

Banco de Dados 2

11 – Representação Vetorial de Registros ou Documentos

Recuperação de Informações

- A **Recuperação da Informação (RI)** é o processo de selecionar e apresentar informações relevantes para atender a uma necessidade de um usuário dentro de **grandes coleções de dados não estruturados ou semiestruturados**, como a **Internet**.
- Em vez de recuperar dados exatos, como em **bancos de dados**, a RI foca em encontrar o material mais adequado para a consulta do usuário, utilizando técnicas como **indexação** e **busca avançada**.
- O objetivo é fornecer ao usuário uma resposta que corresponda à sua demanda, com sistemas que usam **classificações por relevância** e permitem a **navegação entre os resultados**.

Recuperação de Informações

- Suponha a existência de **4 documentos (D)**:
- **D1**: {carro azul rápido}
- **D2**: {carro rápido}
- **D3**: {automóvel azul}
- **D4**: {lancha branca lenta}

Recuperação de Informações

- Agora considere um **usuário** que efetua uma **consulta com dois termos**: “**carro azul**”.
- Em um **modelo tradicional de busca**, comum a todos os **bancos de dados**, emprega-se o chamado **modelo booleano**.

Modelo Booleano

- Baseia-se na **álgebra booleana**.
- Cada **documento** é representado por conjunto de termos (presença/ausência).
- As **consultas** usam operadores **AND, OR, NOT**.
- O resultado é binário: o documento ou satisfaz a consulta, ou não.

Modelo Booleano

- Consulta: "**carro AND azul**".
- D1: "**carro azul rápido**" → satisfaz → retornado
- D2: "**carro rápido**" → não satisfaz → não retornado
- D3: "**automóvel azul**" → não satisfaz → não retornado
- D4: "**lancha branca lenta**" → não satisafaz → não retornado.

Modelo Booleano

- A consulta anterior, com o operador AND, **retornaria** como válido apenas o **documento D1**.
- O **documento D2** realmente **não deve ser retornado** pois o **operador AND** exige a **ocorrência dos dois termos (Carro e Azul)**. Apenas **Carro** foi encontrado.
- Lamentavelmente o **documento D3 não é retornado**.
- O **documento D4** não possui qualquer um dos termos de pesquisa. Portanto, não é retornado.

Modelo Booleano

- Consulta: "**carro OR azul**".
- **D1: "carro azul rápido"** → satisfaz → retornado
- **D2: "carro rápido"** → satisfaz → retornado
- **D3: "automóvel azul"** → satisfaz → retornado
- **D4: "lancha branca lenta"** → não satisafaz → não retornado.

Modelo Booleano

- A consulta anterior, com o operador OR, retornaria como válidos os documentos **D1**, **D2** e **D3**.
- Os documento **D2** e **D3** possuem apenas um dos termos procurados: carro e azul, respectivamente.
- O documento **D1** possui ambos os termos.
- O documento **D4** não possui qualquer um dos termos de pesquisa. Portanto, não é retornado.

Modelo Booleano

- Consulta: "**NOT carro**".
- **D1: "carro azul rápido"** → não satisfaz → não retornado
- **D2: "carro rápido"** → não satisfaz → não retornado
- **D3: "automóvel azul"** → satisfaz → retornado
- **D4: "lancha branca lenta"** → satisafaz → retornado.

Modelo Booleano

- A consulta anterior, com o operador NOT, **retornaria** como válidos os **documentos D3 e D4**.
- Os **documento D1 e D2** possuem o termo a ser evitado na pesquisa: **carro**.
- Mas, perceba que **D3** possui o termo “**automóvel**” – um sinônimo de “**carro**”.

Modelo Booleano

- Vantagens do **Modelo Booleano**:
- **Simples de implementar e entender.**
- **Consulta precisa e determinística.**
- **Bom para conjuntos de documentos pequenos ou muito estruturados.**

Modelo Booleano

- Limitações do **Modelo Booleano**:
- Não diferencia graus de relevância; documentos parcialmente relevantes são descartados.
- Difícil lidar com consultas imprecisas ou vagamente formuladas.
- Não captura sinônimos ou proximidade semântica (ex.: "automóvel" ≠ "carro").

Modelo Vetorial

- **Alternativa ao Modelo Booleano.**
- **Representa documentos e consultas como vetores em um espaço de termos.**
- **Usa TF-IDF ou outros esquemas de peso para cada termo.**
- **A relevância é calculada via similaridade cosseno ou outras métricas.**
- **Resultado é rankeado (graus de relevância).**

Modelo Vetorial

- Consulta: "**carro azul**".
- D1: "**carro azul rápido**"
- D2: "**carro rápido**"
- D3: "**automóvel azul**"
- D4: "**lancha branca lenta**"
- D5: "**carro azul carro branco**"

TERMOS
AUTOMÓVEL
AZUL
BRANCA
BRANCO
CARRO
LANCHAS
LENTA
RÁPIDO

Modelo Vetorial

- Remoção de stopwords
- **Stopwords** são palavras muito frequentes em um idioma, mas que não carregam muito significado para diferenciar documentos.
- Exemplos em português: **de, a, o, e, com, por, que, para, em**.
- Se mantidas, essas palavras aparecem em quase todos os documentos e “atrapalham” a comparação.
- Então, geralmente removemos essas palavras durante o processamento.

Modelo Vetorial

TERMOS
AUTOMÓVEL
AZUL
BRANCA
BRANCO
CARRO
LANCHA
LENTA
RÁPIDO

- Em nossa lista de termos não havia qualquer **stopword** (artigos, e preposições pronomes).
- Se houvesse teríamos de extraí-los da lista.

Modelo Vetorial

- Caso o **conteúdo** de um dado **documento** fosse “**O carro azul está na garagem**”, com a **extração** das **stopwords** o **conteúdo** seria reduzido a “**carro azul garagem**”.

Modelo Vetorial

TERMOS	D1	D2	D3	D4	D5
AUTOMÓVEL			1		
AZUL	1		1		1
BRANCA				1	
BRANCO					1
CARRO	1	1			2
LANCHA				1	
LENTA				1	
RÁPIDO	1	1			

- A representação vetorial dos documentos:
D5(0,1,0,1,2,0,0,0)
- Significa que em D5 há 1 ocorrência do termo **azul**, 1 de **branco** e 2 de **carro**.

Modelo Vetorial

- Nessa **representação vetorial** a **primeira posição** é sempre relativa às ocorrências do termo “**automóvel**”.
- A **segunda posição** é relativa ao termo “**azul**”.
- A **terceira** é do termo “**branca**”.
- E assim por diante ...

Modelo Vetorial

- **Stemming e Lematização:**
- Ambos tentam reduzir as palavras à sua forma básica, mas de maneiras diferentes:
- **Stemming:**
- Técnica mais simples e mecânica. Remove sufixos e prefixos de forma automática, sem entender a gramática. Pode gerar radicais “estranhos”, mas ajuda a reduzir a variação.
- **Exemplo:**
- **amando, amaremos, amou → am**
- **carros, carro → carr**

Modelo Vetorial

- **Lematização:**
- Técnica mais sofisticada e linguística. Usa dicionários e regras da língua para reduzir a palavra à sua forma canônica (lema). Resultados são palavras corretas do idioma.
- Exemplo:
- **amando, amaremos, amou → amar**
- **carros → carro**
- **foi → ir**

Modelo Vetorial

Após lematização (forma canônica)

- D1 → "carro azul rápido"
 - *carro* → carro
 - *azul* → azul
 - *rápido* → rápido
- D2 → "carro rápido"
 - *carro* → carro
 - *rápido* → rápido
- D3 → "automóvel azul"
 - *automóvel* → automóvel
 - *azul* → azul
- D4 → "lancha branco lento"
 - *lancha* → lancha
 - *branca* → branco (forma canônica do adjetivo)
 - *lenta* → lento
- D5 → "carro azul carro branco"
 - *carro* → carro
 - *azul* → azul
 - *carro* → carro
 - *branco* → branco

TERMOS	D1	D2	D3	D4	D5
AUTOMÓVEL			1		
AZUL	1		1		1
BRANCO				1	1
CARRO	1	1			2
LANCHA				1	
LENTO					1
RÁPIDO	1	1			

Modelo Vetorial

TERMOS	D1	D2	D3	D4	D5
AUTOMÓVEL			1		
AZUL	1		1		1
BRANCO				1	1
CARRO	1	1			2
LANCHA				1	
LENTO				1	
RÁPIDO	1	1			

- Agora a representação vetorial de D1 e D5 ficou:

D1(0,1,0,1,0,0,1)

D5(0,1,1,2,0,0,0)

Modelo Vetorial

- Assim, agora nossos **5 documentos** são apresentados da seguinte maneira:

D1: carro, azul, rápido

D2: carro, rápido

D3: automóvel, azul

D4: lancha, branco, lento

D5: carro, azul, carro, branco

Consulta: "carro, azul".

Modelo Vetorial

#	Termo	D1	D2	D3	D4	D5	Frequência Total do Termo na Coleção de Documentos
1	Automóvel			1			1
2	Azul	1		1		1	3
3	Branco				1	1	2
4	Carro	1	1			2	4
5	Lancha				1		1
6	Lento				1		1
7	Rápido	1	1				2
	Tamanho do Documento	3	2	2	3	4	

Modelo Vetorial

- **Freqüência de Ocorrência do Termo (f_{ij}):**
 - É a **freqüência de ocorrência** do termo **K_i** no documento **d_j** .
- **Freqüência Total do Termo (F_i):**
 - É a **frequência total** do termo **K_i** na **coleção** inteira de **documentos**.
- **Frequênciа de Documento do Termo (n_i):**
 - É a quantidade de **documentos** nos quais o termo **K_i** ocorre.

Modelo Vetorial

#	Termo	D1	D2	D3	D4	D5	Frequência Total do Termo na Coleção de Documentos	Frequência de Documento do Termo
1	Automóvel			1			1	1
2	Azul	1		1		1	3	3
3	Branco				1	1	2	2
4	Carro	1	1			2	4	3
5	Lancha				1		1	1
6	Lento				1		1	1
7	Rápido	1	1				2	2
	Tamanho do Documento	3	2	2	3	4		

Modelo Vetorial

- Term Frequency (**TF_{ij}**):
 - $1 + \log f_{ij}$ se **f_{ij} > 0** – log na base 2.
 - **0** caso contrário.
- Quanto mais freqüentemente um **termo K_i** ocorrer no **texto** do **documento D_j** maior será a sua **freqüência de termo TF_{ij}**.
- **Termos com alta freqüência** são importantes para **descrever tópicos-chave** de um **documento**.

Modelo Vetorial

#	Termo	D1	D2	D3	D4	D5	TFi1	TFi2	TFi3	TFi4	TFi5
1	Automóvel			1			0	0	1	0	0
2	Azul	1		1		1	1	0	1	0	1
3	Branco				1	1	0	0	0	1	1
4	Carro	1	1			2	1	1	0	0	2
5	Lancha				1		0	0	0	1	0
6	Lento				1		0	0	0	1	0
7	Rápido	1	1				1	1	0	0	0
Tamanho do Documento		3	2	2	3	4					

$$\text{TFi5} = \text{TF45} = 1 + (\log(2;2)) // 1 mais log de 2 na base 2$$

Modelo Vetorial

- O termo “**carro**” no documento 5 (**D5**) tem **frequência 2 (f_{ij})**.
- O termo “**carro**” aparece **2** vezes no documento **D5**.
- **TF_{i5} = TF₄₅ = 1 + (log(2;2)) = 1 + 1 = 2**
- Então, **TF_{i5}** na linha 4 (**TF₄₅**) recebe valor **2**.

Modelo Vetorial

- O termo “**branco**” no documento 5 (**D5**) tem **frequência 1 (f_{ij})**.
- O termo “**branco**” aparece apenas **1** vez em **D5**.
- **TF_{i5} = TF₃₅ = 1 + (log(1;2)) = 1 + 0 = 1.**
- Então **TF_{i5}** na **linha 3 (TF₃₅)** recebe valor **1**.

Modelo Vetorial

- O termo “**lancha**” no documento 5 (**D5**) tem **frequência 0 (f_{ij})**.
- O termo “**lancha**” aparece nenhuma vez em **D5**.
- **TF_{i5} = TF₅₅ = 0.**
- Então **TF_{i5}** na **linha 5 (TF₅₅)** recebe valor **0**.

Modelo Vetorial

#	Termo	D1	D2	D3	D4	D5	TFi1	TFi2	TFi3	TFi4	TFi5
1	Automóvel			1			0	0	1	0	0
2	Azul	1		1		1	1	0	1	0	1
3	Branco				1	1	0	0	0	1	1
4	Carro	1	1			2	1	1	0	0	2
5	Lancha				1		0	0	0	1	0
6	Lento				1		0	0	0	1	0
7	Rápido	1	1				1	1	0	0	0
	Tamanho do Documento	3	2	2	3	4					

Modelo Vetorial

Ponderação TF-IDF

- O **IDF** é uma interpretação estatística da **especificidade dos termos**.
- Ela tornou-se a pedra fundamental da **ponderação de termos**.
- O **termo** é analisado em relação a sua **exaustividade** e a sua **especificidade**.

Modelo Vetorial

Ponderação TF-IDF

Exaustividade:

É uma propriedade das descrições de documentos.

A abrangência que ela provê para os tópicos principais de um documento.

Especificidade:

É uma propriedade dos termos de indexação.

É interpretada como quão bem um termo descreve o tópico de um documento.

Modelo Vetorial

Ponderação TF-IDF

Se adicionarmos novos termos do vocabulário a um documento ... :

- A **exaustividade** da descrição do documento **aumenta**.
- A **probabilidade** que o documento **satisfaga** uma dada **consulta** também **aumenta**.
- A **probabilidade** de **recuperação** **aumenta**.
- **Risco** de ser recuperado por consultas para as quais o **documento não é relevante**.

Modelo Vetorial

Ponderação TF-IDF

- Existe um **número ótimo de termos de indexação** que define a **exaustividade ótima** para a descrição de **documentos**.
- **Indexar um documento com todos os seus termos de indexação** – como fazem as **máquinas de busca** – **pode não ser a melhor solução**.

Modelo Vetorial

Ponderação TF-IDF

- Solução:
 - Utilizar **todos** os **termos** para indexação e distinguir a **importância** dos vários **termos**, ponderando-as de acordo com a sua **especificidade**.
 - A **especificidade** é uma propriedade da **semântica** do **termo**.
 - Um **termo** é mais ou menos específico dependendo do seu **significado**.

Modelo Vetorial

Ponderação TF-IDF

Exemplo: “bebida” e “chá”.

O termo “bebida” é logicamente **menos específico**.

O termo “chá” é **mais específico**.

Estratégia:

Considerar a **especificidade** como uma **função da utilização dos termos** – uma **propriedade estatística** em vez de uma propriedade **semântica** do termo.

Modelo Vetorial

Ponderação TF-IDF

- **Exaustividade:**
 - O número de termos de indexação que um documento possui.
- **Especificidade:**
 - É uma função do inverso do número de documentos nos quais o termo ocorreu.
 - Caso as descrições dos documentos fiquem mais longas então a especificidade dos termos de indexação tende a ficar mais baixa.

Modelo Vetorial

Ponderação TF-IDF

- Se um **termo** ocorrer em todos os **documentos** da coleção
- Então:
 - A **especificidade** do termo é **mínima**.
 - O **termo não é útil** para a **recuperação**.
- Ponderação de termos por especificidade:
 - **Peso dos termos** é uma função das **frequências relativas dos termos**.

$$\text{IDF} = \log \left(\frac{N}{n_i} \right)$$

Modelo Vetorial

Ponderação TF-IDF

#	Termo	D1	D2	D3	D4	D5	TFi1	TFi2	TFi3	TFi4	TFi5
1	Automóvel			1			0	0	1	0	0
2	Azul	1		1		1	1	0	1	0	1
3	Branco				1	1	0	0	0	1	1
4	Carro	1	1			2	1	1	0	0	2
5	Lancha				1		0	0	0	1	0
6	Lento				1		0	0	0	1	0
7	Rápido	1	1				1	1	0	0	0
	Tamanho do Documento	3	2	2	3	4					

Modelo Vetorial

Ponderação TF-IDF

IDF = log (N/ni)

IDF: Freqüência inversa de documentos do termo Ki.

N: Total de documentos da coleção.

ni: Quantidade de documentos em que o termo aparece.

Em nosso exemplo de 5 documentos os termos mais seletivos na coleção ocorrem em apenas um documento:

“**Automóvel**” (D3), “**Lancha**” (D4) e “**Lento**” (D4) aparecem em apenas **1 documento**.

Os menos seletivos ocorrem em todos os documentos.

“**Carro**” (D1, D2 e D5) e “**Azul**” (D1, D3 e D5) aparecem em **3 documentos**.

Modelo Vetorial

Ponderação TF-IDF

#	Termo	Quantidade de Documentos (ni)	Frequência Inversa de Documentos (IDFi)
1	Automóvel	1	2,321928095
2	Azul	3	0,736965594
3	Branco	2	1,321928095
4	Carro	3	0,736965594
5	Lancha	1	2,321928095
6	Lento	1	2,321928095
7	Rápido	2	1,321928095
Total Documentos (N) = 5			

$$\text{IDF} = \log (N/n_i)$$

IDF: Freqüência inversa de documentos do termo K_i .

N: Total de documentos da coleção.

n_i : Quantidade de documentos em que o termo aparece.

Modelo Vetorial

Ponderação TF-IDF

- O termo “automóvel”, por exemplo, possui um IDF aproximado de **2,3219**.
- **IDF = log (N/ni)** // logaritmo na base 2
- **IDF = log (5/1)** // Aparece em 1 de 5 docs
- **IDF = log (5) = 2,3219...**

Modelo Vetorial

Ponderação TF-IDF

- O termo “**branco**”, por exemplo, possui um **IDF** aproximado de **1,3219**.
- **IDF = log (N/ni)** // logaritmo na base 2
- **IDF = log (5/2)** // Aparece em 2 de 5 docs
- **IDF = log (2,5) = 1,3219...**

Modelo Vetorial

Ponderação TF-IDF

- Em **grandes coleções reais** espera-se que os **termos mais seletivos** sejam **substantivos e grupos de substantivos**.
- Os **termos menos seletivos** são geralmente artigos, conjunções e preposições (stopwords).

Modelo Vetorial

Ponderação TF-IDF

Doc original

Doc : www.filosofia.com

“Se o desonesto soubesse a vantagem de ser honesto, ele seria honesto ao menos por desonestidade.”

Sócrates

Operações de Texto

Doc : www.filosofia.com

desonesto / soubesse / vantagem / honesto / seria / honesto / menos/desonestidade/ socrates

Representação

Doc : www.filosofia.com

honesto	2
desonesto	1
soubesse	1
vantagem	1
seria	1
menos	1
desonestidade	1
socrates	1

Modelo Vetorial

Ponderação TF-IDF

- O esquema de **ponderação de termos** mais popular utiliza **pesos** que combinam os **fatores IDF** e as **frequências dos termos (TF)**.
- A Ponderação TF-IDF:
 - Se **$f_{ij} > 0$**
 - Então **$w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$**
 - Senão **$w_{ij} = 0$**
- Os **termos mais raros têm pesos mais altos** porque **são mais seletivos**.

#	Termo	D1	D2	D3	D4	D5	TFi1	TFi2	TFi3	TFi4	TFi5
1	Automóvel			1			0	0	1	0	0
2	Azul	1		1		1	1	0	1	0	1
3	Branco				1	1	0	0	0	1	1
4	Carro	1	1			2	1	1	0	0	2
5	Lancha				1		0	0	0	1	0
6	Lento				1		0	0	0	1	0
7	Rápido	1	1				1	1	0	0	0
	Tamanho do Documento	3	2	2	3	4					

#	Termo	Quantidade de Documentos (ni)	Frequência Inversa de Documentos (IDFi)
1	Automóvel	1	2,321928095
2	Azul	3	0,736965594
3	Branco	2	1,321928095
4	Carro	3	0,736965594
5	Lancha	1	2,321928095
6	Lento	1	2,321928095
7	Rápido	2	1,321928095
Total Documentos (N) = 5			

Modelo Vetorial

Ponderação TF-IDF

D1: carro, azul, rápido

D2: carro, rápido

D3: automóvel, azul

D4: lancha, branco, lento

D5: carro, azul, carro, branco

Consulta: "carro, azul".

Modelo Vetorial

Ponderação TF-IDF

- O argumento de pesquisa da consulta: “**carro azul**”.

- “**Carro**” em D1:

$$w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$$

$$w_{41} = (1 + \log 1) \times \log (5/3) = (1 + 0) \times (0,7369)$$

$$W_{41} = 0,7369$$

Modelo Vetorial

Ponderação TF-IDF

- O argumento de pesquisa da consulta:
“carro azul”.

- “**Azul**” em D1:

$$w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$$

$$w_{21} = (1 + \log 1) \times \log (5/3) = (1 + 0) \times (0,7369)$$

$$W_{21} = 0,7369$$

Modelo Vetorial

Ponderação TF-IDF

- Ranking de D1 = $w_{41} + w_{21}$
- Ranking de D1 = 0,7369 + 0,7369
- **Ranking de D1 = 1,4738**

Modelo Vetorial

Ponderação TF-IDF

- O argumento de pesquisa da consulta:
“carro azul”.

- “Carro” em D2:

$$w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$$

$$w_{42} = (1 + \log 1) \times \log (5/3) = (1 + 0) \times (0,7369)$$

$$W_{42} = 0,7369$$

Modelo Vetorial

Ponderação TF-IDF

- O argumento de pesquisa da consulta: “**carro azul**”.
- “**Azul**” em D2:

$$w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$$

$$w_{22} = (1 + \log 1) \times \log (5/3) = (0) \times (0,7369)$$

$$W_{22} = 0$$

Modelo Vetorial

Ponderação TF-IDF

- Ranking de D2 = $w_{42} + w_{22}$
- Ranking de D2 = $0,7369 + 0$
- **Ranking de D2 = 0,7369**
- **D1 = 1,4738**
- **D2 = 0,7369**

#	Termo	D1	D2	D3	D4	D5	TFi1	TFi2	TFi3	TFi4	TFi5
1	Automóvel			1			0	0	1	0	0
2	Azul	1		1		1	1	0	1	0	1
3	Branco				1	1	0	0	0	1	1
4	Carro	1	1			2	1	1	0	0	2
5	Lancha				1		0	0	0	1	0
6	Lento				1		0	0	0	1	0
7	Rápido	1	1				1	1	0	0	0
	Tamanho do Documento	3	2	2	3	4					

#	Termo	Quantidade de Documentos (ni)	Frequência Inversa de Documentos (IDFi)
1	Automóvel	1	2,321928095
2	Azul	3	0,736965594
3	Branco	2	1,321928095
4	Carro	3	0,736965594
5	Lancha	1	2,321928095
6	Lento	1	2,321928095
7	Rápido	2	1,321928095
Total Documentos (N) = 5			

Modelo Vetorial

Ponderação TF-IDF

- O argumento de pesquisa da consulta:
“carro azul”.

- “**Carro**” em D3:

$$w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$$

$$w_{43} = 0 \times \log (5/3) = (0) \times (0,7369)$$

$$W_{43} = 0$$

Modelo Vetorial

Ponderação TF-IDF

- O argumento de pesquisa da consulta: “**carro azul**”.

- “**Azul**” em D3:

$$w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$$

$$w_{23} = (1 + \log 1) \times \log (5/3) = (1) \times (0,7369)$$

$$W_{23} = 0,7369$$

Modelo Vetorial

Ponderação TF-IDF

- Ranking de D3 = $w_{43} + w_{23}$
- Ranking de D3 = 0 + 0,7369
- **Ranking de D2 = 0,7369**
- **D1 = 1,4738**
- **D2 = 0,7369**
- **D3 = 0,7369**

Modelo Vetorial

Ponderação TF-IDF

- O argumento de pesquisa da consulta: “**carro azul**”.

- “**Carro**” em D4:

$$w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$$

$$w_{44} = 0 \times \log (5/3) = (0) \times (0,7369)$$

$$W_{44} = 0$$

Modelo Vetorial

Ponderação TF-IDF

- O argumento de pesquisa da consulta: “**carro azul**”.

- “**Azul**” em D4:

$$w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$$

$$w_{24} = 0 \times \log (5/3) = (0) \times (0,7369)$$

$$W_{24} = 0$$

Modelo Vetorial

Ponderação TF-IDF

- Ranking de D4 = w44 + w24
- Ranking de D4 = 0 + 0 = 0

Modelo Vetorial

Ponderação TF-IDF

- **D1 = 1,4738**
- **D2 = 0,7369**
- **D3 = 0,7369**
- **D4 = 0**

#	Termo	D1	D2	D3	D4	D5	TFi1	TFi2	TFi3	TFi4	TFi5
1	Automóvel			1			0	0	1	0	0
2	Azul	1		1		1	1	0	1	0	1
3	Branco				1	1	0	0	0	1	1
4	Carro	1	1			2	1	1	0	0	2
5	Lancha				1		0	0	0	1	0
6	Lento				1		0	0	0	1	0
7	Rápido	1	1				1	1	0	0	0
	Tamanho do Documento	3	2	2	3	4					

#	Termo	Quantidade de Documentos (ni)	Frequência Inversa de Documentos (IDFi)
1	Automóvel	1	2,321928095
2	Azul	3	0,736965594
3	Branco	2	1,321928095
4	Carro	3	0,736965594
5	Lancha	1	2,321928095
6	Lento	1	2,321928095
7	Rápido	2	1,321928095
Total Documentos (N) = 5			

Modelo Vetorial

Ponderação TF-IDF

- O argumento de pesquisa da consulta: “**carro azul**”.

- “**Carro**” em D5:

$$w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$$

$$w_{45} = (1 + 1) \times \log (5/3) = (2) \times (0,7369)$$

$$W_{45} = 1,4738$$

Modelo Vetorial

Ponderação TF-IDF

- O argumento de pesquisa da consulta: “**carro azul**”.

- “**Azul**” em D5:

$$w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$$

$$w_{24} = (1 + 0) \times \log (5/3) = (1) \times (0,7369)$$

$$W_{24} = 0,7369$$

Modelo Vetorial

Ponderação TF-IDF

- Ranking de D5 = $w_{45} + w_{25}$
- Ranking de D5 = $1,4738 + 0,7369$
- **Ranking de D5 = 2,2107**

Modelo Vetorial

Ponderação TF-IDF

- **D1 = 1,4738**
- **D2 = 0,7369**
- **D3 = 0,7369**
- **D4 = 0**
- **D5 = 2,2107**

Modelo Vetorial

Ponderação TF-IDF

- O que significa esse resultado?

D1: carro, azul, rápido

D2: carro, rápido

D3: automóvel, azul

D4: lancha, branco, lento

D5: carro, azul, carro, branco

Consulta: "carro, azul".

A consulta no Modelo Booleano exibiria: D1 e D5.

Todavia, sem especificar qual dos dois documentos possui maior relevância.

A consulta no Modelo Vetorial exibiria: D5, D1, D2 e D3 (apenas D4 ficaria de fora).

Contudo, D5 é o documento mais relevante, seguido por D1 e D2 (com carro mas sem azul) em conjunto com D3 (com azul mas sem carro).

Outro Exemplo

documento 1 (d1):

TO DO IS TO BE.

TO BE IS TO DO.

documento 2 (d2):

TO BE OR NOT TO BE.

I AM WATH I AM.

documento 3 (d3):

I THINK THEREFORE I AM.

DO BE DO BE DO.

documento 4 (d4):

DO DO DO, DA DA DA.

LET IT BE, LET IT BE.

Outro Exemplo

#	Termo	Doc 1	Doc 2	Doc 3	Doc 4
01	TO	4	2	-	-
02	DO	2	-	3	3
03	IS	2	-	-	-
04	BE	2	2	2	2
05	OR	-	1	-	-
06	NOT	-	1	-	-
07	I	-	2	2	-
08	AM	-	2	1	-
09	WHAT	-	1	-	-
10	THINK	-	-	1	-
11	THEREFORE	-	-	1	-
12	DA	-	-	-	3
13	LET	-	-	-	2
14	IT	-	-	-	2
Tamanho do Documento		10	11	10	12

Outro Exemplo

#	Termo	Doc 1	Doc 2	Doc 3	Doc 4	Freqüência Total do Termo na Coleção
01	TO	4	2	-	-	6
02	DO	2	-	3	3	8
03	IS	2	-	-	-	2
04	BE	2	2	2	2	8
05	OR	-	1	-	-	1
06	NOT	-	1	-	-	1
07	I	-	2	2	-	4
08	AM	-	2	1	-	3
09	WHAT	-	1	-	-	1
10	THINK	-	-	1	-	1
11	THEREFORE	-	-	1	-	1
12	DA	-	-	-	3	3
13	LET	-	-	-	2	2
14	IT	-	-	-	2	2
Tamanho do Documento		10	11	10	12	

Outro Exemplo



#	Termo	Doc 1	Doc 2	Doc 3	Doc 4	Freqüência Total do Termo na Coleção	Freqüência de Documento do Termo
01	TO	4	2	-	-	6	2
02	DO	2	-	3	3	8	3
03	IS	2	-	-	-	2	1
04	BE	2	2	2	2	8	4
05	OR	-	1	-	-	1	1
06	NOT	-	1	-	-	1	1
07	I	-	2	2	-	4	2
08	AM	-	2	1	-	3	2
09	WHAT	-	1	-	-	1	1
10	THINK	-	-	1	-	1	1
11	THEREFORE	-	-	1	-	1	1
12	DA	-	-	-	3	3	1
13	LET	-	-	-	2	2	1
14	IT	-	-	-	2	2	1
Tamanho do Documento		10	11	10	12		

Outro Exemplo

#	Termo	Doc 1	Doc 2	Doc 3	Doc 4	TFi1	TFi2	TFi3	TFi4
01	TO	4	2	-	-	3	2	-	-
02	DO	2	-	3	3	2	-	2,585	2,585
03	IS	2	-	-	-	2	-	-	-
04	BE	2	2	2	2	2	2	2	2
05	OR	-	1	-	-	-	1	-	-
06	NOT	-	1	-	-	-	1	-	-
07	I	-	2	2	-	-	2	2	-
08	AM	-	2	1	-	-	2	1	-
09	WHAT	-	1	-	-	-	1	-	-
10	THINK	-	-	1	-	-	-	1	-
11	THEREFORE	-	-	1	-	-	-	1	-
12	DA	-	-	-	3	-	-	-	2,585
13	LET	-	-	-	2	-	-	-	2
14	IT	-	-	-	2	-	-	-	2
Tamanho do Documento		10	11	10	12				

Outro Exemplo

Ponderação de Termos			
#	Termo (Ki)	ni	IDFi
1	TO	2	1
2	DO	3	0,415
3	IS	1	2
4	BE	4	0
5	OR	1	2
6	NOT	1	2
7	I	2	1
8	AM	2	1
9	WHAT	1	2
10	THINK	1	2
11	THE THEREFORE	1	2
12	DA	1	2
13	LET	1	2
14	IT	1	2
N = 4			

Ponderação TF-IDF

#	termo	Doc 1	Doc 2	Doc 3	Doc 4
1	TO	3	2	-	-
2	DO	0,830	-	1,073	1,073
3	IS	4	-	-	-
4	BE	-	-	-	-
5	OR	-	2	-	-
6	NOT	-	2	-	-
7	I	-	2	2	-
8	AM	-	2	1	-
9	WHAT	-	2	-	-
10	THINK	-	-	2	-
11	THEREFORE	-	-	2	-
12	DA	-	-	-	5,170
13	LET	-	-	-	4
14	IT	-	-	-	4
Tamanho do Documento (normas dos vetores):		5,068	4,899	3,762	7,738

Outro Exemplo

Suponha que o usuário utilize os seguintes argumentos de pesquisa: “**TO LET**”.

“**TO**” no documento **Doc 1**:

$$w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$$

$$w_{ij} = (1 + \log 4) \times \log (4/2)$$

$$w_{ij} = (1 + 2) \times \log 2$$

$$w_{ij} = 3 \times 1 = 3$$

w_{ij} = 3

Outro Exemplo

- “**LET**” no documento “**Doc 1**”:
 - $w_{ij} = (1 + \log f_{ij}) \times \log (N/n_i)$
 - $w_{ij} = (0) \times \log (4/1) = 0$
 - $w_{ij} = 0$
- Ranking do documento **Doc 1** = $3 + 0 = 3$.
- “**TO**” no documento “**Doc 2**”: $w_{ij} = 2$.
- “**LET**” no documento “**Doc 2**”: $w_{ij} = 0$.
- Ranking do documento **Doc 2** = $2 + 0 = 2$.

Outro Exemplo

- **Ranking de Documentos** para a **consulta “TO LET”**:
 - Doc 1 = 3.
 - Doc 2 = 2.
 - Doc 3 = $0 + 0 = 0$.
 - Doc 4 = $0 + 4 = 4$.
- Portanto, essa **consulta não** retornaria **Doc 3**, mas indicaria **Doc 4**, **Doc 1** e **Doc 2**.

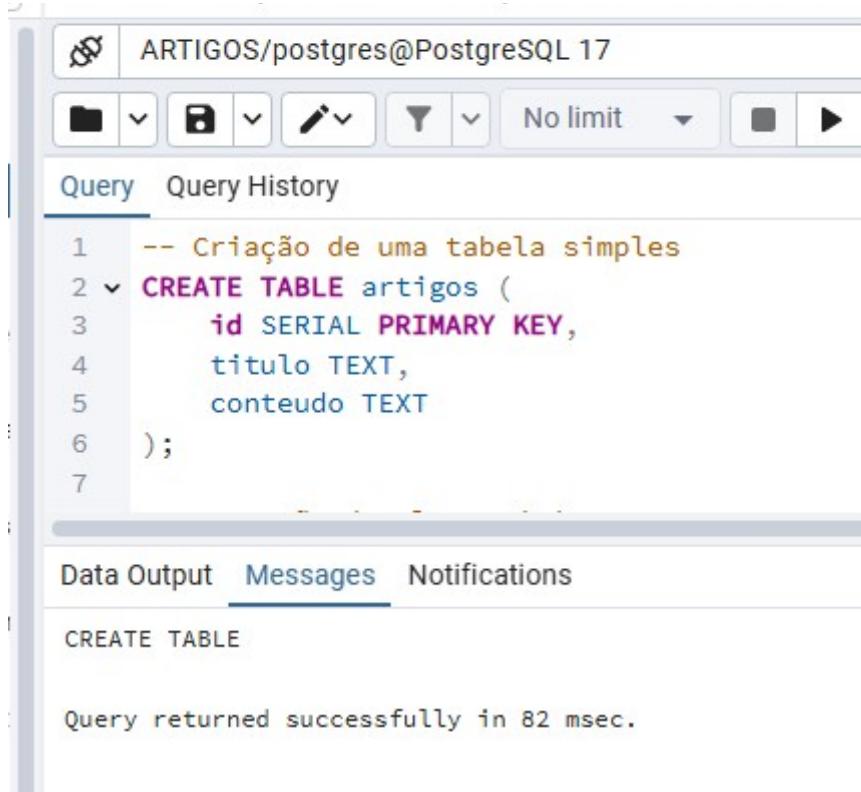
Full Text

- No PostgreSQL, as **pesquisas FULL TEXT** (ou Full-Text Search, FTS) são um recurso que permite realizar buscas textuais avançadas em documentos armazenados no banco de dados. Diferente do LIKE ou de expressões regulares, o FULL TEXT entende a linguagem natural e consegue:
 - Tokenizar (quebrar o texto em palavras ou lexemas);
 - Remover stopwords (palavras muito comuns, como “de”, “a”, “o”, “em”);
 - Normalizar palavras (stemming, reduzindo variações como “correr”, “correndo”, “correu” para a mesma raiz);
 - Ordenar resultados pela relevância (através de funções como `ts_rank`).

Full Text

- Como funciona no PostgreSQL.
- O PostgreSQL usa dois tipos principais de dados para Full Text Search:
 - **tsvector** → Representa um documento processado em forma de lista de lexemas.
 - **tsquery** → Representa a consulta (as palavras a buscar).

Full Text



The screenshot shows a PostgreSQL pgAdmin interface. The top bar displays the connection information: ARTIGOS/postgres@PostgreSQL 17. Below the bar are various toolbar icons. The main area is divided into two tabs: 'Query' (which is selected) and 'Query History'. The 'Query' tab contains the following SQL code:

```
1 -- Criação de uma tabela simples
2 CREATE TABLE artigos (
3     id SERIAL PRIMARY KEY,
4     titulo TEXT,
5     conteudo TEXT
6 );
7
```

Below the code, there are three tabs: 'Data Output', 'Messages' (which is selected), and 'Notifications'. The 'Messages' tab displays the output of the query:

CREATE TABLE

Query returned successfully in 82 msec.

- Banco de Dados:
artigos.

Full Text

Query Query History

```
1 -- Criação de uma tabela simples
2 CREATE TABLE artigos (
3     id SERIAL PRIMARY KEY,
4     titulo TEXT,
5     conteudo TEXT
6 );
7
8 -- Inserção de alguns dados
9 INSERT INTO artigos (titulo, conteudo) VALUES
10 ('Banco de Dados', 'O PostgreSQL oferece suporte a pesquisa full text.'),
11 ('Esportes', 'O time venceu a partida de futebol.'),
12 ('Tecnologia', 'A inteligência artificial está em alta.'),
13 ('PostgreSQL', 'É um banco de dados híbrido (relacional e orientado a objetos) que permite pesquisa full text'),
14 ('DB2', 'É um banco de dados relacional da IBM desenvolvido na década de 1970.'),
15 ('Flamengo', 'Em 1978 o Flamengo, time de futebol, foi campeão da taça guanabara com Canaterelli, Zico e Rondinelli');
16
```

Data Output Messages Notifications

INSERT 0 6

Query returned successfully in 77 msec.

Full Text

```
7
8  -- Inserção de alguns dados
9  ✓ INSERT INTO artigos (titulo, conteudo) VALUES
10 ('Banco de Dados', 'O PostgreSQL oferece suporte a pesquisa full text.'),
11 ('Esportes', 'O time venceu a partida de futebol.'),
12 ('Tecnologia', 'A inteligência artificial está em alta.'),
13 ('PostgreSQL', 'É um banco de dados híbrido (relacional e orientado a objetos) que permite pesquisa full text'),
14 ('DB2', 'É um banco de dados relacional da IBM desenvolvido na década de 1970.'),
15 ('Flamengo', 'Em 1978 o Flamengo, time de futebol, foi campeão da taça guanabara com Canaterelli, Zico e Rondinelli');
16
17 ✓ SELECT titulo, conteudo
18 FROM artigos
19 WHERE to_tsvector(conteudo) @@ to_tsquery('postgresql');
20
```

Data Output Messages Notifications



	titulo text	conteudo text	
1	Banco de Dados	O PostgreSQL oferece suporte a pesquisa full text.	

Full Text

```
8 -- Inserção de alguns dados
9 ✓ INSERT INTO artigos (titulo, conteudo) VALUES
10 ('Banco de Dados', 'O PostgreSQL oferece suporte a pesquisa full text.'),
11 ('Esportes', 'O time venceu a partida de futebol.'),
12 ('Tecnologia', 'A inteligência artificial está em alta.'),
13 ('PostgreSQL', 'É um banco de dados híbrido (relacional e orientado a objetos) que permite pesquisa full text'),
14 ('DB2', 'É um banco de dados relacional da IBM desenvolvido na década de 1970.'),
15 ('Flamengo', 'Em 1978 o Flamengo, time de futebol, foi campeão da taça guanabara com Canaterelli, Zico e Rondinelli');
16
17 ✓ SELECT titulo, conteudo
18 FROM artigos
19 WHERE to_tsvector(conteudo) @@ to_tsquery('postgresql');
20
21 ✓ SELECT titulo, conteudo
22 FROM artigos
23 WHERE to_tsvector(conteudo) @@ to_tsquery('full text');
24
25
```

Data Output Messages Notifications

ERROR: erro de sintaxe em tsquery: "full text"

ERRO: erro de sintaxe em tsquery: "full text"

SQL state: 42601

Full Text

```
20
21 -- Query Errada:
22 ▼ SELECT titulo, conteudo
23   FROM artigos
24 WHERE to_tsvector(conteudo) @@ to_tsquery('full text');
25
26 -- documentos que contenham as duas palavras (full e text):
27 ▼ SELECT titulo, conteudo
28   FROM artigos
29 WHERE to_tsvector(conteudo) @@ to_tsquery('full & text');
30
```

Data Output Messages Notifications



	titulo text	conteudo text
1	Banco de Dados	O PostgreSQL oferece suporte a pesquisa full text.
2	PostgreSQL	É um banco de dados híbrido (relacional e orientado a objetos) que permite pesquisa full text

Full Text

```
25
26 -- documentos que contenham as duas palavras (full e text):
27 ▼ SELECT titulo, conteudo
28 FROM artigos
29 WHERE to_tsvector(conteudo) @@ to_tsquery('full & text');
30
31 -- documentos que contenham qualquer uma das palavras (full ou text):
32 ▼ SELECT titulo, conteudo
33 FROM artigos
34 WHERE to_tsvector(conteudo) @@ to_tsquery('full | text');
35
```

Data Output Messages Notifications



	titulo text	conteudo text
1	Banco de Dados	O PostgreSQL oferece suporte a pesquisa full text.
2	PostgreSQL	É um banco de dados híbrido (relacional e orientado a objetos) que permite pesquisa full text

Full Text

```
35
36 -- Para consultas de texto digitado pelo usuário, normalmente não usamos to_tsquery()
37 -- (que exige essa sintaxe rígida), mas sim plainto_tsquery(), que interpreta a frase
38 -- de forma natural:
39 ▼ SELECT titulo, conteudo
40 FROM artigos
41 WHERE to_tsvector(conteudo) @@ plainto_tsquery('full text');
42
```

Data Output Messages Notifications



	titulo text	conteudo text
1	Banco de Dados	O PostgreSQL oferece suporte a pesquisa full text.
2	PostgreSQL	É um banco de dados híbrido (relacional e orientado a objetos) que permite pesquisa full text

Full Text

```
42
43 -- ordenar pelo grau de correspondência:
44 ▼ SELECT titulo, ts_rank(to_tsvector(conteudo), to_tsquery('postgresql')) AS rank
45 FROM artigos
46 WHERE to_tsvector(conteudo) @@ to_tsquery('futebol | flamengo')
47 ORDER BY rank DESC;
48
```

Data Output Messages Notifications



	titulo text	rank real
1	Esportes	0
2	Flamengo	0

Full Text

```
48
49 -- ordenar pelo grau de correspondência (exemplo 2):
50 SELECT titulo, ts_rank(to_tsvector(conteudo), to_tsquery('postgresql')) AS rank
51 FROM artigos
52 WHERE to_tsvector(conteudo) @@ to_tsquery('postgresql')
53 ORDER BY rank DESC;
54
```

Data Output Messages Notifications



	titulo text	rank real
1	Banco de Dados	0.06079271

Full Text

```
54
55 ✓ INSERT INTO artigos (titulo, conteudo) VALUES
56 ('SGBD', 'O PostgreSQL, MySQL e DB2 são exemplos de bancos de dados. Estudaremos o PostgreSQL. O MySQL foi descontinuado e o PostgreSQL permanece.');
57
58 -- ordenar pelo grau de correspondência (exemplo 3):
59 ✓ SELECT titulo, ts_rank(to_tsvector(conteudo), to_tsquery('postgresql')) AS rank
60 FROM artigos
61 WHERE to_tsvector(conteudo) @@ to_tsquery('postgresql | DB2')
62 ORDER BY rank DESC;
63
```

Data Output Messages Notifications



Showing rows: 1 to 2

	titulo text	rank real
1	SGBD	0.082745634
2	Banco de Dados	0.06079271

Full Text

- **Vantagens do FULL TEXT no PostgreSQL:**
- Busca muito mais eficiente que LIKE '%palavra%';
- Reconhecimento de variações linguísticas (stemming);
- Possibilidade de indexação com GIN ou GiST, tornando a pesquisa muito rápida mesmo em grandes volumes de texto;
- Ranking de relevância nativo (ts_rank, ts_rank_cd);
- Suporte a múltiplos idiomas (to_tsvector('portuguese', texto) por exemplo).

Full Text

- Tanto **ts_rank** quanto **ts_rank_cd** servem para **calcular a relevância de um documento em relação a uma consulta (tsquery)**, mas a forma de cálculo é diferente.
- **Em relação ao ts_rank:**
 - Usa frequência de ocorrência dos lexemas no documento.
 - Quanto mais vezes uma palavra da consulta aparece no documento, maior o peso.
 - Bom quando você quer dar destaque a textos que repetem várias vezes os termos pesquisados.

Full Text

Query Query History

```
1 v SELECT titulo, ts_rank(to_tsvector(conteudo), plainto_tsquery('postgresql db2')) AS rank
2   FROM artigos
3 WHERE to_tsvector(conteudo) @@ plainto_tsquery('postgresql')
4 ORDER BY rank DESC;
5
```

Data Output Messages Notifications

SQL

	titulo text	rank real
1	SGBD	0.14907968
2	Banco de Dados	1e-20

Full Text

- Em relação ao `ts_rank_cd` (Cover Density Ranking):
 - Usa o método de Cover Density.
 - Não se baseia tanto na quantidade de ocorrências, mas sim em como os termos aparecem agrupados e próximos.
 - Beneficia documentos onde os termos da consulta aparecem juntos, na mesma região do texto (melhor para buscas com várias palavras).
 - Evita que um texto longo e repetitivo fique artificialmente com pontuação maior.

Full Text

The screenshot shows the pgAdmin 4 interface. At the top, there's a connection bar with a connection icon and the text "ARTIGOS/postgres@PostgreSQL 17". Below it is a toolbar with various icons for file operations, search, and navigation. The main area has tabs for "Query" (which is selected) and "Query History". The "Query" tab contains two numbered SQL queries. The first query uses `ts_rank` and the second uses `ts_rank_cd`, both filtering by a specific PostgreSQL database name. The results grid at the bottom shows two rows of data: "SGBD" with a rank of 0.3 and "Banco de Dados" with a rank of 0.1.

```
1 SELECT titulo, ts_rank(to_tsvector(conteudo), plainto_tsquery('postgresql db2')) AS rank
2 FROM artigos
3 WHERE to_tsvector(conteudo) @@ plainto_tsquery('postgresql')
4 ORDER BY rank DESC;
5
6 SELECT titulo, ts_rank_cd(to_tsvector(conteudo), plainto_tsquery('postgresql')) AS rank
7 FROM artigos
8 WHERE to_tsvector(conteudo) @@ plainto_tsquery('postgresql')
9 ORDER BY rank DESC;
10
```

Data Output Messages Notifications

	titulo text	rank real
1	SGBD	0.3
2	Banco de Dados	0.1

Full Text

```
12
13 ✓ INSERT INTO ARTIGOS (TITULO, CONTEUDO) VALUES
14 ('EXEMPLO 1','Alessandra correu bastante'),
15 ('EXEMPLO 2','Mírian corre sempre'),
16 ('EXEMPLO 3','Sofia vive correndo muito'),
17 ('EXEMPLO 4','Renan gosta de correr maratonas inteiras');
18
19
```

Data Output Messages Notifications

INSERT 0 4

Query returned successfully in 110 msec.

Full Text

```
12
13 ✓ INSERT INTO ARTIGOS (TITULO, CONTEUDO) VALUES
14   ('EXEMPLO 1','Alessandra correu bastante'),
15   ('EXEMPLO 2','Mírian corre sempre'),
16   ('EXEMPLO 3','Sofia vive correndo muito'),
17   ('EXEMPLO 4','Renan gosta de correr maratonas inteiras');
18
19 ✓ SELECT titulo, conteudo, ts_rank_cd(to_tsvector(conteudo), plainto_tsquery('correr')) AS rank
20   FROM artigos
21   WHERE to_tsvector(conteudo) @@ plainto_tsquery('correr')
22   ORDER BY rank DESC;
23
```

Data Output Messages Notifications



	titulo text	conteudo text	rank real
1	EXEMPLO 1	Alessandra correu bastante	0.1
2	EXEMPLO 2	Mírian corre sempre	0.1
3	EXEMPLO 3	Sofia vive correndo muito	0.1
4	EXEMPLO 4	Renan gosta de correr maratonas inteiras	0.1