ Falso positivos, como referido anteriormente
✔ A pesquisa pelo índice pode estoirar devido ao tamanho do dicionário (enorme)

Conclusão: índices Biword não são uma solução standard (para todas as biwords) mas poderão fazer parte da estratégia de solução.

Google N-Grams

✔ Índice n-grams de motores de pesquisa
✔ Exemplo do Google:

<table>
<thead>
<tr>
<th>N-grams</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tokens:</td>
<td>1,024,908,267,229</td>
</tr>
<tr>
<td>Number of sentences:</td>
<td>95,119,665,584</td>
</tr>
<tr>
<td>Number of unigrams:</td>
<td>13,588,391</td>
</tr>
<tr>
<td>Number of bigrams:</td>
<td>314,543,401</td>
</tr>
<tr>
<td>Number of trigrams:</td>
<td>977,069,902</td>
</tr>
<tr>
<td>Number of fourgrams:</td>
<td>1,313,818,354</td>
</tr>
<tr>
<td>Number of fivegrams:</td>
<td>1,176,470,663</td>
</tr>
</tbody>
</table>

✔ Trigrama mais frequente em Inglês é "all rights reserved".
  o Em chinês, "limited liability corporation"
Solução 3: Índices posicionais

✓ • No conteúdo, armazenar, para cada termo a posição(s) e em que documentos ele aparece:

<term, number of docs containing term;
   doc1: position1, position2 ... ;
   doc2: position1, position2 ... ;

etc.>

Exemplo de índice posicional

<be: 993427;
   1: 7, 18, 33, 72, 86, 231;
   2: 3, 149;
   4: 17, 191, 291, 430, 434;
   5: 363, 367, ...>

✓ Para consultas de frase, usamos um algoritmo de junção recursivamente no nível do documento.
✓ Mas agora precisamos lidar com mais do que apenas a igualdade

Processar uma frase na query

✓ Extrair entradas dos índices invertidos para cada termos distinto: to, be, or, not.
✓ Verificar a intersecção dos documentos:p lista de posições para enumerar todas as posições com “to be or not to be”.
   o to:

   2:1,17,74,222,551; 4:8,16,190,429,433;
   7:13,23,191; ...

   o be:

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

✓ Utiliza-se o mesmo método geral para pesquisas de proximidade.
Tamanho do índice posicional

✓ Podemos comprimir os valores de posição / deslocamentos: veremos mais adiante
✓ Ter em conta que um índice posicional aumenta substancialmente o armazenamento de dados

Conclusão: um índice posicional é agora o standard de uso por causa do poder e utilidade da frase e proximidade de query... Quer usado explicitamente ou implicitamente num sistema de recuperação classificada.

✓ Há necessidade de uma entrada para cada ocorrência, não apenas uma vez por documento.
✓ O tamanho do índice depende do tamanho médio de documento.
  ○ Média de uma página web <1000 termos
  ○ Livros, ... facilmente 100,000 termos
✓ Considerar a frequência de termos 0.1%

<table>
<thead>
<tr>
<th>Document size</th>
<th>Postings</th>
<th>Positional postings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>100,000</td>
<td>1</td>
<td>100</td>
</tr>
</tbody>
</table>

Regras de ouro

✓ Um índice posicional é cerca de 2–4 vezes maior que um índice não-posicional
✓ O tamanho de um índice posicional é cerca de 35–50% do volume global do texto.

Observação: estas conclusões destinam-se para os idiomas “English-like”.
Índices

✓ Os índices são estruturas de dados desenhadas para aumentar a rapidez da pesquisa.
✓ Pesquisar texto tem requisitos únicos, que leva a uma estrutura de dados exclusiva.
✓ A estrutura mais comum é o índice invertido.
  ○ De uma maneira geral, o conceito “invertido” consiste nas associações dos documentos com os termos, em vez dos termos com documentos.

Avaliação e Índices (Indexes and Ranking)

✓ Os índices foram projetados para suportar pesquisas
  ○ Tempo de resposta rápido e suporta atualizações
✓ Os motores de pesquisa de texto utilizam uma forma particular de pesquisa: avaliação (ranking)
  ○ Os documentos são recuperados pela ordem de classificação de acordo com uma pontuação, calculada através de computação, utilizando a representação de documentos, a consulta e um algoritmo de classificação

O que é um modelo abstrato razoável para o ranking?

✓ Possibilita a discussão de índices sem pormenores do modelo de recuperação.

Modelo abstrato de Avaliação
**Índice Invertido (Inverted Index)**

- Cada termo do índice está associado a uma lista invertida
  - Contém listas de documentos, ou lista de nº de ocorrências, ou outra informação
  - Cada entrada é chamada de *posting*
  - A parte do posting que se refere a um documento específico é chamada de *apontador*
  - Cada documento da coleção tem um número único
  - As listas, geralmente, são ordenadas pelos documentos (ordenadas pelo nº do documento)

\[
R(Q, D) = \sum_i g_i(Q) f_i(D)
\]

- \(f_i\) é uma função de recurso documental
- \(g_i\) é uma função de consulta

---

Fred's Tropical Fish Shop is the best place to find tropical fish at low, low prices. Whether you're looking for a little fish or a big fish, we've got what you need. We even have fake seaweed for your fish tank (and little surfboards too).
Exemplo de “Coleção”

$S_1$ Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.

$S_2$ Fishkeepers often use the term tropical fish to refer only those requiring fresh water, with saltwater tropical fish referred to as marine fish.

$S_3$ Tropical fish are popular aquarium fish, due to their often bright coloration.

$S_4$ In freshwater fish, this coloration typically derives from iridescence, while salt water fish are generally pigmented.

Four sentences from the Wikipedia entry for tropical fish

Índice Invertido Simples

<table>
<thead>
<tr>
<th>Term</th>
<th>Frequency</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>aquarium</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>are</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>around</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>as</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>both</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>bright</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>coloration</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>derives</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>due</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>environments</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>fish</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>fishkeepers</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>found</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>fresh</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>freshwater</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>from</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>generally</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>in</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>include</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>including</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>iridescence</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>marine</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>often</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Página 41 de 128
Índice Invertido com nº de ocorrências

<table>
<thead>
<tr>
<th>term</th>
<th>count</th>
<th>position</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>1,15</td>
<td>2,22</td>
</tr>
<tr>
<td>aquarium</td>
<td>5,5</td>
<td>2,10</td>
</tr>
<tr>
<td>are</td>
<td>3,3</td>
<td>4,10</td>
</tr>
<tr>
<td>around</td>
<td>1,9</td>
<td>6,16</td>
</tr>
<tr>
<td>as</td>
<td>2,31</td>
<td>2,9</td>
</tr>
<tr>
<td>both</td>
<td>1,13</td>
<td>4,1</td>
</tr>
<tr>
<td>bright</td>
<td>5,11</td>
<td>4,5</td>
</tr>
<tr>
<td>coloration</td>
<td>3,12</td>
<td>2,18</td>
</tr>
<tr>
<td>derives</td>
<td>4,7</td>
<td>2,55</td>
</tr>
<tr>
<td>due</td>
<td>5,7</td>
<td>4,4</td>
</tr>
<tr>
<td>environments</td>
<td>1,8</td>
<td>2,7</td>
</tr>
<tr>
<td>fish</td>
<td>1,2</td>
<td>2,7</td>
</tr>
<tr>
<td>fishkeepers</td>
<td>1,6</td>
<td>2,11</td>
</tr>
<tr>
<td>found</td>
<td>1,9</td>
<td>1,17</td>
</tr>
<tr>
<td>fresh</td>
<td>2,13</td>
<td>2,8</td>
</tr>
<tr>
<td>freshwater</td>
<td>1,14</td>
<td>2,20</td>
</tr>
<tr>
<td>from</td>
<td>4,5</td>
<td>1,17</td>
</tr>
<tr>
<td>generally</td>
<td>4,16</td>
<td>4,6</td>
</tr>
<tr>
<td>in</td>
<td>1,1</td>
<td>4,1</td>
</tr>
<tr>
<td>include</td>
<td>1,1</td>
<td>4,1</td>
</tr>
<tr>
<td>including</td>
<td>1,1</td>
<td>4,1</td>
</tr>
<tr>
<td>iridescence</td>
<td>4,1</td>
<td>2,15</td>
</tr>
<tr>
<td>marine</td>
<td>2,1</td>
<td>2,10</td>
</tr>
<tr>
<td>often</td>
<td>2,1</td>
<td>3,3</td>
</tr>
<tr>
<td>only</td>
<td>2,2</td>
<td>3,10</td>
</tr>
<tr>
<td>pigmented</td>
<td>2,10</td>
<td>4,15</td>
</tr>
<tr>
<td>popular</td>
<td>4,16</td>
<td>2,9</td>
</tr>
<tr>
<td>refer</td>
<td>2,16</td>
<td>2,9</td>
</tr>
<tr>
<td>referred</td>
<td>2,9</td>
<td>4,1</td>
</tr>
<tr>
<td>requiring</td>
<td>2,19</td>
<td>4,1</td>
</tr>
<tr>
<td>salt</td>
<td>2,19</td>
<td>1,16</td>
</tr>
<tr>
<td>saltwater</td>
<td>1,1</td>
<td>4,1</td>
</tr>
<tr>
<td>species</td>
<td>1,1</td>
<td>1,1</td>
</tr>
<tr>
<td>term</td>
<td>1,1</td>
<td>1,1</td>
</tr>
<tr>
<td>the</td>
<td>1,1</td>
<td>1,1</td>
</tr>
<tr>
<td>their</td>
<td>3,1</td>
<td>1,1</td>
</tr>
<tr>
<td>their</td>
<td>4,1</td>
<td>1,1</td>
</tr>
<tr>
<td>those</td>
<td>2,1</td>
<td>2,2</td>
</tr>
<tr>
<td>those</td>
<td>4,1</td>
<td>2,2</td>
</tr>
<tr>
<td>tropical</td>
<td>2,2</td>
<td>2,2</td>
</tr>
<tr>
<td>typically</td>
<td>4,1</td>
<td>2,2</td>
</tr>
<tr>
<td>use</td>
<td>2,1</td>
<td>2,2</td>
</tr>
<tr>
<td>water</td>
<td>1,1</td>
<td>2,1</td>
</tr>
<tr>
<td>while</td>
<td>2,1</td>
<td>2,1</td>
</tr>
<tr>
<td>world</td>
<td>1,1</td>
<td>1,1</td>
</tr>
</tbody>
</table>

Índice Invertido com posições

<table>
<thead>
<tr>
<th>term</th>
<th>count</th>
<th>position</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>1,15</td>
<td>2,22</td>
</tr>
<tr>
<td>aquarium</td>
<td>5,5</td>
<td>2,10</td>
</tr>
<tr>
<td>are</td>
<td>3,3</td>
<td>4,10</td>
</tr>
<tr>
<td>around</td>
<td>1,9</td>
<td>6,16</td>
</tr>
<tr>
<td>as</td>
<td>2,31</td>
<td>2,9</td>
</tr>
<tr>
<td>both</td>
<td>1,13</td>
<td>4,1</td>
</tr>
<tr>
<td>bright</td>
<td>5,11</td>
<td>4,5</td>
</tr>
<tr>
<td>coloration</td>
<td>3,12</td>
<td>2,18</td>
</tr>
<tr>
<td>derives</td>
<td>4,7</td>
<td>2,55</td>
</tr>
<tr>
<td>due</td>
<td>5,7</td>
<td>4,4</td>
</tr>
<tr>
<td>environments</td>
<td>1,8</td>
<td>2,7</td>
</tr>
<tr>
<td>fish</td>
<td>1,2</td>
<td>2,7</td>
</tr>
<tr>
<td>fishkeepers</td>
<td>1,6</td>
<td>2,11</td>
</tr>
<tr>
<td>found</td>
<td>1,9</td>
<td>1,17</td>
</tr>
<tr>
<td>fresh</td>
<td>2,13</td>
<td>2,8</td>
</tr>
<tr>
<td>freshwater</td>
<td>1,14</td>
<td>2,20</td>
</tr>
<tr>
<td>from</td>
<td>4,5</td>
<td>1,17</td>
</tr>
<tr>
<td>generally</td>
<td>4,16</td>
<td>4,6</td>
</tr>
<tr>
<td>in</td>
<td>1,1</td>
<td>4,1</td>
</tr>
<tr>
<td>include</td>
<td>1,1</td>
<td>4,1</td>
</tr>
<tr>
<td>including</td>
<td>1,1</td>
<td>4,1</td>
</tr>
<tr>
<td>iridescence</td>
<td>4,1</td>
<td>2,15</td>
</tr>
<tr>
<td>marine</td>
<td>2,1</td>
<td>2,10</td>
</tr>
<tr>
<td>often</td>
<td>2,1</td>
<td>3,3</td>
</tr>
<tr>
<td>only</td>
<td>2,2</td>
<td>3,10</td>
</tr>
<tr>
<td>pigmented</td>
<td>2,10</td>
<td>4,15</td>
</tr>
<tr>
<td>popular</td>
<td>4,16</td>
<td>2,9</td>
</tr>
<tr>
<td>refer</td>
<td>2,16</td>
<td>2,9</td>
</tr>
<tr>
<td>referred</td>
<td>2,9</td>
<td>4,1</td>
</tr>
<tr>
<td>requiring</td>
<td>2,19</td>
<td>4,1</td>
</tr>
<tr>
<td>salt</td>
<td>2,19</td>
<td>1,16</td>
</tr>
<tr>
<td>saltwater</td>
<td>1,1</td>
<td>4,1</td>
</tr>
<tr>
<td>species</td>
<td>1,1</td>
<td>2,2</td>
</tr>
<tr>
<td>term</td>
<td>1,1</td>
<td>4,1</td>
</tr>
<tr>
<td>the</td>
<td>1,1</td>
<td>4,1</td>
</tr>
<tr>
<td>their</td>
<td>3,1</td>
<td>1,1</td>
</tr>
<tr>
<td>their</td>
<td>4,1</td>
<td>1,1</td>
</tr>
<tr>
<td>those</td>
<td>2,1</td>
<td>2,2</td>
</tr>
<tr>
<td>those</td>
<td>4,1</td>
<td>2,2</td>
</tr>
<tr>
<td>tropical</td>
<td>2,2</td>
<td>2,2</td>
</tr>
<tr>
<td>typically</td>
<td>4,1</td>
<td>2,2</td>
</tr>
<tr>
<td>use</td>
<td>2,1</td>
<td>2,2</td>
</tr>
<tr>
<td>water</td>
<td>1,1</td>
<td>2,2</td>
</tr>
<tr>
<td>while</td>
<td>2,1</td>
<td>2,2</td>
</tr>
<tr>
<td>world</td>
<td>1,1</td>
<td>1,1</td>
</tr>
</tbody>
</table>
Correspondências de Proximidade

✓ Encontrar a correspondência de frases ou palavras com proximidade
✓ P.e. “tropical fish”, ou “find tropical within 5 words of fish”
✓ As posições das palavras nas listas invertidas torna este tipo de ferramentas de pesquisa eficientes
✓ P.e.,

```
<table>
<thead>
<tr>
<th></th>
<th>tropical</th>
<th>fish</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,1</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td>1,7</td>
<td>1,4</td>
</tr>
<tr>
<td>2</td>
<td>2,6</td>
<td>2,7</td>
</tr>
<tr>
<td></td>
<td>2,17</td>
<td>2,18</td>
</tr>
<tr>
<td>3</td>
<td>3,1</td>
<td>3,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3,6</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>4,3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4,13</td>
</tr>
</tbody>
</table>
```

Campos e extensões

✓ A estrutura de documentos é útil na pesquisa
  o Restrições de campos
    ▪ P.e., data, de., etc.
  o Alguns campos mais importantes
    ▪ P.e., título
✓ Opções:
  o Separar as listas invertidas para cada tipo de campo
  o Adicionar informações sobre os campos nos postings
  o Utilizar listas extensas

Listas extensas (Extent Lists)

✓ Uma **extensão** é uma região contígua de um documento
  o As extensões podem ser utilizadas para posições do termo
  o A lista invertida regista todas as extensões para um determinado tipo de campo
  o P.e.,

```
<table>
<thead>
<tr>
<th></th>
<th>fish</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td>1,4</td>
</tr>
<tr>
<td>2</td>
<td>2,7</td>
</tr>
<tr>
<td></td>
<td>2,18</td>
</tr>
<tr>
<td></td>
<td>2,23</td>
</tr>
<tr>
<td>3</td>
<td>3,2</td>
</tr>
<tr>
<td></td>
<td>3,6</td>
</tr>
<tr>
<td>4</td>
<td>4,3</td>
</tr>
<tr>
<td></td>
<td>4,13</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1:(1,3)</td>
</tr>
<tr>
<td></td>
<td>2:(1,5)</td>
</tr>
<tr>
<td>4</td>
<td>4:(9,15)</td>
</tr>
</tbody>
</table>
```
Outras questões

- Pontuações pré-processadas nas lista invertida
  - P.e., lista para “fish” [(1:3.6), (3:2.2)], onde 3.6 é a classificação total do documento 1
  - Otimiza a rapidez mas reduz a flexibilidade

- Listas de pontuações ordenadas
  - O mecanismo de processamento da consulta pode-se concentrar apenas na parte superior de cada lista invertida, onde os documentos de maior pontuação são colocados
  - Torna-se muito eficiente para consultas com palavras únicas

Estrutura de Dados

Estrutura de dados para o dicionário de índices invertidos

- A estrutura de dados do dicionário armazena o vocabulário de termos, frequência de documentos, apontadores para cada um, lista de postings,... mas em que estrutura de dados?

```
Brutus → 1 2 4 11 31 45 173 174
Caesar → 1 2 4 5 6 16 57 132 ...
Calpurnia → 2 31 54 101
...
```

---

Página 44 de 128
Um dicionário “ingênuo”

✓ Uma estrutura de array:

<table>
<thead>
<tr>
<th>term</th>
<th>document frequency</th>
<th>pointer to postings list</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>656,265</td>
<td></td>
</tr>
<tr>
<td>aachen</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>zulu</td>
<td>221</td>
<td></td>
</tr>
</tbody>
</table>

char[20]     int     Postings*
20 bytes  4 bytes   4 bytes

✓ Como é que armazenamos o dicionário na memória de uma forma eficiente?
✓ Qual a velocidade que encontraremos os elementos no tempo de pesquisa?

Estruturas de dados do Dicionário

✓ Existem, principalmente, duas alternativas:
  o Árvores – Tree (binary tree, B-tree, Patricia tree, Trie,..)
  o Hash table
✓ Alguns sistemas de RI utilizam Árvores, outros Hashes.

Árvores de Pesquisa Binária (Binary Search Tree – BST)
BST Balanceada

✓ A BST necessita $O(\log_2 n)$ de tempo médio, mas....
  o Precisa $O(n)$ de tempo no pior dos casos, quando a árvore se assemelha a uma lista ligada desequilibrada (árvore degenerada).

✓ Para que a BST esteja balanceada, devemos verificar o balanceamento em cada inserção ou remoção

✓ BST está balanceada se:
  o A diferença entre a parte esquerda e a direita é de 0 ou 1
  o Todos os ramos se encontram balanceados

Árvore: B-tree

Definição

✓ Cada nódulo interno tem um nº de filhos no intervalo $[a,b]$ onde $a,b$ são nºs naturais, p.e., $[2,4]$. 
Árvores

✓ Mais simples: árvore binária
✓ Mais usual: B-tree
✓ Árvores que exigem ordenação de strings
✓ Prós:
  o Resolve o problema de prefixo (p.e. termos começando com hyp)
✓ Contras:
  o Mais lento: O \((\log_2 M)\) [e requer que a árvore esteja balanceada]
  o Reequilíbrio de árvores binárias pode tornar-se pesado
    ▪ Mas a árvore B-tree mitigam o problema do reequilíbrio

Tabela Hash (Hash Table)

<table>
<thead>
<tr>
<th>keys</th>
<th>buckets</th>
<th>entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Smith</td>
<td>000</td>
<td>Lisa Smith 521-8976</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>002</td>
<td></td>
</tr>
<tr>
<td>Lisa Smith</td>
<td>151</td>
<td>John Smith 521-1234</td>
</tr>
<tr>
<td></td>
<td>152</td>
<td>Sandra Dee 521-9655</td>
</tr>
<tr>
<td></td>
<td>153</td>
<td></td>
</tr>
<tr>
<td>Sam Doe</td>
<td>154</td>
<td>Ted Baker 418-4165</td>
</tr>
<tr>
<td>Sandra Dee</td>
<td>253</td>
<td>Sam Doe 521-5030</td>
</tr>
<tr>
<td>Ted Baker</td>
<td>254</td>
<td></td>
</tr>
<tr>
<td></td>
<td>255</td>
<td></td>
</tr>
</tbody>
</table>

Hashes

✓ Cada termo do vocabulário é hashed para um inteiro
✓ Prós:
  o Pesquisa é mais rápida do que numa árvore: O(1)
✓ Contras:
  o Não é fácil encontrar variantes menores:
    ▪ decisão / sentença
  o Não permite pesquisa por prefixo [recuperação tolerante]
  o Se o vocabulário crescer continuamente, é necessário, ocasionalmente, fazer a pesada operação de efetuar o hashing a tudo.

Junção de postings mais rápida: Ignorar apontadores/Ignorar listas

Junção simples do retorno (recall)

✓ Percorre simultaneamente os dois postings, num tempo linear ao número total de entradas de postings.

If the list lengths are $m$ and $n$, the merge takes $O(m+n)$ operations.

Can we do better?
Yes (if index isn't changing too fast).

Aumentar os postings ignorando apontadores (na altura da indexação)

Porquê?
✓ Para ignorar postings que não irão estar nos resultados de pesquisa.
Processamento da Query ignorando apontadores (saltos)

Suponhamos que percorremos as listas até que seja encontrado o nº 8 em ambas. Vamos alternado e avançando.

Temos os valores 41 em cima e 11 em baixo, pelo que 11 é o menor.

Mas o sucessor de 11 na lista de baixo é o valor 31, pelo que podemos ignorar os valores no intervalo, permitindo assim poupar tempo.

Onde colocamos os saltos (skips)?

Diversos cenários:

- **Mais saltos** → intervalos de salto mais pequenos ⇒ maior probabilidade de haver saltos. Mas também muitas comparações para saltar apontadores.

- **Menos saltos** → pouca comparação de apontadores, mas saltos maiores ⇒ poucos saltos de sucesso.
Colocar os saltos

✔ Simples heuristica: a cada posting de tamanho \( L \), utilizar \( \sqrt{L} \) com espaços uniformes em apontadores de salto.
  
  ○ Isto ignora a distribuição dos termos da query.

✔ É fácil caso o índice seja relativamente estático.

✔ Difícil se \( L \) for mudando por causa de atualizações.

Interseção de Postings com Apontadores de salto

```
INTERSECTWITHSKIPS(p1, p2)
1  answer ← ()
2  while p1 ≠ NIL and p2 ≠ NIL
3    do if docID(p1) = docID(p2)
4       then ADD(answer, docID(p1))
5       p1 ← next(p1)
6       p2 ← next(p2)
7    else if docID(p1) < docID(p2)
8       then if hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
9          then while hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
10             do p1 ← skip(p1)
11             p1 ← next(p1)
12          else p1 ← next(p1)
13    else if hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
14       then while hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
15          do p2 ← skip(p2)
16          else p2 ← next(p2)
17  return answer
```
Construção do Índice

✓ Como é que construímos um índice?
✓ Que estratégias podemos utilizar no que respeita à limitação da memória?

Princípios de Hardware

✓ Muitas decisões de concepção em recuperação de informação são baseadas nas características de hardware.

Comecemos por rever conceitos básicos de hardware:

✓ O acesso a dados na memória é muito mais rápido do que o acesso aos dados no disco.
✓ Pesquisa no Disco: Não são transferidos dados do disco enquanto a cabeça do disco está a posicionar-se.
✓ Portanto: Transferência de uma parte grande de dados do disco para a memória é mais rápido do que a transferência de muitas pequenas partes.
✓ Disco I/O é baseado em blocos: Leitura e escrita de blocos inteiros (ao contrário de partes menores).
✓ Servidores usados em sistemas de RI, geralmente, têm vários GB de memória principal, por vezes dezenas de GB.
✓ O espaço em disco disponível é cerca de 2-3 de maior magnitude.
✓ A tolerância a falhas torna-se muito caro: é muito mais barato usar muitas máquinas regulares, em vez de uma máquina tolerante a falhas.

Princípios de Hardware (obsoleto)

<table>
<thead>
<tr>
<th>symbol</th>
<th>statistic</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>average seek time</td>
<td>5 ms = 5 x 10^{-3} s</td>
</tr>
<tr>
<td>b</td>
<td>transfer time per byte</td>
<td>0.02 μs = 2 x 10^{-8} s</td>
</tr>
<tr>
<td></td>
<td>processor’s clock rate</td>
<td>10^9 s^{-1}</td>
</tr>
<tr>
<td>p</td>
<td>low-level operation</td>
<td>0.01 μs = 10^{-8} s</td>
</tr>
<tr>
<td></td>
<td>(e.g., compare &amp; swap a word)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>size of main memory</td>
<td>several GB</td>
</tr>
<tr>
<td></td>
<td>size of disk space</td>
<td>1 TB or more</td>
</tr>
</tbody>
</table>
Coleção Reuters RCV1

- Coleção bem conhecida na RI
- Contém um ano de notícias da Reuters
- Documento típico:

Estatísticas Reuters RCV1

- symbol: statistic
- N: documents
- L: avg. # tokens per doc
- M: terms (= word types)
- avg. # bytes per token (incl. spaces/punct.)
- avg. # bytes per token (without spaces/punct.)
- avg. # bytes per term
- non-positional postings

<table>
<thead>
<tr>
<th>symbol</th>
<th>statistic</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>documents</td>
<td>800,000</td>
</tr>
<tr>
<td>L</td>
<td>avg. # tokens per doc</td>
<td>200</td>
</tr>
<tr>
<td>M</td>
<td>terms (= word types)</td>
<td>400,000</td>
</tr>
<tr>
<td></td>
<td>avg. # bytes per token (incl. spaces/punct.)</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>avg. # bytes per token (without spaces/punct.)</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>avg. # bytes per term</td>
<td>7.5</td>
</tr>
<tr>
<td></td>
<td>non-positional postings</td>
<td>100,000,000</td>
</tr>
</tbody>
</table>

Recordar a construção do índice

- Os documentos são analisados para extrair as palavras e são armazenadas com a informação do Document ID.
Passo fundamental

✓ Finalizadas a análise a todos os documentos, o índice invertido é ordenado por termos.

We focus on this sort step.
We have 100M items to sort.

Escalabilidade na construção do Índice

✓ A construção do Índice na memória não é escalável.
✓ Então, como podemos construir índices de coleções grandes?
  o Temos que ter em conta as limitações do hardware vistas anteriormente
  o Memória, discos, processador, etc.
Indexação de blocos baseada em ordenação

- Assim que vamos construindo o Índice, os documentos são analisados um a um.
- Os postings finais de qualquer termos estão incompletos até o índice estar pronto.
- Se considerarmos 12 bytes por cada entrada de posting não-posicional (termo, doc, freq), exige muito espaço para coleções grandes.
- T = 100,000,000 no caso do RCV1
  - Então... podemos trabalhar apenas com a memória hoje em dia, mas as coleções típicas são muito maiores.
  - Por exemplo, o New York Times fornece um índice de > 150 anos de notícias.

Conclusão: há necessidade de armazenar as informações do Índice no disco em vários momentos intermédios.

Algoritmo BSBI – Blocked Sort-Based Indexing

- 12-byte (4+4+4) registos (termo, doc, freq).
- Os registos são gerados enquanto os documentos são analisados.
- É necessário agora ordenar 100M com registos de 12-bytes por cada termo.
- Definir um bloco ~ 10M de registos
  - É possível colocar alguns em memória
  - Começamos com 10 blocos.
- Ideia principal do algoritmo
  - Acumular postings para cada bloco, ordená-los e escrevê-los nos disco.
  - Posteriormente, juntar os blocos num maior devidamente ordenado.
Como juntar os diversos blocos ordenados?

- Podemos fazer junções binárias, através de uma árvore de junções de $\log_2 10 = 4$ camadas.
- Em cada camada, é lido na memória blocos de 10M, juntar e escrever novamente.

- É mais eficiente fazer $n$-vezes junções, onde estamos a ler de todos os blocos em simultâneo.
- Proporcionando a leitura na memória de partes com tamanho decente para cada bloco e depois escrevê-lo no disco, não estamos a sobrecarregar os acessos ao disco.

Recordar o problema com o algoritmo de ordenação

- A ideia era trabalhar com o dicionário na memória.
- É necessário o dicionário (que cresce dinamicamente) ordenado para implementar o termo no mapa dos termID.
- Atualmente, podemos trabalhar com o termo, docID, postings em vez de termID, docID postings...
- ... mas os ficheiros intermédios ficam muito extensos (podemos acabar com um método escalável, mas torna-se um método muito lento na construção do índice).
SPIMI: Indexação de único passo em memória
(Single-pass in-memory indexing)

Ideia principal 1
Gerar dicionários separados para cada bloco – não é necessário manter o termo - mapeamento termID através dos blocos.

Ideia principal 2
Não ordena. Acumula os postings em listas de postings assim que vão aparecendo.

- Com estas duas ideias, podemos gerar um índice invertido completo para cada bloco.
- Estes índices separados podem ser juntos num índice maior.
- A junção de blocos é análoga ao BSBI

Algoritmo SPIMI-Invert

```
SPIMI-INVERT(token_stream)
1    output_file = NEWFile()
2    dictionary = NEWHash()
3    while (free memory available)
4        do token = next(token_stream)
5            if term(token) ≠ dictionary
6                then postings_list = ADDToDictionary(dictionary, term(token))
7                else postings_list = GETPostingsList(dictionary, term(token))
8            if full(postings_list)
9                then postings_list = DOUBLEPostingsList(dictionary, term(token))
10               ADDToPostingsList(postings_list, docID(token))
11               sorted_terms = SORTTerms(dictionary)
12    WRITEBlockToDisk(sorted_terms, dictionary, output_file)
13    return output_file
```

- A diferença entre o BSBI e SPIMI é que o SPIMI adiciona o endereçamento diretamente na lista de endereçamento (linha 10), ao contrário de devolver todos os pares de termID-docID e depois ordená-los como faz o BSBI.
- Cada lista de endereçamento é dinâmica e é imediatamente disponibilizada na coleção de endereçamento.
- O SPIMI tem duas vantagens: é mais rápido porque não requer ordenação e conserva a memória porque mantém um caminho da lista de endereçamento onde está o termo, então a lista de termID não é armazenada.
Indexação Distribuída (Distributed indexing)

- Para índices de escalabilidade web
  - É necessário utilizar um cluster de computação distribuída
- Máquinas individuais não são eficientes
  - Podem falhar ou fazer com que o processo fique muito lento
- Como resolvemos isto? Como podemos explorar um conjunto de máquinas para este tema?

Google data centers

- Data centres da Google contêm na maioria máquinas de serviço.
- Os Data centers estão distribuídos pelo mundo
- Estima-se que têm um total de 1 milhão de servidores, 3 milhões de processadores/núcleos (Gartner 2007)
- Estima-se que o Google instala 100,000 servidores a cada trimestre.
  - Baseando-se em gastos em cerca de 200–250 milhões de dólares por ano
- Estes valores representam cerca de 10% da capacidade computacional de todo o mundo.

Indexação Distribuída

- Mantém um máquina principal direta com o trabalho de indexação – considerada “segura”.
- Divide-se a indexação em conjuntos (paralelos) de tarefas.
- A máquina principal atribui tarefas a máquinas livres que estejam na fila

Tarefas paralelas (Parallel tasks)

- Iremos utilizar dois conjuntos de tarefas paralelas
  - Analisadores
  - Inversores
- Separa a coleção da entrada de documentos em diversas partes
- Cada parte é um subconjunto de documentos (correspondentes a blocos no BSBI/SPIMI)
**Analisadores (Parsers)**

- O computador principal (mestre) atribui uma parte a uma máquina livre para analisar
- Essa máquina lê cada documento e gera os pares (termo, doc)
- O analisador escreve os pares em j partições.
- Cada partição corresponde a um intervalo de letras iniciais.
  - P.e.: (a-f, g-p, q-z), onde j= 3.
- Para finalizar, veremos o inversor do Índice

**Inversores (Inverters)**

- Um inversor coleciona todos os pares (termo,doc) (= postings) para uma partição de termos.
- Ordena-os e escreve-os nas listas de postings

**Ciclo dos dados (Data flow)**

- Na figura acima as partições dos termos são a primeira letra de cada um: a-f, g-p, q-z e j=3 onde j é a quantidade de partições.
- As partições dos termos são definidas por uma pessoa que opera o sistema de indexação.
- Os parsers então escrevem segmentos de ficheiros, um para cada partição de termo.
- Cada partição de termo corresponde a r ficheiros de segmentos, onde r é o número de parsers.
✓ Com todos os valores (docIDs) para uma chave dada (termID) numa lista de tarefas dos invertes numa fase de reduce, o master designa cada partição termo para um inversor diferente – e, como no caso dos parsers, reatribui as partições dos termos, no caso de falha ou inversores lentos.
✓ Cada partição termo é processada por um inversor.
✓ Finalmente, a lista de valores é classificada para cada chave e escrita na lista de endereçamentos final devidamente ordenada.

MapReduce
✓ O algoritmo de construção do Índice visto anteriormente é uma instância do MapReduce.
✓ “MapReduce é um robusto e simples framework conceptual para a computação distribuída, onde não é necessário escrever código para a distribuição das partes.” (Dean and Ghemawat 2004)
✓ Eles descrevem o Sistema de Indexação da Google (ca. 2002) num nº consistente de fases, cada uma implementada no MapReduce.
✓ A construção do Índice tem apenas uma fase.
✓ Outra fase: transformar um índice particionado por termos num índice particionado por documentos.
  o **Particionado por termos**: uma máquina lida com um subconjunto de termos
  o **Particionado por documentos**: uma máquina lida com o subconjunto de documentos
✓ A maioria dos motores de pesquisa utiliza um índice particionado por documentos
  o Apresenta melhor balanceamento de carga, etc.

Esquema para a construção do Índice no MapReduce
✓ Esquema do mapa e funções de redução

\[
\text{map: input } \rightarrow \text{ list(k, v) } \quad \text{reduce: (k, list(v)) } \rightarrow \text{ output}
\]

✓ Instanciação do esquema da construção n

\[
\begin{align*}
\text{map: web collection } & \rightarrow \text{ list(termID, docID)} \\
\text{reduce: (} & \text{<termID1, list(docID)>}, \text{ <termID2, list(docID)>}, \ldots) \\
& \rightarrow (postings list1, postings list2, \ldots)
\end{align*}
\]

✓ Exemplo de construção do Índice
Indexação Dinâmica (Dynamic indexing)

✓ Ate agora, assumimos que as coleções eram estáticas
✓ No entanto, é raro que elas o sejam:
   o Novos documentos aparecem de tempos a tempos para serem inseridos
   o Os documentos podem ser eliminados e modificados
✓ Isto significa que o dicionário e as listas de postings também terão que ser continuamente alteradas
   o Atualização de postings dos termos já presentes no dicionário
   o Inserção de novos termos no dicionário

Medida simplista (Simplest approach)

✓ Manter o índice principal “grande”
✓ Os novos documentos vão para um índice auxiliar “pequeno”
✓ Pesquisa em ambos juntando os resultados
✓ Periodicamente, reindexar o auxiliar no índice principal

Problemas com Índices principais e auxiliares

✓ Problema frequentes na fusão – alterações em muitos dados
✓ Fraco desempenho durante a fusão
✓ Na verdade:
   o Misturando o índice auxiliar no índice principal é eficiente se mantivermos um ficheiro separado para cada lista de postings.
   o Juntar é o mesmo que um acréscimo simples.
   o Mas, então, seria preciso uma grande quantidade de ficheiros - ineficiente para O/S.

Assumir para o resto da matéria: O índice é um ficheiro grande.
Junção Logarítmica (Logarithmic merge)

- Manter uma série de índices, cada um com o dobro do tamanho do anterior
- Manter o menor ($Z_0$) em memória
- O maior ($l_0, l_1, ...$) no disco
- Se $Z_0$ ficar muito grande (> $n$), escrever no disco como $l_0$
- Ou então juntar com o $l_0$ (caso o $l_0$ já exista) como $Z_1$
- Cada escrita junta o $Z_1$ para o disco como $l_1$ (caso $l_1$ não exista)
- Ou então juntar com o $l_1$ (caso o $l_1$ já exista) como $Z_2$
- etc.

L\textsc{MergeAddToken}(indexes, $Z_0$, token)
\begin{algorithmic}
  \State $Z_0 \leftarrow \text{Merge}(Z_0, \{\text{token}\})$
  \If{$|Z_0| = n$}
    \For{$i \leftarrow 0$ \KwTo $\infty$}
      \If{$l_i \in \text{indexes}$}
        \State $Z_{i+1} \leftarrow \text{Merge}(l_i, Z_i)$
        \State ($Z_{i+1}$ is a temporary index on disk.)
        \State indexes $\leftarrow$ indexes $\setminus \{l_i\}$
      \Else
        \State $l_i \leftarrow Z_i$ \Comment{$Z_i$ becomes the permanent index $l_i$.}
        \State indexes $\leftarrow$ indexes $\cup \{l_i\}$
        \State \textbf{break}
      \EndIf
    \EndFor
  \EndIf
  \State $Z_0 \leftarrow \emptyset$
\end{algorithmic}

Log\textsc{arithmicMerge}()
\begin{algorithmic}
  \State $Z_0 \leftarrow \emptyset$ \Comment{$Z_0$ is the in-memory index.}
  \State indexes $\leftarrow \emptyset$
  \While{true}
    \State \textbf{do L\textsc{MergeAddToken}(indexes, $Z_0$, getNextToken())}
  \EndWhile
\end{algorithmic}

- Índice auxiliar e Índice principal: o tempo de construção do índice é $O(T)$ dado que cada posting é considerado em cada junção.
- Junção logarítmica: cada posting é junto $O(\log T)$ vezes, então a é complexidade é $O(T \log T)$

**Conclusão:** Então, a junção logarítmica é mais eficiente na construção do índice.

- Mas o processamento da query requer a junção de $O(\log T)$ indices
  - Onde $O(1)$ caso tenhamos apenas um índice auxiliar e principal
Outras questões com vários índices

- As estatísticas de coleções grandes são difíceis de manter

- Por exemplo, quando falamos de spell-correction: qual das várias alternativas válidas são apresentadas ao utilizador?
  - Podemos dizer que escolheríamos a que tivesse a maioria de acessos

- Como manter os topos com vários índices e vetores de bits inválidos?
  - Uma possibilidade: ignorar tudo, mas o principal índice ordenado de uma forma

Indexação dinâmica nos motores de pesquisa

- Todos os grandes motores de pesquisa fazem indexação dinâmica

- Os índices apresentam mudanças incrementais frequentes
  - As notícias, blogs, novos tópicos nas páginas da web

- Mas (às vezes / geralmente) também convém reconstruir o índice a partir do zero (periodicamente)
  - O processamento das queries é transferido para o novo índice e o índice antigo é eliminado.

Outras ordenações de Índices

- Índices posicionais
  - O mesmo problema de ordenação.... Demasiado grande

- Construir índices n-grams:
  - Durante a análise do texto, são enumerados os n-grams.
  - Para cada n-gram, é necessário apontadores para todos os termos do dicionário que o contenham – os “postings”
  - Convém notar que todas as “entradas posting” irão aparecer repetidas na análise dos documentos – necessita de um hashing eficiente para controlar essa questão.
    - P.e.: suponhamos que o trograma **uou** ocorre no termo **deciduous** e que será descoberto em cada ocorrência do termo **deciduous**
    - É necessário processar apenas uma vez.
Introdução

✔ Enorme variedade de palavras utilizadas no texto.

✔ Muitas características das estatísticas da ocorrência de palavras são previsíveis.
  - Exemplo: Distribuição da contagem de palavras;

✔ Modelos de recuperação e algoritmos de ranking dependem largamente nas propriedades estatísticas das palavras.
  - Exemplo: Palavras importantes ocorrem muitas vezes em documentos mas não de alta frequência nas coleções;

Lei de Zipf

✔ A distribuição da frequência de palavras é altamente distorcida:
  - Algumas palavras ocorrem frequentemente, mas muitas palavras raramente ocorrem;
  - Exemplo: As duas palavras mais comuns (“the”, “of”) representam cerca de 10% de todas as ocorrências de palavras em documentos;

✔ Lei de Zipf:
  - Noção que o rank ou classificação (r) de uma palavra multiplicada pela sua frequência (f) resulta, aproximadamente, numa constante (k);
    - Assumindo que as palavras são classificadas (ranked) em ordem de frequência descendente.
  - i.e., \( rf \approx k \) ou \( rPr \approx c \), onde \( Pr \) é a probabilidade de ocorrência de uma palavra e \( c \approx 0.1 \) para a língua inglesa.
Consequências de Zipf

✓ Se o termo mais frequente ("the") ocorrer \( f_1 \) vezes
  
  - Então o segundo termo mais frequente ("of") ocorre \( f_1/2 \) vezes;
  
  - O terceiro termo mais frequente ("and") ocorre \( f_1/3 \) vezes...

✓ Equivalente: \( f_i = k/r \) onde \( k \) é o factor normalizante, ou seja:
  
  - \( \log f_i = \log k - \log r \) (sendo \( r \) o rank da palavra)
  
  - Relação linear entre o \( \log f_i \) e \( \log r \)

✓ Uma relação de lei poderosa

2 - Lei de Zipf para o Reuters RCV1 (\( \log f_i - \log r \))
Estatísticas da coleção de notícias (AP89)

- Número total de documentos: 84,678
- Número total de ocorrência de palavras: 39,749,179
- Tamanho do vocabulário: 198,763
- Palavras que ocorrem > 1000 vezes: 4,169
- Palavras que ocorrem uma única vez: 70,064

<table>
<thead>
<tr>
<th>Palavra</th>
<th>Frequência</th>
<th>r</th>
<th>Pr(%)</th>
<th>r.Pr (calc. Zipf)</th>
</tr>
</thead>
<tbody>
<tr>
<td>assistant</td>
<td>5,095</td>
<td>1,021</td>
<td>.013</td>
<td>.013</td>
</tr>
<tr>
<td>sewers</td>
<td>100</td>
<td>17,110</td>
<td>2.56 × 10⁻⁴</td>
<td>0.04</td>
</tr>
<tr>
<td>toothbrush</td>
<td>10</td>
<td>51,555</td>
<td>2.56 × 10⁻⁵</td>
<td>0.01</td>
</tr>
<tr>
<td>hazmat</td>
<td>1</td>
<td>166,945</td>
<td>2.56 × 10⁻⁶</td>
<td>0.04</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word</th>
<th>Freq.</th>
<th>r</th>
<th>P_r (%)</th>
<th>r.P_r (calc. Zipf)</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>2,420,778</td>
<td>1</td>
<td>6.49</td>
<td>0.065</td>
</tr>
<tr>
<td>of</td>
<td>1,045,733</td>
<td>2</td>
<td>2.80</td>
<td>0.056</td>
</tr>
<tr>
<td>to</td>
<td>968,882</td>
<td>3</td>
<td>2.60</td>
<td>0.078</td>
</tr>
<tr>
<td>a</td>
<td>892,429</td>
<td>4</td>
<td>2.39</td>
<td>0.096</td>
</tr>
<tr>
<td>and</td>
<td>865,644</td>
<td>5</td>
<td>2.32</td>
<td>0.120</td>
</tr>
<tr>
<td>in</td>
<td>847,825</td>
<td>6</td>
<td>2.27</td>
<td>0.140</td>
</tr>
<tr>
<td>said</td>
<td>504,937</td>
<td>7</td>
<td>1.35</td>
<td>0.095</td>
</tr>
<tr>
<td>for</td>
<td>363,865</td>
<td>8</td>
<td>0.98</td>
<td>0.078</td>
</tr>
<tr>
<td>that</td>
<td>347,072</td>
<td>9</td>
<td>0.93</td>
<td>0.084</td>
</tr>
<tr>
<td>was</td>
<td>293,027</td>
<td>10</td>
<td>0.79</td>
<td>0.079</td>
</tr>
<tr>
<td>on</td>
<td>291,947</td>
<td>11</td>
<td>0.78</td>
<td>0.086</td>
</tr>
<tr>
<td>he</td>
<td>250,912</td>
<td>12</td>
<td>0.67</td>
<td>0.081</td>
</tr>
<tr>
<td>is</td>
<td>245,843</td>
<td>13</td>
<td>0.65</td>
<td>0.086</td>
</tr>
<tr>
<td>with</td>
<td>223,846</td>
<td>14</td>
<td>0.60</td>
<td>0.084</td>
</tr>
<tr>
<td>at</td>
<td>210,064</td>
<td>15</td>
<td>0.56</td>
<td>0.085</td>
</tr>
<tr>
<td>by</td>
<td>209,366</td>
<td>16</td>
<td>0.56</td>
<td>0.090</td>
</tr>
<tr>
<td>it</td>
<td>195,621</td>
<td>17</td>
<td>0.52</td>
<td>0.089</td>
</tr>
<tr>
<td>from</td>
<td>189,451</td>
<td>18</td>
<td>0.51</td>
<td>0.091</td>
</tr>
<tr>
<td>as</td>
<td>181,714</td>
<td>19</td>
<td>0.49</td>
<td>0.093</td>
</tr>
<tr>
<td>be</td>
<td>157,300</td>
<td>20</td>
<td>0.42</td>
<td>0.084</td>
</tr>
<tr>
<td>were</td>
<td>153,913</td>
<td>21</td>
<td>0.41</td>
<td>0.087</td>
</tr>
<tr>
<td>an</td>
<td>152,576</td>
<td>22</td>
<td>0.41</td>
<td>0.090</td>
</tr>
<tr>
<td>have</td>
<td>149,749</td>
<td>23</td>
<td>0.40</td>
<td>0.092</td>
</tr>
<tr>
<td>his</td>
<td>142,235</td>
<td>24</td>
<td>0.38</td>
<td>0.092</td>
</tr>
<tr>
<td>but</td>
<td>140,830</td>
<td>25</td>
<td>0.38</td>
<td>0.094</td>
</tr>
</tbody>
</table>

3 - Top das 50 palavras de AP89
Crescimento do vocabulário – Lei de Heap

✓ À medida que o corpus cresce, também cresce o tamanho do vocabulário;
  o Quando o corpus já é grande, poucas são as novas palavras.
✓ Relação observada (Lei de Heap)

\[ v = k \cdot n^\beta \]

onde:
  o \( v \) é o tamanho do vocabulário (número de palavras únicas);
  o \( n \) é o número de palavras no corpus;
  o \( k, \beta \) são os parâmetros que variam para cada corpus (valores típicos dados são \( 10 \leq k \leq 100 \) e \( \beta \approx 0.5 \)).
Previsões na lei de Heap

- Previsões para colecções TREC (Text Retrieval Conference) são mais precisas para um grande número de palavras;
  - Exemplo: Primeiras 10,879,522 palavras da colecção AP89 digitalizadas;
  - Previsão de 100,151 palavras únicas;
  - Número real de 100,024 palavras únicas.
- Previsões para um número menor de palavras (< 1000) são bastante piores.
Lei de Heap (log - log)

✓ Para o RCV1 (ver gráfico abaixo), a linha tracejada $\log_{10} M = 0.49 \log_{10} T + 1.64$ é a que melhor se adapta, com menos quadrados.

✓ Consequentemente, $M = 10^{1.64} * T^{0.49}$, então $k = 10^{1.64}$ que é $\approx 44$ e $b = 0.49$.

✓ Isto é uma boa previsão empírica para o Reuters RCV1.

✓ Para os primeiros 1,000,000 tokens, a lei prevê 38,323; na realidade são 38,365.
AULA 5 – Classificação (Ranking) e Modelo de Espaço Vectorial

Agenda

✔ Recuperação classificada (Ranked retrieval)
✔ Classificar Documentos (Scoring documents)
✔ Term frequency
✔ Estatísticas da coleção (Collection statistics)
✔ Esquemas de pesagem (Weighting schemes)
✔ Classificação do espaço vectorial (Vector space scoring)

Recuperação classificada

✔ Até agora, todas as nossas pesquisas (queries) têm sido booleanas (os documentos ou são compatíveis ou não);
✔ Bom para utilizadores avançados com conhecimentos precisos das suas necessidades e da coleção (bom também para as aplicações, consumindo milhares de resultados);
✔ Não ideal para a maioria dos utilizadores: estes são incapazes de escrever pesquisas (queries) booleanas – ou são, mas não o querem fazer – ou então não querem vaguear por milhares de resultados (particularmente verdade para pesquisas na web).

Problema com pesquisa booleana: bonança ou escassez

✔ Queries booleanas resultam frequentemente em poucos resultados (=0) ou demasiados resultados (1000+):
  o Pesquisa 1: “standard user dlink 650” - 200,000 resultados;
  o Pesquisa 2: “standard user dlink 650 no card found” - 0 resultados.
✔ É preciso muito conhecimento para escrever uma query que consiga produzir resultados que possam ser geridos
  o O E devolve poucos; o OU devolve muitos
✔ Solução?
Recuperação classificada

✓ **Modelo de recuperação classificada**
  - sistema devolve uma ordenação sobre os documentos (mais relevantes) da colecção que digam respeito à query;

✓ **Queries de texto livre**
  - em vez de uma linguagem de query de operadores e expressões, a query do utilizador é apenas uma ou mais palavras na linguagem humana.
  - Por princípio, existem duas escolhas separadas aqui; mas na prática, ambos os modelos têm sido associados.

**Bonança ou escassez: não é um problema numa recuperação classificada**

✓ Quando um sistema produz um conjunto composto por um resultado classificado, conjuntos maiores de resultados não são um problema:
  - O tamanho do conjunto de resultados não é um problema;
  - Apenas se mostra os resultados do top k (≈ 10);
  - Não se sobrecarrega o utilizador;

**Nota:** isto pensando que o algoritmo de classificação funciona.

Pontuações como sendo a base da recuperação classificada

✓ Queremos retornar, em ordem, os documentos mais prováveis a serem relevantes para o utilizador;
✓ Como é que organizamos por classificação os documentos de uma colecção que digam respeito a uma query?
✓ Dar uma pontuação, de [0, 1] por exemplo, a cada documento.
✓ Esta pontuação indica o quanto é que um documento e uma query se assemelham.

Pontuações de semelhanças entre *Query*-Documento

✓ Precisamos de uma maneira de atribuir uma pontuação a um par query/documento;
  - Começando com uma query de um só termo:
✓ Se a query não ocorre no documento, a pontuação deve ser 0. Contrariamente, quanto maior a frequência do termo da query no documento, maior deve ser a pontuação.
Take 1: Coeficiente de Jaccard

✓ Uma medida usualmente usada de sobreposição de dois conjuntos A e B é:

\[
\text{jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|} \\
\text{jaccard}(A, A) = 1 \\
\text{jaccard}(A, B) = 0 \text{ if } A \cap B = 0
\]

✓ A e B não têm de ter o mesmo tamanho.
✓ Atribui sempre um número entre 0 e 1.

Coeficiente de Jaccard: exemplo de pontuação

✓ Qual é a pontuação de semelhança query-documento que o coeficiente vai calcular para cada um dos documentos?

- **Pesquisa:** ides of march
- **Documento 1:** ceaser died in march
- **Documento 2:** the long march

**Resposta:** 1/6 para o primeiro e 1/5 para o segundo, sendo necessário ver o número de termos da reunião entre query/documento, valor que será o denominador da fracção, e ver o número de termos da query que aparecem no documento (intersecção), sendo esse o numerador.

Problemas com o coeficiente de Jaccard (desvantagens)

✓ Não considera a frequência do termo (quantas vezes o termo aparece no documento);
✓ Termos raros numa coleção são mais informativos que os mais frequentes – o Jaccard não toma isto em consideração;
✓ É necessário uma maneira mais sofisticada de normalização para comprimento.
✓ Mais tarde será usada a fórmula \( \frac{|A \cap B|}{\sqrt{|A \cup B|}} \) que substituirá a que é utilizada de momento \( \frac{|A \cap B|}{|A \cup B|} \) para a normalização de comprimento.
Matriz binária de incidência termo-documento

<table>
<thead>
<tr>
<th></th>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>worser</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Each document is represented by a binary vector $\in \{0,1\}^{|V|}$

Matrizes de contagem termo-documento

- Considere o número de ocorrências de um termo num documento:
  - Cada documento é um vetor de contagem em $\mathbb{N}^{|V|}$: uma coluna abaixo

<table>
<thead>
<tr>
<th></th>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>157</td>
<td>73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Brutus</td>
<td>4</td>
<td>157</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>232</td>
<td>227</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>57</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>worser</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Modelo “saco de palavras”

- Representação em vector não considera a ordenação das palavras num documento:
  - “John is quicker than Mary” e “Mary is quicker than John” têm o mesmo vector.
- Neste sentido é um passo atrás, dado que um índice posicional seria capaz de distinguir estes 2 documentos.

**Nota:** Como recuperar informação posicional será visto mais abaixo.
Frequência de termo tf

✓ A frequência de termo $tf_{t,d}$ de termo $t$ num documento $d$ é definido como o número de vezes que $t$ ocorre em $d$.
✓ Queremos usar tf para o cálculo de pontuações de semelhança Query-Documento. Como?
✓ Frequência de termos não normalizada (em bruto) não é o que queremos:
  o Um documento com 10 ocorrências de um termo é mais relevante que um documento com 1 ocorrência de termo.
  o Mas não 10 vezes mais relevante.
✓ A relevância não aumenta proporcionalmente com a frequência do termo.

Peso log-frequência

✓ Um peso de log-frequência de um termo $t$ em $d$ é

$$
\nu^t_d = \begin{cases}
1 + \log_{10} tf_{t,d}, & \text{if } tf_{t,d} > 0 \\
0, & \text{otherwise}
\end{cases}
$$

$0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 1.3, 10 \rightarrow 2, 1000 \rightarrow 4, \text{ etc.}$

✓ Pontuação para um par documento-query: somatório de termos $t$ em ambos $q$ (query) e $d$ (documento):

$$
\text{score} = \sum_{t \in q \cap d} (1 + \log \nu^t_d)
$$

✓ A pontuação é zero se nenhum dos termos da query estiver presente do documento.

Frequência de documentos (document frequency)

✓ Termos raros são mais informativos que termos frequentes (de relembrar as stopwords);
✓ Considere um termo da query que é raro na coleção (ex., “blackjack”)
✓ Um documento que contenha este termo provavelmente será relevante para a query blackjack.

Queremos um peso alto para termos raros como estes.

✓ Considere um termo da query frequente na coleção (ex., “trabalhar”, “estudar”)
✓ Um documento que contenha tal termo provavelmente será mais relevante que um documento que não o contenha.
Mas não é um indicador claro de relevância.
Para estes termos, queremos pesos altos positivos, mas mais baixos que os de termos raros.
Iremos usar a frequência de documentos para capturar isto.

**Peso idf (idf weight)**

- $df_t$ é a frequência de documentos de $t$: o número de documentos que contêm $t$.
  - $df_t$ é uma medida inversa do grau de informação de $t$
  - $df_t \leq N$
- Nós definimos $idf$ (frequência de documentos inversa) de $t$ como
  \[
  idf_t = \log_{10}(N/df_t)
  \]
- Nós usamos $\log(N/df_t)$ em vez de $N/df_t$ para amortecer o efeito do $idf$

**Exemplo de idf**

- Supondo que $N = 1$ milhão:

<table>
<thead>
<tr>
<th>term</th>
<th>$df_t$</th>
<th>$idf_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sabotia</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>animal</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>sunday</td>
<td>1,000</td>
<td>3</td>
</tr>
<tr>
<td>fly</td>
<td>10,000</td>
<td>2</td>
</tr>
<tr>
<td>under</td>
<td>100,000</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>1,000,000</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
idf_t = \log_{10}(N/df_t)
\]

*There is one idf value for each term $t$ in a collection.*

**Efeito do idf no ranking**

- O idf **não tem nenhum** efeito no ranking de queries **de um termo**.
  - Apenas afecta com pelo menos 2 termos.
  - Para uma query “capricious person”, o peso por $idf$ faz com que as ocorrências do termo “capricious” contem muito mais no ranking de documentos final que ocorrências de “person”.


---

Página 74 de 128
Frequência de documentos vs coleção (document frequency vs collection)

✓ A frequência de coleção de \( t \) é o número de ocorrências de \( t \) na coleção, contando ocorrências múltiplas.
✓ Exemplo:

<table>
<thead>
<tr>
<th>Word</th>
<th>Collection frequency</th>
<th>Document frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>insurance</td>
<td>10440</td>
<td>3997</td>
</tr>
<tr>
<td>try</td>
<td>10422</td>
<td>8760</td>
</tr>
</tbody>
</table>

✓ Qual palavra é o melhor termo para pesquisa (e deveria receber maior peso)?

**Resposta**: Insurance, devido à menor quantidade de documentos em que está presente.

**Peso por tf-idf**

✓ O peso \( tf-idf \) de um termo é o produto do seu peso \( tf \) e do seu peso \( idf \):

\[
  w_{t,d} = (1 + \log tf_{t,d}) \times \log_{10} \left( N / df_t \right)
\]

✓ É o melhor esquema de atribuição de peso conhecido em recuperação de informação
✓ Aumenta com o número de ocorrências dentro de um documento e com a raridade do termo na coleção.

**Ranking final dos documentos para a query**

\[
  \text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}
\]
Binario → contagem → matriz de pesos

<table>
<thead>
<tr>
<th></th>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>5.25</td>
<td>3.18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.35</td>
</tr>
<tr>
<td>Brutus</td>
<td>1.21</td>
<td>6.1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>8.59</td>
<td>2.54</td>
<td>0</td>
<td>1.51</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1.54</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>2.85</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>1.51</td>
<td>0</td>
<td>1.9</td>
<td>0.12</td>
<td>5.25</td>
<td>0.88</td>
</tr>
<tr>
<td>worser</td>
<td>1.37</td>
<td>0</td>
<td>0.11</td>
<td>4.15</td>
<td>0.25</td>
<td>1.95</td>
</tr>
</tbody>
</table>

Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^n$

**Documentos como vectores**

✓ Temos então um vetor espacial multi-dimensional, em que os termos são os eixos do espaço, documentos são pontos ou vectores neste espaço.

✓ Bastante-dimensional: dezenas de milhões de dimensões quando se aplica isto a um motor de pesquisa online.

✓ Este vectores são muitos esparsos – muitas entradas são zero.

**Queries como vectores**

✓ **Ideia principal 1**: fazer o mesmo para queries: representá-las como vectores no espaço;

✓ **Ideia principal 2**: classificar documentos de acordo com a sua proximidade à query neste espaço.

✓ **Proximidade = Similaridade entre vectores** (queremos usar isto para nos distanciarmos da aproximação radicalista do modelo booleano).

✓ Em vez disto, pretendemos classificar documentos mais relevantes com maiores valores que documentos menos relevantes.

**Formalizar a proximidade de vector espacial**

✓ **Primeira tentativa:**
  
  o distânci a entre dois pontos (distância entre os dois pontos finais no vector).

✓ **Distância euclidiana?**
  
  o Má ideia porque é grande para vectores de diferentes tamanhos.
Exemplo:

A distância é grande apesar das similaridades entre ambos os vectores q e d2.

Usar ângulo em vez de distância

Cálculo mental: pegamos num documento d e anexamo-lo a ele mesmo, chamando-lhe d*.

- Semanticamente, têm o mesmo conteúdo
- A distância euclidiana pode ser bastante grande entre os dois.
- O ângulo seria 0, correspondendo ao máximo de similaridade.

Ideia principal: Classificar documentos usando o ângulo com query.

De ângulos para cosenos

- As seguintes noções são equivalentes:
  - Classificar documentos em ordem decrescente do ângulo entre query e documento
  - Classificar documentos em ordem crescente do coseno (query, documento)
- O coseno é uma função decrescente no intervalo [0º, 180º].

Normalização do comprimento

- Os vectores podem ser normalizados (a nível de comprimento), dividindo cada um dos seus componentes pelo seu comprimento, usando a norma:

  \[
  \| \mathbf{x} \|_2 = \sqrt{\sum x_i^2}
  \]

  \(L_2\) norm:
Desta divisão pela norma L2, fazemos dele um vetor (na superfície da unidade hiperesfera) unitário.

Efeito nos dois documentos $d$ e $d^\ast$ (d anexado a si mesmo) anteriores: eles possuem vetores idênticos depois da normalização

- documentos longos/curtos têm agora pesos comparáveis.

### Coseno($query$, documento)

Para vetores de comprimento normalizado, coseno de similaridade é simplesmente o produto vectorial (ou produto escalar)

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

para $q_i$ e $d_i$ vetores normalizados.

**Onde:**

- $q_i$ é o peso tf-idf do termo i na query
- $d_i$ é o peso tf-idf do termo i no documento

**Coseno para vetores comprimento-normalizados**

- Para vetores de comprimento normalizado, coseno de similaridade é simplesmente o produto vectorial (ou produto escalar)

$$\cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_{i=1}^{|V|} q_i d_i$$
Coseno de similaridade ilustrado

Coseno de similaridade entre 3 documentos

Quão similares são as novelas

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>affection</td>
<td>115</td>
<td>58</td>
<td>20</td>
</tr>
<tr>
<td>jealous</td>
<td>10</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>gossip</td>
<td>2</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>wuthering</td>
<td>0</td>
<td>0</td>
<td>38</td>
</tr>
</tbody>
</table>

Frequência de termos (contagem)

Nota: Para simplificar este exemplo, nós não fizemos a ponderação idf.

Ponderação Log frequência

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>affection</td>
<td>3.06</td>
<td>2.76</td>
<td>2.30</td>
</tr>
<tr>
<td>jealous</td>
<td>2.00</td>
<td>1.85</td>
<td>2.04</td>
</tr>
<tr>
<td>gossip</td>
<td>1.30</td>
<td>0</td>
<td>1.78</td>
</tr>
<tr>
<td>wuthering</td>
<td>0</td>
<td>0</td>
<td>2.58</td>
</tr>
</tbody>
</table>

Depois da normalização de comprimento

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>affection</td>
<td>0.789</td>
<td>0.832</td>
<td>0.524</td>
</tr>
<tr>
<td>jealous</td>
<td>0.515</td>
<td>0.555</td>
<td>0.465</td>
</tr>
<tr>
<td>gossip</td>
<td>0.335</td>
<td>0</td>
<td>0.405</td>
</tr>
<tr>
<td>wuthering</td>
<td>0</td>
<td>0</td>
<td>0.588</td>
</tr>
</tbody>
</table>

\[
\cos(\text{SaS,PaP}) = 0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 = 0.94
\]

\[
\cos(\text{SaS,WH}) = 0.79
\]

\[
\cos(\text{PaP,WH}) = 0.69
\]
Calculando a pontuação por cosenos

\[
\text{\textsc{CosineScore}}(q)
\]

1. \( \text{float Scores[N]} = 0 \)
2. \( \text{float Length[N]} \)
3. \( \text{for each query term } t \)
4. \( \text{do calculate } w_{t,q} \text{ and fetch postings list for } t \)
5. \( \text{for each pair } (d, tf_{t,d}) \text{ in postings list} \)
6. \( \text{do } \text{Scores}[d] += w_{t,d} \times w_{t,q} \)
7. \( \text{Read the array Length} \)
8. \( \text{for each } d \)
9. \( \text{do } \text{Scores}[d] = \text{Scores}[d]/\text{Length}[d] \)
10. \( \text{return Top } K \text{ components of Scores[]} \)

### Ponderação \( \text{tf-idf} \) tem muitas variantes

<table>
<thead>
<tr>
<th>Term frequency</th>
<th>Document frequency</th>
<th>Normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n ) (natural)</td>
<td>( tf_{t,d} )</td>
<td>( n ) (no)</td>
</tr>
<tr>
<td>( l ) (logarithm)</td>
<td>( 1 + \log(tf_{t,d}) )</td>
<td>( t ) (idf)</td>
</tr>
<tr>
<td>( a ) (augmented)</td>
<td>( 0.5 + \frac{0.5 \times tf_{t,d}}{\max_i(tf_{i,d})} )</td>
<td>( p ) (prob idf)</td>
</tr>
<tr>
<td>( b ) (boolean)</td>
<td>( \begin{cases} 1 &amp; \text{if } tf_{t,d} &gt; 0 \ 0 &amp; \text{otherwise} \end{cases} )</td>
<td>( c ) (cosine)</td>
</tr>
<tr>
<td>( L ) (log ave)</td>
<td>( \frac{1 + \log(tf_{t,d})}{1 + \log(\text{ave}<em>{i \in d}(tf</em>{i,d}))} )</td>
<td>( u ) (pivoted unique)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( b ) (byte size)</td>
</tr>
</tbody>
</table>

### Ponderação pode diferir em consultas VS documentos

- Muitos sistemas de busca permitem ponderações diferentes para consultas VS documentos;
- **Notação SMART**: denota a combinação em uso por um sistema, com a notação \( \text{ddd.qqq} \), usando o acrônimo da tabela anterior;
- Um esquema muito comum de ponderação é: \( \text{inc.ltc} \):
  - **Documento**: \( \log tf \) (l como o primeiro caracter), nenhuma normalização \text{idf} (pode ser má ideia) e
  - **Query**: \( \log tf \) (l na coluna mais a esquerda), \text{idf} (t na segunda coluna), e \text{coseno};
Exemplo tf-idf: Inc.ltc

Document: *car insurance auto insurance*

Query: *best car insurance*

<table>
<thead>
<tr>
<th>Term</th>
<th>tf-raw</th>
<th>tf-wt</th>
<th>df</th>
<th>idf</th>
<th>wt</th>
<th>n'lize</th>
<th>Document</th>
<th>tf-raw</th>
<th>tf-wt</th>
<th>wt</th>
<th>n'lize</th>
<th>Prod</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>2.3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.52</td>
<td>0</td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
<td>5000</td>
<td>1.3</td>
<td>1.3</td>
<td>0.34</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>2.0</td>
<td>2.0</td>
<td>0.52</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.52</td>
<td>0.27</td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>3.0</td>
<td>3.0</td>
<td>0.78</td>
<td>2</td>
<td>1.3</td>
<td>1.3</td>
<td>0.68</td>
<td>0.53</td>
<td></td>
</tr>
</tbody>
</table>

Exercise: what is $N$, the number of docs?

$$\text{Doc length} = \sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$$

$$\text{Score} = 0 \times 0 + 0.27 \times 0.53 = 0.8$$
AULA 6 – Avaliação

Medidas para um mecanismo de pesquisa

✔ Para se selecionar um mecanismo de pesquisa é necessário ter em conta a rapidez da sua indexação
  - Número de documentos/hora
  - O tamanho médio do documento.
✔ É necessário saber o tempo de resposta de uma pesquisa
  - Latência em função do tamanho do índice.
✔ Verificar a expressividade da linguagem de consulta
  - O mecanismo demonstra capacidade de expressar a necessidade de informações complexas
  - E a sua velocidade em consultas complexas.

Medindo a felicidade de utilizador

✔ Todos os critérios anteriores são mensuráveis, mas a medida-chave será a felicidade do utilizador.
✔ Para isso é necessário saber qual é o tipo de utilizador da aplicação, assim pode-se saber
  - Se for um sistema web...
    ▪ Que o utilizador encontra o que quer e retorna ao sistema (pode-se medir a taxa de retorno do utilizador)
    ▪ Concluindo a tarefa - a pesquisa como um meio e não como um fim
  - Mas se for um site de e-commerce o utilizador encontra o que quer e compra, sendo a felicidade do utilizador ou do site que se pretende obter a medida (medida do tempo para comprar, fração de utilizadores que se tornam compradores).
  - Se forem empresas (companhias/governo/academia) a medida será sobre a “produtividade do utilizador”, ou seja, quanto tempo os meus utilizadores economizam quando procuram por informação, e outros critérios como a largura de acesso ou acesso seguro, etc.
Avaliando um sistema de IR

**Nota:** a necessidade de informação é traduzida numa query.

- A relevância é avaliada em relação à informação necessária e não à query.
- Por exemplo, informação necessária: “Estou à procura de informações sobre se beber vinho tinto é mais eficaz para reduzir o risco de ataques cardíacos do que o vinho branco”.
  - **query:** vino branco vermelho ataque cardíaco eficaz.
- Avaliar se o documento aborda a informação necessária, não se possuí estas palavras.

Medida de relevância padrão

- O TREC - National Institute of Standards and Technology (NIST) - correu um grande teste base de RI por muitos anos utilizando coleções
- Incluindo da Reuters, de documentos de referência usados.
- Foram especificadas “tarefas de recuperação”, por vezes como consultas.
- Sendo feita a atribuição, por um especialista, para cada query e cada documento, como **Relevante** ou **Não-Relevante**(ou pelo menos um subconjunto de documentos que alguns sistemas retornam para a consulta).

Precisão e Retorno (Precision and Recall)

**Precisão (Precision)**

Fração de documentos recuperados que são relevantes = $P(\text{relevante} | \text{recuperado})$

**NOTA:** A precisão é utilizada quando a probabilidade de um resultado positivo é correcta é importante.

**Retorno (Recall)**

Fração de documentos relevantes que são recuperados = $R(\text{recuperado} | \text{relevante})$
Precisão \( P = \frac{tp}{tp + fp} \)
Retorno \( R = \frac{tp}{tp + fn} \)

### Erros de classificação

- **Falso positivo** (erro do tipo 1)
  - um documento não-relevante é devolvido.
  
  \[ \text{Fallout} = \frac{fp}{tn + fp} \]

- **Falso negativo** (erro do tipo 2)
  - um documento relevante não é devolvido

  \[ 1 - \text{Recall} \]

### Deveríamos, inversamente, usar a medida de precisão para a avaliação?

- Dada uma consulta, um sistema classifica cada documento como “Relevante” ou “Não-relevante”.

- A **precisão** de um sistema: a fração destas classificações são corretas

\[ \frac{(tp + tn)}{(tp + fp + fn + tn)} = \text{exatidão} \]
Precisão é uma medida de avaliação comum usada em trabalhos de classificação de máquinas de aprendizagem.

Por que não é uma medida de avaliação muito útil em RI?

Por que não usar precisão?

• Como construir um sistema de pesquisa com precisão de 99.9999% com baixo orçamento...
• As pessoas que fazem recuperação de informação querem encontrar alguma coisa e tem uma certa tolerância a lixo.

Precisão/Retorno (precision/Recall)

• Pode-se ter alto retorno (mas baixa precisão) ao recuperar todos os documentos para todas as consultas!
• Retorno (recall) é uma função crescente de número de documentos recuperados.
• Num bom sistema, quando a precisão decresce o número de documentos recuperados aumenta (retorno aumenta).
  o Isto não é um teorema, mas um resultado com forte confirmação empírica.

Dificuldade no uso da precisão/retorno

• Precisa da decisão humana de relevância
• As pessoas não são assessores confiáveis
• Decisão tem que ser binária
• Decisões com nuances?
• Fortemente enviesado por coleção/autoria
• Resultados podem não traduzir de um domínio para outro
Uma medida combinada: F

✓ Medida combinada que avalia o intercâmbio precisão/retorno é a medida F (modo de ponderação harmônica):

\[
F = \frac{1}{\frac{1}{P} + \frac{1}{R}} = \frac{2RP}{(R+P)}
\]

✓ Pessoas normalmente usam a medida balanceada F, i.e., com \(\beta = 1\) ou \(\beta = \frac{1}{2}\)

✓ Modo harmónico é uma média conservadora.

Fórmula geral

\[
F_\beta = \frac{(\beta^2 + 1)RP}{(R + \beta^2 P)}
\]

Onde \(\beta\) é o parâmetro que determina a importância relativa da precisão e retorno.

F1 e outras médias
Eficácia do Ranking

Resumindo o Ranking

✓ Calcular **Retorno** e **Precisão** em posições fixas do Ranking
✓ Cálculo de precisão em níveis de **Retorno** padrão de 0,0 até 1,0
   - requer interpolação
✓ Calcular a média dos valores de **Precisão** das posições do Ranking onde um documento relevante foi recuperado

Precisão média

Ranking #1: \( (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6) / 6 = 0.78 \)

Ranking #2: \( (0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6) / 6 = 0.52 \)
Média entre Queries

- Resume rankings de várias queries pela média da Precisão média.
- Medida mais comum utilizada em trabalhos de pesquisa
- Assume que o utilizador está interessado em encontrar muitos documentos relevantes para cada consulta
- Exige decisões de relevância em muitas coleções de texto

Calculando a média

Mean Average Precision (MAP)

<table>
<thead>
<tr>
<th>Query 1</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query 2</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>0.33</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>0.33</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>0.67</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>0.67</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>0.38</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Gráficos de Retorno-Precisão

✓ Também são resumos úteis.
Interpolation

✓ Calcular precisão em níveis de retorno padrão, para gráficos de médias

\[ P(R) = \max\{P' : R' \geq R \land (R', P') \in S}\]

em que S é o conjunto de valores observados (R, P) pontos

✓ Define a Precisão em qualquer nível de Retorno com a Precisão máxima observada em qualquer ponto de de Retorno-Precisão num nível alto de de Retorno.
  
  o cria uma função que executa passo-a-passo
  
  o define Precisão no Retorno 0.0
Precisão média nos níveis da Retorno padrão

<table>
<thead>
<tr>
<th>Recall</th>
<th>0.0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ranking 1</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.67</td>
<td>0.67</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Ranking 2</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.43</td>
<td>0.43</td>
<td>0.43</td>
<td>0.43</td>
<td>0.43</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>Average</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.59</td>
<td>0.47</td>
<td>0.47</td>
<td>0.47</td>
<td>0.47</td>
<td>0.47</td>
<td>0.47</td>
<td>0.47</td>
</tr>
</tbody>
</table>

✓ O gráfico Retorno-Precisão apresentado, simplesmente une os pontos de precisão média nos níveis de retorno padrão.

Gráfico para 50 queries

![Gráfico de precisão de retorno para 50 queries](image1)

![Gráfico de precisão de retorno para 50 queries](image2)
Focando nos Documentos do Topo

✓ Os utilizadores tendem a olhar apenas para a parte superior da lista de resultados classificados para encontrar documentos relevantes;
✓ Algumas tarefas de pesquisa têm apenas um documento relevante;
✓ Retorno não apropriado
  o é necessário medir o quão bem o motor de pesquisa faz a recuperação de documentos relevantes nas classificações muito altas

✓ Precisão na classificação R
  o R tipicamente 5, 10, 20
  o fácil de computar, de calcular a média, de entender
  o não é sensível a classificar posições menos de R

✓ Classificação Recíproca (Reciprocal Rank - RR)
  o recíproco da classificação em que o primeiro documento relevante é recuperado

\[
\text{RR} = \frac{1}{2}, \quad \frac{1}{5}
\]

  o muito sensível à posição de classificação

Mean Reciprocal Rank (MRR)

✓ É a média das classificações de reciprocidade ao longo de um conjunto de consultas

\[
\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{Q} \frac{1}{\text{rank}_i}. 
\]

<table>
<thead>
<tr>
<th>Query</th>
<th>Results</th>
<th>Correct response</th>
<th>Rank</th>
<th>Reciprocal rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>catten, cati, cats</td>
<td>cats</td>
<td>3</td>
<td>1/3</td>
</tr>
<tr>
<td>torus</td>
<td>torii, tori, toruses</td>
<td>tori</td>
<td>2</td>
<td>1/2</td>
</tr>
<tr>
<td>virus</td>
<td>viruses, viri, viri</td>
<td>viruses</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\text{MRR} = (1/3 + 1/2 + 1)/3 = 0,61
\]
Ganho acumulado descontado (Discounted Cumulative Gain - DCG)

- Medida popular para avaliar a pesquisa na web e tarefas relacionadas.
- Podem haver dois pressupostos:
  - Documentos mais relevantes são mais úteis que documentos marginalmente relevantes.
  - Quanto mais baixa é a posição da classificação de um documento relevante, menos útil é para o utilizador, porque é menos provável ser visto.
- Utiliza relevância classificada como uma medida da utilidade, ou ganho, a partir de análise de um documento.
- O ganho é acumulado, começando no topo da classificação e pode ser reduzido, ou descontado, em níveis inferiores.

O **Desconto típico** é \(1/\log (\text{classificação})\)

- Com base 2, o desconto na classificação 4 é \(1/2\), e na classificação 8, ele é \(1/3\).

- DCG é o ganho total acumulado numa determinada classificação \(p\):
  \[
  DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}
  \]

- Formulação alternativa:
  \[
  DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log(1+i)}
  \]

- Utilizada por algumas empresas de pesquisa na web
- Enfatiza recuperação de documentos altamente relevantes
Exemplo DCG

- 10 ranked documents judged on 0-3 relevance scale:
  3, 2, 3, 0, 0, 1, 2, 2, 3, 0
- discounted gain:
  3, 2/1, 3/1.59, 0, 0, 1/2.59, 2/2.81, 2/3, 3/3.17, 0
  = 3, 2, 1.89, 0, 0, 0.39, 0.71, 0.67, 0.95, 0
- DCG:
  3, 5, 6.89, 6.89, 6.89, 7.28, 7.99, 8.66, 9.61, 9.61

DCG normalizado

✓ Os números são calculados através de um conjunto de consultas de valores de classificação específicas
  - (p.e., DCG com a classificação 5 é 6.89 e com a classificação 10 é 9.61).
✓ Os valores são muitas vezes normalizados comparando a DCG em cada classificação com o valor DCG para o ranking perfeito
  - fazendo uma média de cálculo mais fácil para consultas com diferentes números de documentos relevantes.

Exemplo NDCG

- Perfect ranking:
  3, 3, 3, 2, 2, 2, 1, 0, 0, 0
- ideal DCG values:
  3, 6, 7.89, 8.89, 9.75, 10.52, 10.88, 10.88, 10.88, 10
- NDCG values (divide actual by ideal):
  1, 0.83, 0.87, 0.76, 0.71, 0.69, 0.73, 0.8, 0.88, 0.88
  - NDCG ≤ 1 at any rank position
### Métricas de eficiência

<table>
<thead>
<tr>
<th>Nome da Métrica</th>
<th>Descrição</th>
</tr>
</thead>
<tbody>
<tr>
<td>tempo de indexação decorrido</td>
<td>Mede a quantidade de tempo necessário para a construção de um índice de documentos num sistema</td>
</tr>
<tr>
<td>tempo de processamento na indexação</td>
<td>Mede os segundos de CPU utilizados na construção de um índice de documento. Isto é semelhante ao tempo de indexação decorrido, mas não conta o tempo de espera de I/O ou ganhos de velocidade de paralelismo.</td>
</tr>
<tr>
<td>processamento da consulta</td>
<td>Número de queries processadas por segundo</td>
</tr>
<tr>
<td>latência de consulta</td>
<td>A quantidade de tempo que um utilizador deve esperar após a emissão de uma query antes de receber uma resposta, medido em milissegundos. Isto pode ser medido utilizando a média, mas é muitas vezes mais instrutivo quando usado com a mediana ou um percentual ligado.</td>
</tr>
<tr>
<td>A indexação de espaço temporário</td>
<td>Quantidade temporário de espaço em disco usado durante a criação de um índice.</td>
</tr>
<tr>
<td>Tamanho do índice</td>
<td>Quantidade de armazenamento necessário para armazenar os ficheiros do índice.</td>
</tr>
</tbody>
</table>

### Apresentação de Resultados

- Tendo classificado os documentos que correspondem com a query, queremos apresentá-los numa lista de resultados.
- Geralmente, uma lista dos títulos de documentos com um breve resumo, conhecido como “os 10 links azuis”.

---

**John McCain**

John McCain 2008 - The Official Website of John McCain's 2008 Campaign for President ... African American Coalition; Americans of Faith; American Indians for McCain; Americans with ...  
[www.johnmccain.com](http://www.johnmccain.com) - [Cached page](http://www.johnmccain.com)

**John McCain - McCain-Palin 2008**

John McCain 2008 - The Official Website of John McCain's 2008 Campaign for President ... African American Coalition; Americans of Faith; American Indians for McCain; Americans with ...  
[www.johnmccain.com](http://www.johnmccain.com) - [Cached page](http://www.johnmccain.com)

**John McCain News - msnbc.com**

Complete political coverage of John McCain ... Republican leaders said Saturday that they were worried that Sen. John McCain was heading for defeat unless he brought stability to ...  
[www.msnbc.com](http://www.msnbc.com) - [Cached page](http://www.msnbc.com)

**John McCain | Facebook**

Welcome to the official Facebook Page of John McCain. Get exclusive content and interact with John McCain right from Facebook. Join Facebook to create your own Page or to start ...  
[www.facebook.com/johnmccain](http://www.facebook.com/johnmccain) - [Cached page](http://www.facebook.com/johnmccain)
Resumos

✔ O título é frequentemente extraído automaticamente do metadado do documento. E sobre os resumos?
  o A descrição é crucial.
  o O utilizador pode identificar dicas como boas/relevantes baseando-se na descrição.

✔ Dois tipos básicos:
  o Estático
  o Dinâmico

✔ Um resumo estático de um documento que é sempre o mesmo, independentemente da consulta que fez devolver o documento.

✔ Um resumo dinâmico é uma tentativa dependente de consulta para explicar porque o documento foi recuperado para a consulta em questão.

Resumos estáticos

✔ Em sistemas típicos, o resumo estático é um subconjunto do documento.

✔ Heurística simples: as primeiras 50 (ou outro valor – pode ser variado) palavras do documento
  o Resumo é guardado(cache) na hora da indexação.

✔ Mais sofisticado: extrair de cada documento um conjunto de sentenças “chave”
  o Heurística NLP simples para classificar cada frase.
  o Resumo é composto de frases com maior classificação.

✔ O mais sofisticado: NLP usado para sintetizar um resumo
  o Raramente usado em RI; cf. sumarização de texto funciona

Resumos dinâmicos

✔ Apresenta uma ou mais “janelas” no documento que contém muitos dos termos da consulta.
  o Fragmentos “KWIC” : Palavra chave na apresentação do Contexto
Técnicas para resumos dinâmicos

- Encontrar pequenas janelas no documento que contêm os termos da consulta
  - Requer procura rápida de janela na cache do documento
- Pontua cada janela da consulta (Score each window wrt)
  - Usa várias características como tamanho de janela, posição no documento, etc.
  - Combina recursos através de uma função de pontuação
- Desafios na avaliação: julgar resumos
  - Mais fácil fazer comparações de pareamento (pairwise) do que avaliações de relevância binárias

Links rápidos (Quicklinks)

- Para uma consulta navegacional como united Airlines, a necessidade dos utilizadores provavelmente serão satisfeitas com www.united.com.
- Quicklinks fornecem dicas navegacionais nas suas páginas principais.
Agenda

- Queries Wild-card
- Spelling correction
- Fonética (Soundex)

Queries Wild-card*

- mon*: pesquisa todos os documentos que contenham qualquer palavra começada por “mon”
- É fácil com a léxica nas árvores binária (ou B-tree): recupera todas as palavras entre o intervalo: mon ≤ w < moo
- *mon*: pesquisa todos os documentos que contenham qualquer palavra acabada por “mon”
- É mais complicado
  - Manter uma B-tree adicional para os termos “finais”
  - recupera todas as palavras entre o intervalo: nom ≤ w < non.

Processamento da Query

- Neste momento, temos uma enumeração de todos os termos no dicionário que corresponde à query wild-card.
- Ainda teremos que verificar os postings para cada termo enumerado.
- P.e.: considerar a query:
  
  se*ate AND fil*er
  
  Isto pode resultar na execução de muitas Boolean AND queries.

B-trees que contenha *’s no final de um termo da query

- Como podemos tratar *’s que estejam no meio do termo da query?
  - Exemplo: co*tion
✓ Podemos pesquisar por `co* AND *tion` numa B-tree e fazer a intersecção dos dois conjuntos devolvidos.
   
   o Demasidado pesado (dispendioso)

✓ **Solução:** transformar queries wild-card para que *'s ocorram no final

✓ Esta solução traz-nos o conceito de **Índice Permuterm.**

**Índice Permuterm**

✓ Para o termo `hello`, indexar com:
   
   o `hello$, ello$h, llo$he, lo$hel, o$hel` onde $ é um símbolo especial.

✓ Queries:
   
   - X pesquisa em X$
   - *X pesquisa em X*$
   - X*Y pesquisa em Y$X*$
   - X pesquisa em $X*$
   - *X* pesquisa em X*
   - X*Y*Z (exercício) → Z$X*Y

![Query = hel{o}](image1)

X=hel, Y=o
Lookup o$hel*

**Processamento da query Permuterm**

✓ Rodar a query wild-card para a direita

✓ Depois, utilizar a pesquisa na B-tree como visto anteriormente.

✓ Problema Permuterm:
   
   o Aumenta o tamanho do Índice
   o Aumenta cerca de 10 vezes o tamanho (para o idioma Inglês)
Índices Bigram (k-gram)

- Enumerar todos os k-grams (sequência de todos os k caracteres) que ocorrem num termo.
- P.e.: do texto “April is the cruelest month” obtemos 2-grams (bigrams)

\[
$\text{a, ap, pr, ri, il, l$, s$, i, s$, t, th, he, e$, c, cr, ru, }
\text{ue, el, le, es, st, t$, m, mo, on, nt, h$}
\]

- O símbolo $ é um caractere especial que serve como fronteira.
- Mantemos um segundo índice invertido a partir de bigramas para termos do dicionário que correspondem a cada bigrama.

Exemplo de um Índice Bigram

- O índice k-gram pesquisa termos com base na query que consiste nos k-grams (neste caso, k=2).

![Diagrama de bigramas](image)

Processando wild-cards

- A query mon* pode agora ser tratada como
  - Sm AND mo AND on
- Obtém termos que correspondam à versão AND da nossa query wild-card.
- Mas poderemos obter o termo moon.
- É necessário filtrar posteriormente estes termos com a query.
- Os termos resultantes são pesquisados no índice invertido de documentos por termos.
- Torna-se mais rápido e maior eficácia no tamanho (comparando com o método Permuterm).
- Como antes, temos que executar uma query booleana para cada enumeração dos termos filtrados.
- Wild-cards podem resultar em execuções de queries dispendiosas (disjunções muito grandes,...)
Se incentivarmos à “preguiça”, os utilizadores irão reagir!

**Spelling correction**

**Correção Spell**

- Duas principais utilizações
  - Corrigir documentos enquanto estão a ser indexados
  - Corrigir queries do utilizador para recuperar informações “corretas”

- Duas principais vertentes:
  - Palavra isolada
    - Verificar cada palavra para pesquisar erros de ortografia
    - Não consegue detetar erros de digitação resultando em palavras corretas
      - P.e.: from → form
  - Contexto sensível
    - Pesquisar as palavras circundantes (próximas)
    - P.e.: I flew form Heathrow to Narita.

**Correção do Documento**

- Especialmente necessário para documentos OCR
  - Os algoritmos de correção estão sintonizados para: rn/m
  - Podem utilizar o conhecimento do domínio específico.
    - P.e.: OCR pode confundir O e D mais vezes d que confundirá O e I (adjacente nos teclados QWERTY, logo muito mais vezes digitado erradamente).
• Mas também, páginas web e mesmo material impresso têm erros de digitação
• **Objetivo:** o dicionário conter menos erros ortográficos
• Muitas vezes não alteramos os documentos, mas sim corrigimos o mapeamento de consulta de documentos.

### Query mis-spellings

✓ Vejamos o exemplo: a query **Alanis Morisett**
✓ Podemos recuperar documentos que contenham os termos corretos ou...
✓ ... devolver algumas sugestões alternativas para tentar completar (corrigir) os termos da query.
  o Geralmente, é a ferramenta “Did you mean ... ?”

### Correção de palavras isoladas

✓ Premissa fundamental – existe uma forma léxica de onde provêm as palavras corretas
✓ Basicamente, temos duas hipóteses para isso:
  o Um padrão léxico, tal como:
    ▪ Webster’s English Dictionary
    ▪ An “industry-specific” lexicon – hand-maintained
  o O léxico do corpus indexado
    ▪ P.e., todas as palavras da web
    ▪ Todos os nomes, acrónimos, etc.
✓ Suponhamos que dado um léxico e uma sequência de caracteres Q, queremos retornar as palavras no léxico mais próximo de Q.
✓ Como se define "mais próximo"?
✓ As alternativas que iremos ver são:
  o Editar distância (distância Levenshtein)
  o Editar distância ponderada
  o Sobreposição n-gram
Editar a distância

✓ Dadas duas strings $S_1$ e $S_2$, qual o número mínimo de operações para converter uma para a outra?
✓ Operações são tipicamente a nível de caracteres:
  o Inserir, eliminar, substituir (Transposição)
✓ Por exemplo, a distância de edição de DOF para DOG é 1
  o De cat para act é de 2 (Apenas 1 com transposição.)
  o De cat para dog é 3.

Editar distância ponderada

✓ Como visto acima, mas o peso de uma operação depende do(s) caracter e(s) envolvido(s)
  o Significa que, para capturar erros de OCR ou de teclado, por exemplo, o m tem mais probabilidade de ser mal-escrito como n que como q
  o Portanto, substituindo m por n é uma distância menor do que por q
  o Isto pode ser formulado como um modelo de probabilidade
✓ Requer uma matriz de peso como entrada
✓ Modificar a programação dinâmica para lidar com pesos

Utilizando distâncias de edição

✓ Dada uma query, primeiro deveremos enumerar todas as sequências de caracteres de acordo com distância de edição predefinida (ponderada) (p.e. 2)
✓ Intersetar este conjunto com a lista de palavras “corretas”
✓ Mostrar os termos ao utilizador que foram encontrados como sugestões
✓ Por outro lado,
  o Podemos verificar para todas as possíveis correções no nosso índice invertido e devolver todos os documentos encontrados... mais lento!
  o Podemos executar uma ferramenta “correção mais parecida”
✓ As alternativas tiram poder ao utilizador, no entanto permite continuar a interação com o utilizador.
Distância editada para todos os termos do dicionário?

- Dada uma query (mal escrita) - podemos calcular a sua distância de edição para cada termo do dicionário?
  - Fica dispendioso e lento
  - Qual a alternativa?
- Como é que vamos eliminar o conjunto de candidatos dos termos do dicionário?
- Uma possibilidade é a utilização de n-grams para esta sobreposição
- Isto também pode ser usado por si mesmo pela spelling correction.

Sobreposição n-gram

- Enumerar todos os n-grams na cadeia da query, bem como no léxico
- Utilizar o índice n-grams (lembremos a pesquisa wild-card) para recuperar todos os termos do léxico correspondentes a qualquer da consulta n-grams
- Colocar um limite mínimo pelo número de correspondência de n-grams

Exemplos com trigrams

- Suponhamos que o texto é **november**
  - Trigrams são **nov, ove, vem, emb, mbe, ber**.
- A query é **december**
  - Trigrams são **dec, ece, cem, emb, mbe, ber**.
- Então, 3 trigrams sobrepõem-se (de 6 para cada termo)
- Como conseguimos colocar esta medida de forma normalizada?

Uma opção – coeficiente Jaccard

- É comum utilizar esta opção para medir sobreposições
- Suponhamos que X e Y são dois conjuntos, então C.J. será:

\[
|X \cap Y|/|X \cup Y|
\]
✓ Caso seja igual a 1, então X e Y contêm os mesmos elementos
✓ Caso seja 0, os conjuntos são disjuntos
✓ Não é necessário que X e Y tenham o mesmo tamanho
✓ O resultado estará sempre entre 0 e 1
  o Com o resultado, basta decidir se há correspondência ou não
  o P.e., se C.J > 0.8, declarar que se trata de uma correspondência.

Junção de trigrams

✓ Consideremos a query lord – pretendemos identificar palavras que correspondam a 2 dos seus 3 bigramas.

![Diagrama de Junção de Trigrams]

Correção de palavras sensível ao contexto (Context-sensitive spell correction)

✓ Consideremos o texto: I flew from Heathrow to Narita.
✓ Consideremos a frase na query: “flew form Heathrow”
✓ O objetivo é respondermos:

  Did you mean “flew from Heathrow”?

Uma vez que não existem correspondência de documentos relativos à query.

Correção sensível ao contexto

✓ É necessário verificar o contexto para dar a solução.
✓ Primeira ideia: devolver o dicionário de termos mais próximo (no distância de edição ponderada) para cada termo da query.
✓ Depois, testar todos os possíveis resultados com um palavra “fixa” de cada vez
  o flew from heathrow
  o fled form heathrow
  o flea form heathrow
✓ Dica de correção: sugerir a alternativa que tenha mais visitas.

Questões gerais na correção ortográfica

✓ Listamos diversas alternativas da ferramenta “Did you mean?”
✓ É necessário descobrir o que se deve apresentar ao utilizador
✓ Utilizando heurística
  o Devolvemos a alternativa que tiver mais visitas
  o A alternativa que tenha mais visitas nos documentos
  o Análise de logs da Query + otimização
    ▪ Em especial para queries de tópicos
✓ Spell-correction é dispendioso a nível da computação
  o Evitar correr cada vez que se faz uma query?
  o Devemos apenas percorrer em queries que tenham poucos resultados
Fonética (Soundex)

Fonética – Algoritmo típico

✓ Transformar cada token para que seja indexado de uma forma que esteja reduzido em 4 caracteres.
✓ Executar o mesmo para os termos da query
✓ Construir e pesquisar num índice nas formas reduzidas
  o Apenas quando a query “indica” que se trata de uma correspondência por fonética

Algoritmo

1. Reten a primeira letra da palavra.
2. Alterar todas as ocorrências das seguintes letras para ‘0’ (zero):
   
   'A', E', 'I', 'O', 'U', 'H', 'W', 'Y'.

3. Alterar letras para números conforme é mostrado abaixo:

   B, F, P, V → 1
   C, G, J, K, Q, S, X, Z → 2
   D, T → 3
   L → 4
   M, N → 5
   R → 6

4. Remover todos os pares de dígitos seguidos.
5. Remover todos os zeros da cadeia que resultou.
6. Preencher o resultado com zeros à direita e devolver as quatro primeiras posições, que ficará na forma <letra maiúscula> <digito> <digito> <digito>.

✓ P.e., Herman transforma-se em H655.

Will hermann generate the same code?
Fonética (Soundex)

✓ Fonética é o algoritmo clássico, fornecido pela maioria das bases de dados (Oracle, Microsoft, ...)
✓ Zobel e Dart (1996) mostraram que outros algoritmos para esta correspondência de fonética têm uma eficiência maior no contexto de RI
  - Metaphone
  - Phonix
  - Editex
  - ...

Que queries podemos processar?

✓ Poderemos ter:
  - Índice posicional com apontadores de saltos
  - Índice Wild-card
  - Spell-correction
  - Fonética (Soundex)
✓ Queries tal como:

\[(\text{SPELL}(\text{moriset}) \text{ OR } \text{SOUNDEX}(\text{chaikofski}))\]
AULA 8 – Classificação e criação de Resultados

Agenda

✔ Aumentar a velocidade da classificação espaço-vetor
✔ Juntando um sistema de pesquisa completo
  ○ É necessário aprender uma mistura de tópicos e heurísticas

Computando a pontuação de cossenos

\[
\text{COSINE} \text{SCORE}(q)
\]

1. \( float\ Scores[N] = 0 \)
2. \( float\ Length[N] \)
3. \textbf{for each} query term \( t \)
4. \textbf{do} calculate \( w_{t,q} \) and fetch postings list for \( t \)
5. \textbf{for each} pair \((d, tf_{t,d})\) in postings list
6. \textbf{do} \( Scores[d] += w_{t,d} \times \ w_{t,q} \)
7. \text{Read the array} Length
8. \textbf{for each} \( d \)
10. \textbf{return} Top \( K \) components of \( Scores[] \)

Classificação eficiente de cossenos

✔ Encontrar os \( K \) documentos na coleção “aproximada” à consulta ⇒ para \( K \) cossenos consulta-documento maiores.

✔ Classificação eficiente:
  ○ Computando um único cosseno eficientemente
  ○ Escolhendo eficientemente os \( K \) maiores valores de cosseno
    ▪ Podemos fazer isto sem computar todos os \( N \) cossenos?
O que estamos a fazer: resolver o problema dos vizinhos mais próximos a k, para um vetor consulta.

Em geral, não sabemos como fazer isto eficientemente para espaços super-dimensionados

Mas isto é resolvido por consultas pequenas, e índices padrões suportam isto bem.

Caso especial – consultas não ponderadas

Sem ponderação nos termos de consulta
  - Assumimos que cada termo da consulta ocorre apenas uma vez
  - Assim, para a classificação, não precisamos normalizar os vetores da consulta.
    - Ligeira simplificação do algoritmo do capítulo 6

Cosseno rápido: consulta não ponderada

```c
FASTCOSINESCORE(q)
1  float Scores[N] = 0
2  for each d
3    do Initialize Length[d] to the length of doc d
4     for each query term t
5       do calculate w_{t,q} and fetch postings list for t
6           for each pair (d, tf_{t,d}) in postings list
7             do add wf_{t,d} to Scores[d]
8     Read the array Length[d]
9    for each d
10   do Divide Scores[d] by Length[d]
11  return Top K components of Scores[]
```
Computando os $K$ cossenos maiores: seleção vs. ordenação

✓ Tipicamente nós queremos recuperar os top $K$ documentos (na classificação de cossenos para a consulta)
  ○ Não ordenar totalmente todos os documentos da coleção
✓ Podemos escolher os documentos com os $K$ maiores cossenos?
✓ Seja $J =$ número de documentos com cossenos diferentes de zero
  ○ Vamos devolver os $K$ melhores destes $J$

Uso de pilha (heap) para selecionar os top $K$

✓ Árvore binária em que cada valor de nó > os valores dos filhos
✓ Seja $2J$ operações para construir, então cada $K$ “vencedor” é lido em $2\log J$ passos.
✓ Para $J=1M$, $K=100$, isto é, sensivelmente, 10% do custo para ordenação.

![Diagrama de heap](attachment:image.png)

Agulhas (Bottleneck)

✓ Agulha computacional primária na pontuação: computação de cossenos
✓ Podemos evitar toda esta computação?
✓ Sim, mas às vezes podemos errar
  ○ Um documento que não está nos top $K$ podem influenciar a lista dos $K$ documentos de saída
  ○ Isto é algo mau?
Similaridade de cosseno é apenas um proxy (representante, procurador)

✔ Utilizadores possuem a tarefa da formulação da consulta
✔ Cosseno faz corresponder documentos à query
✔ Deste modo, o cosseno é apenas um proxy para a felicidade do utilizador
✔ Se nós recebemos a lista dos \( K \) documentos “próximos” aos top \( K \) por medida de cosseno, deve estar bom.

Abordagem genérica

✔ Considere um conjunto \( A \) de candidatos, com \( K < |A| < N \)
  - \( A \) não contém necessariamente os top \( K \), mas possui muitos dos documentos dentre os top \( K \)
  - Retorna os top \( K \) documentos em \( A \)
✔ Imagine \( A \) como uma poda de não-candidatos (pruning non-contenders)
✔ A mesma abordagem é também utilizada para outras (não-cosseno) funções de pontuação
✔ Vamos ver diversos esquemas a seguir desta abordagem

Eliminação de índices

✔ Apenas considera termos de consulta com alto idf
  - Os termos com baixo idf são tratados como stopwords e não contribuem para a classificação
✔ Apenas considera documentos que contém muitos termos na query

Termos de consulta com alto idf apenas

✔ Para uma consulta do tipo \textit{catcher in the rye}
✔ Apenas acumula a pontuação de \textit{catcher e rye}
✔ \textbf{Intuição}: \textit{in e the} contribuem muito pouco para a pontuação e então não altera muito a ordenação do rank
✔ \textbf{Benefício}:
  - Postings de termos com baixo idf possuem muitos documentos \( \rightarrow \) estes (muitos) documentos são eliminados do conjunto \( A \) de candidatos
Documentos contendo muitos termos de consulta

☑ Qualquer documento com pelo menos um termo de consulta é um candidato para a lista de saída dos top \( K \)

☑ Para consultas multi-termos, apenas é computada a pontuação para documentos contendo muitos dos termos da consulta
  - Digamos, ao menos 3 dos 4
  - Estabelece uma “conjunção suave” nas consultas vistas em sistemas de busca web (Google)

☑ Fácil de implementar percorrendo os postings

3 de 4 termos da consulta

<table>
<thead>
<tr>
<th>Termo</th>
<th>Postings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>3 4 8 16 32 64 128</td>
</tr>
<tr>
<td>Brutus</td>
<td>2 4 8 16 32 64 128</td>
</tr>
<tr>
<td>Caesar</td>
<td>1 2 3 5 8 13 21 34</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>13 16 32</td>
</tr>
</tbody>
</table>

☑ Pontuação apenas é computada para os docs 8, 16 e 32.

Listas dos campeões

☑ Pré-computar para cada termo \( t \) do dicionário, os \( r \) documentos de maior peso dentre os \( t \)’s postings
  - Chamado de lista de campeões (champion list) para \( t \)
  - (aka fancy list ou top docs para \( t \))

☑ Note que \( r \) deve ser escolhido no tempo de construção do índice
  - Portanto, é possível que \( r < K \)

☑ No tempo da consulta, apenas computa pontuação para documentos na lista de campeões de algum termo da query
  - Devolve os \( K \) documentos de maior pontuação entre estes
Pontuação de qualidade estática

- Queremos os documentos do topo do ranking que sejam relevantes e autoritários
- Relevância está sendo modelada pela pontuação de cossenos.
- Autoridade é uma propriedade tipicamente independente de consulta de um documento.

- Exemplos de sinais de autoridade
  - Wikipedia entre os sites
  - Artigos em certos jornais
  - Um paper com muitas citações
  - Many diggs, Y!buzzes ou del.icio.us marks
  - (Pagerank)

Modelagem de autoridade

- Atribui a cada documento uma pontuação de qualidade independente de consulta em [0,1] para cada documento d
  - Denote isto por \( g(d) \)
- Portanto, uma quantidade como o número de citações é dimensionado em [0,1]

Pontuação líquida

- Considere uma pontuação total simples combinando relevância de cossenos e autoridade

  \[
  \text{net-score}(q,d) = g(d) + \text{cosseno}(q,d)
  \]

  - Podemos usar alguma outra combinação do que ponderação de igualdade
  - De facto, qualquer função dos dois “sinais” da felicidade do utilizador (visto adiante)
- Agora nós pesquisamos os top \( K \) documentos por pontuação líquida
Top K por pontuação líquida – métodos rápidos

- **Primeira ideia**: Ordene todos os postings por $g(d)$
- **Chave**: este é uma ordenação comum para todos os postings
- Portanto, simultaneamente podemos percorrer os postings dos termos da consulta para
  - Interseção de Postings
  - Computação de pontuação de cossenos

Por que ordenar os postings por $g(d)$?

- Sob ordenação-$g(d)$, documentos com pontuação top são prováveis de aparecer antes percorrendo os postings
- Em aplicações de limite de tempo (digamos, são retornados quaisquer resultados de pesquisa possível em 50 ms), permitindo parar de percorrer os postings mais cedo
- Redução de pontuação de computação para todos documentos nos postings

Lista de Campeões em ordenação $g(d)$

- Podemos combinar a lista de campeões com ordenação $g(d)$
- Mantendo para cada termo uma lista de campeões dos r documentos com maior $g(d) + \text{tf-idf}_d$
- Pesquisar os top-K resultados de apenas os documentos que estão na lista de campeões

Listas de altas e baixas

- Para cada termo, mantemos duas listas de postings chamados altas e baixas
  - Pense como alta como a lista de campeões
- Quando percorreremos os postings de uma consulta, apenas atravessar a lista de altos primeiros
  - Se nós tivermos mais do que $K$ documentos, selecionamos os top $K$ e paramos
  - Pelo contrário, procedemos pegando nos documentos da lista de baixas
- Podemos utilizar mesmo para pontuações simples de cossenos, sem a qualidade global $g(d)$
- Um modo de segmentar o índice em dois níveis
Postings Impacto-ordenado

✓ Queremos apenas computar a pontuação para documentos para o qual o $wf_{t,d}$ é alto o suficiente
  o Ordenamos a lista de postings $wf_{t,d}$
  o Agora: nem todos os postings estão numa ordem comum!
✓ Como computamos a pontuação a fim de recuperar os top $K$?
  o Vejamos as duas ideias seguintes.

1. Terminação mais cedo

✓ Quando percorremos as listas dos postings dos termos, paramos mais cedo quando
  a. Um número fixo $r$ de documentos é atingido
  b. $wf_{t,d}$ cai para menos de um limite fixado
  c. O tempo foi atingido
✓ Seja a união dos conjuntos de resultados dos documentos
  a. Um para os postings de cada termo da consulta
✓ Computa apenas a pontuação dos documentos nesta união

2. Termos idf-ordenados

✓ Quando consideramos os postings dos termos da consulta
✓ Imagine eles em ordem decrescente de idf
  o Termos com idf altos provavelmente contribuem mais para a pontuação
✓ À medida que atualizamos a pontuação de contribuição de cada termo da consulta
  o Paramos se a pontuação do documento não muda relativamente
✓ Podemos aplicar para cosseno ou alguma outra pontuação líquida

Poda de grupo: pré-processamento

✓ Peguemos nos \N documentos aleatórios: chame-os de líderes
  o Aleatório = rápido + reflete bem a distribuição
✓ Para cada outro documento, pré-calcular proximidade com o líder
  o Documentos ligados a um líder: são os seguidores;
  o Provavelmente: cada líder tem $\sim \sqrt{N}$ seguidores.
Poda de grupo: processamento de consulta

- Processa uma query da seguinte forma:
  - Dada uma query $Q$, encontra o líder $L$ mais próximo.
  - Encontre os $k$ documentos mais próximos entre os seguidores de $L$.

Visualização

![Diagrama de cluster com líderes e seguidores]

Variantes gerais

- Tenha cada seguidor ligados a $b_1=3$ (digamos) líderes mais próximos.
- Da query, encontrar $b_2=4$ (digamos) líderes mais próximos e seus seguidores.
- Pode reincindir na construção Líder/Seguidor.
  - Tratar cada cluster como um espaço, encontrar subespaços, repetir...
Índices paramétricos e de zona

- Até ao momento, um documento foi uma sequência de termos
- De facto, documentos tem múltiplas partes, algumas com semântica especial:
  - Autor
  - Título
  - Data de publicação
  - Linguagem
  - Formato
  - etc.
- Isto constitui os metadados de um documento.

Campos

- Por vezes queremos procurar por estes metadados
  - Exemplo, encontre documentos cujo autor é William Shakespeare do ano 1601, contendo *alas poor Yorick*
- Ano = 1601 é um exemplo de um campo
- Também, sobrenome do autor = shakespeare, etc

Índices paramétricos ou de campo: postings para cada valor de campo

- Por vezes, construindo árvores de faixa (exemplo, para datas)
- Campos de consulta tipicamente são tratados como conjunções
  - (documento deve ter como autor shakespeare)

Zona

- Uma zona é uma região do documento que contém uma quantidade arbitrária de texto, por exemplo,
  - Título
  - Abstract
  - Referências ...
- A construção de índices invertidos de zonas também permitem as consultas
  - Exemplo, “encontre documentos com merchant na zona de título e correspondam à consulta gentle rain”
Exemplo de índices de zonas

Índices diferenciados

✓ Quebre postings em listas hierárquicas
  ○ Muito importante
  ○ ...
  ○ Menos importante

✓ Pode ser feito por \( g(d) \) ou outra medida

✓ Índices invertidos divididos em camadas de importâncias decremental

✓ Na hora da query, utilize as maiores camadas a não ser que não complete os \( K \) documentos
  ○ Se acontecer passe para a camada abaixo

Exemplo de índice de camadas
Proximidade da Consulta e termo

- **Consulta de texto livre**: apenas um conjunto de termos escrito numa caixa de pesquisa – comum na web.
- Utilizadores preferem documentos em que os termos da consulta ocorrem com proximidade entre si.

- Seja w a menor janela num documento que contém todos os termos da consulta, por exemplo,
  - Para a consulta *strained mercy* a menor janela no documento *The quality of mercy is not strained* é 4 (palavra)

- Queremos uma função de pontuar que leve isto em conta – como?

**Analisadores de consultas (query parsers)**

- Consulta de texto livre de um utilizador pode, de facto, resultar numa ou mais queries para indexar, por exemplo, a consulta *rising interest rates*.
  - Executar a query como uma query de frase
  - Se <K documentos que contêm a a frase *rising interest rates*, executar as duas consultas de frases *rising interest e interest rates*
  - Se ainda <K documentos, executar a query de espaço vetor *rising interest rates*
  - Classificar os documentos correspondentes à pontuação espaço vetor

- Esta sequência é um problema para um analisador de consulta

**Pontuação Global**

- Vimos que funções de pontuação podem combinar cosseno, qualidade estática, proximidade, etc.
- Como sabemos qual a melhor combinação?
- Algumas aplicações – expert-tuned (sintonia por especialista).
- Cada vez mais comum: aprendizagem-máquina.
Colocando tudo junto
Lucene - O que é o Lucene?

- Novos utilizadores, normalmente, pensam que é um programa pronto a funcionar, como um programa de pesquisa por ficheiros ou motor de pesquisa online – um erro comum.
- O Lucene é uma biblioteca de software.

Características

- Alta performance, escalável, biblioteca de pesquisa por texto;
- Concentração na indexação e pesquisa dos documentos;
- 100% Java, sem dependências;
- Sem crawlers ou parsing de documentos.
Analysers

```java
private static final Analyzer[] analyzers = new Analyzer[]{
    new WhitespaceAnalyzer(),
    new SimpleAnalyzer(),
    new StopAnalyzer(Version.LUCENE_30),
    new StandardAnalyzer(Version.LUCENE_30),
    new SnowballAnalyzer(Version.LUCENE_30, "English")
};

private static void analyze(String text) throws IOException {
    System.out.println("Analyzing \"" + text + ",\\n");
    for (int i = 0; i < analyzers.length; i++) {
        Analyzer analyzer = analyzers[i];
        System.out.println("\t" + analyzer.getClass().getName() + ",\\n");
        TokenStream stream = analyzer.tokenStream("contents", new StringReader(text));
        while (stream.incrementToken()) {
            if (streamgetAttribute(TermAttribute.class)) {
                TermAttribute attr =
                    stream.getAttribute(TermAttribute.class);
                System.out.println(\"[ + attr + "]\\n");
            }
        }
        System.out.println("\n");
    }
}
```

- ArabicAnalyzer, BrazilianAnalyzer, ChineseAnalyzer, CJKAnalyzer, CollationKeyAnalyzer, CzechAnalyzer, DutchAnalyzer, FrenchAnalyzer, GermanAnalyzer, GreekAnalyzer, ICUCollationKeyAnalyzer, KeywordAnalyzer, PatternAnalyzer, PerFieldAnalyzerWrapper, PersianAnalyzer, QueryAutoStopWordAnalyzer, RussianAnalyzer, ShingleAnalyzerWrapper, SimpleAnalyzer, SmartChineseAnalyzer, SnowballAnalyzer, StandardAnalyzer, StopAnalyzer, ThaiAnalyzer, WhitespaceAnalyzer

TokenFilters

- LowerCaseFilter
- StopFilter
- PorterStemFilter
- SnowballFilter
- com analisadores para outras linguagens e SynonymFilter e WordDelimiterFilter (vindos do Solr)
Classes de documentos

✓ Directorias (org.apache.lucene.store.Directory)
  o DbDirectory, FileSwitchDirectory, FSDirectory, JEDirectory, RAMDirectory;

✓ Documentos (org.apache.lucene.document.Document)
  o Documentos são a unidade de indexação e pesquisa – um documento é um conjunto de campos;

✓ Field (org.apache.lucene.document.Field)
  o Cada campo tem um nome (Título, autor, data, etc) e um valor textual.
  o Field.Index, Field.Store, ..

Aplicação básica

Segmentos de indexação

✓ Cada índice consiste num ou mais segmentos;
  o Cada um é apenas um índice por si mesmo, contendo um subconjunto de todos os documentos indexados.

✓ Um novo segmento é criado quando o escritor descarregar os documentos adicionados no buffer/eliminações pendentes para a directoria;
  o Aquando da pesquisa, cada segmento é visitado separadamente e os resultados são combinados;
Cada segmento, por sua vez, consiste em múltiplos ficheiros, cada um na forma _X.<ext>
Um ficheiro especial chamado segments_<N> contem referencias para todos os segmentos em utilização.

Estrutura de indexação

Pesquisando um índice

Apenas algumas classes são precisas para realizar operações básicas de pesquisa:
Classes de *query*

- O Lucene vem com diversas subclasses de *query*:
  - TermQuery (find fields with specific values)
  - MultiTermQuery
  - BooleanQuery
  - WildcardQuery
  - PhraseQuery
  - PrefixQuery
  - MultiPhraseQuery
  - FuzzyQuery
  - TermRangeQuery
  - NumericRangeQuery
  - SpanQuery

Pontuação

- **VSM** – Vector Space Model (modelo de espaço vectorial);
- **tf** – term frequency (frequência do termo): numero de termos que coincidem no campo;
- **lengthNorm** – numero de tokens num campo;
- **idf** – inverse document frequency (frequência de documentos inversa);
- **coord** – factor de coordenação, numero de termos que coincidem;
- **document boost**;
- **query clause boost** (prioridade nas clausulas de *query*).

Construção de *queries*

- QueryParser
  - Exemplo: queryParser.parse("name:Spider-Man");
  - Boas queries introduzidas por humanos, debugging, IPC;
  - Faz análise textual e constrói queries apropriadas;
  - Nem todos os tipos de *queries* são suportados.

- Construção programática de *queries*
  - Example: new TermQuery(new Term("name","Spider-Man"))

- Explicito, não é necessário fazer o escape dos caracteres (os cuidados normais das aspas e apóstrofos);
- Não faz análise textual por nós.
Exemplos de queries

**Query Examples**

1. *justice league*
   - EQUIV: justice OR league
   - QueryParser default is “optional”
2. *justice* ~league ~name:aquaman
   - EQUIV: justice AND league NOT name:aquaman
3. "justice league" ~name:aquaman
4. title:spiderman"10 description:spiderman
5. description:“spiderman movie”~10

**Query Examples2**

1. **rangeDate:** [2000 TO 2007]
   - Range search: lexicographic ordering, so beware of numbers
2. Wildcard searches: sup?r, su?r, super*
3. spider~
   - Fuzzy search: Levenshtein distance
   - Optional minimum similarity: spid?r~0.7
4. *:*~
5. (Superman AND “Lex Luthor”) OR (~Batman +joker)

Eliminação de documentos

- IndexReader.deleteDocument(int id);
- Eliminação com o IndexWriter
  - void deleteDocuments(Query... queries)
    - Elimina os documentos que coincidam com qualquer uma das queries dadas
      (pode ser dada apenas uma query)
  - void deleteDocuments(Term... terms)
    - Elimina os documentos que coincidam com qualquer um dos termos dados
      (pode ser dado apenas um termo)
  - updateDocument(Term t, Document d)
- Eliminação não recupera imediatamente o espaço

Luke

- Ver por número de documento ou por termo;
- Ver documentos/ copiar para área de transferências;
- Recuperar uma lista classificada (ranked list) dos termos mais frequentes;
- Executar uma pesquisa, visualizar os resultados;
- Analisar os resultados da pesquisa;
- Eliminar selectivamente os documentos do índice;
- Reconstruir os campos originais dos documentos, editar e reinserir no índice;
- Otimizar índices.
Nutch

✓ Aplicação de pesquisa web open-source;
✓ Crawlers;
✓ Link-graph database;
✓ Parsers de documentos (HTML, word, pdf, etc);
✓ Detecção de linguagem + charset
✓ Utiliza o Hadoop (DFS + MapReduce) para escalabilidade massiva;

Solr

✓ REST XML/http, JSON, APIs
✓ Pesquisa Faceted;
✓ Esquema de dados flexível;
✓ Hit Highlighting;
✓ Caching com configurações avançadas;
✓ Replicação;
✓ Interface para administração web;
✓ Solr Flare: interface de utilizador em Ruby on Rails