Minhash

Inteligência na Web e Big Data

Fabricio Olivetti de França e Thiago Ferreira Covões
folivetti@ufabc.edu.br, thiago.covoes@ufabc.edu.br

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

UFABC

CMCC
Centro de Matemática, Computação e Cognição
Center for Mathematics, Computation and Cognition

Minhash

Encontrando documentos similares

Uma tarefa frequente de mineração de textos:

- Dado um conjunto de textos, quais apresentam alta similaridade?
- Comparamos todos dois a dois, fazendo $N \cdot (N-1)/2$ comparações?
- E se N for muito grande?

Bag-of-Words matricial

Lembrando em nossa representação Bag-of-Words:

Token	D1	D2	D3	D4
A	1	0	0	1
B	0	0	1	0
C	0	1	0	1
D	1	0	1	1
E	0	0	1	0

Bag-of-Words matricial

Ela gera uma matriz esparsa e de alta dimensão!

Token	D1	D2	D3	D4
A	1	0	0	1
B	0	0	1	0
C	0	1	0	1
D	1	0	1	1
E	0	0	1	0

Bag-of-Words matricial

Uma forma de calcular a similaridade dos documentos é com o índice de Jaccard:

$$J(Di, Dj) = \frac{|Di \cap Dj|}{|Di \cup Dj|}$$

Bag-of-Words matricial

O cálculo da similaridade (de Jaccard) pode se tornar muito custosa.

Token	D1	D2	D3	D4
A	1	0	0	1
B	0	0	1	0
C	0	1	0	1
D	1	0	1	1
E	0	0	1	0

Assinaturas da matriz

Pré-calcular uma assinatura da matriz que é equivalente a sim. de Jaccard.

Token	D1	D2	D3	D4
A	1	0	0	1
B	0	0	1	0
C	0	1	0	1
D	1	0	1	1
E	0	0	1	0

Permutação

Se permutarmos aleatoriamente as linhas dessa matriz:

Token	D1	D2	D3	D4
B	0	0	1	0
A	1	0	0	1
E	0	0	1	0
C	0	1	0	1
D	1	0	1	1

Permutação

E representarmos cada documento pelo índice da primeira linha contendo 1:

Minhash	2	4	1	2
Token	D1	D2	D3	D4
B	0	0	1	0
A	1	0	0	1
E	0	0	1	0
C	0	1	0	1
D	1	0	1	1

Permutação

Dessa forma cada documento pode ser representado por apenas 1 valor!

Minhash	2	4	1	2
Token	D1	D2	D3	D4
B	0	0	1	0
A	1	0	0	1
E	0	0	1	0
C	0	1	0	1
D	1	0	1	1

Minhash

Qual a probabilidade de $mh(Di) = mh(Dj)$?

$P(mh(Di) = mh(Dj))_\pi = ?$

Minhash

Quantas linhas o valor 1 coincide dividido pelo total de linhas com pelo menos um valor 1:

$$P(mh(Di) = mh(Dj))_\pi = \frac{|Di \cap Dj|}{|Di \cup Dj|}$$

Minhash

Comparando $D1$ com $D4$

Minhash	2	4	1	2
Token	D1	D2	D3	D4
B	0	0	1	0
A	1	0	0	1
E	0	0	1	0
C	0	1	0	1
D	1	0	1	1

Minhash

2 linhas com os dois iguais a 1 3 linha com pelo menos um 1

Minhash	2	4	1	2
Token	D1	D2	D3	D4
B	0	0	1	0
A	1	0	0	1
E	0	0	1	0
C	0	1	0	1
D	1	0	1	1

Minhash

$$J = 2/3 = P(mh(D1) = mh(D4))$$

Minhash	2	4	1	2
Token	D1	D2	D3	D4
B	0	0	1	0
A	1	0	0	1
E	0	0	1	0
C	0	1	0	1
D	1	0	1	1

Minhash

Se temos $P = 2/3$, se fizermos 100 permutações diferentes, teremos cerca de 67 valores iguais de minhash entre os dois documentos!

Minhash

Cada documento pode ser representado por N minhashes diferentes.

$Di = [mh_1(Di), mh_2(Di), \ldots, mh_N(Di)]$

Minhash

Só que computar permutações custa caro computacionalmente. Vamos utilizar funções de hash para simplificar.

Minhash

Dada a função hash:

$$h(x) = (ax + b) \mod p$$

com x sendo o índice do atributo. Ela gera uma permutação aleatória com diferentes valores de a, b e mantendo p fixo.

Minhash

Sorteamos N valores de a e b e escolhemos um p primo.

Escolhemos um p grande o suficiente para representar nossas variáveis.

Minhash

Exemplo:

$h1(x) = (x + 1) \mod 5$

$h2(x) = (3x + 1) \mod 5$

Minhash

$h1(x) = (x + 1) \mod 5$

$h2(x) = (3x + 1) \mod 5$

Token	D1	D2	D3	D4	h1	h2
A (0)	1	0	0	1	1	1
B (1)	0	0	1	0	2	4
C (2)	0	1	0	1	3	2
D (3)	1	0	1	1	4	0
E (4)	0	0	1	0	0	3

Minhash

Cada documento é representado com a lista do menor valor de h_i para todas as funções hash, em que ele possui um valor 1 no token correspondente.

Token	D1	D2	D3	D4	h1	h2
A (0)	1	0	0	1	1	1
B (1)	0	0	1	0	2	4
C (2)	0	1	0	1	3	2
D (3)	1	0	1	1	4	0
E (4)	0	0	1	0	0	3

$SIG(D_1) = [1, 0]$

Token	D1	D2	D3	D4	h1	h2
A (0)	1	0	0	1	1	1
B (1)	0	0	1	0	2	4
C (2)	0	1	0	1	3	2
D (3)	1	0	1	1	4	0
E (4)	0	0	1	0	0	3

Minhash (Algoritmo)

SIG1 e SIG2 inicializam com o maior valor possível acrescido de um.

SIG1	5	5	5	5		
SIG2	5	5	5	5		
Token	D1	D2	D3	D4	h1	h2
A (0)	1	0	0	1	1	1
B (1)	0	0	1	0	2	4
C (2)	0	1	0	1	3	2
D (3)	1	0	1	1	4	0
E (4)	0	0	1	0	0	3

Minhash (Algoritmo)

Para cada linha, atualizamos os valores de cada documento que possui o valor 1 nessa linha.

SIG1	1	5	5	1		
SIG2	1	5	5	1		
Token	D1	D2	D3	D4	h1	h2
A (0)	1	0	0	1	1	1
B (1)	0	0	1	0	2	4
C (2)	0	1	0	1	3	2
D (3)	1	0	1	1	4	0
E (4)	0	0	1	0	0	3

Minhash (Algoritmo)

SIG1	1	5	2	1		
SIG2	1	5	4	1		
Token	D1	D2	D3	D4	h1	h2
A (0)	1	0	0	1	1	1
B (1)	0	0	1	0	2	4
C (2)	0	1	0	1	3	2
D (3)	1	0	1	1	4	0
E (4)	0	0	1	0	0	3

Minhash (Algoritmo)

SIG1	1	3	2	1		
SIG2	1	2	4	1		
Token	D1	D2	D3	D4	h1	h2
A (0)	1	0	0	1	1	1
B (1)	0	0	1	0	2	4
C (2)	0	1	0	1	3	2
D (3)	1	0	1	1	4	0
E (4)	0	0	1	0	0	3

Minhash (Algoritmo)

SIG1	1	3	2	1		
SIG2	0	2	0	0		
Token	D1	D2	D3	D4	h1	h2
A (0)	1	0	0	1	1	1
B (1)	0	0	1	0	2	4
C (2)	0	1	0	1	3	2
D (3)	1	0	1	1	4	0
E (4)	0	0	1	0	0	3

Minhash (Algoritmo)

SIG1	1	3	0	1		
SIG2	0	2	0	0		
Token	D1	D2	D3	D4	h1	h2
A (0)	1	0	0	1	1	1
B (1)	0	0	1	0	2	4
C (2)	0	1	0	1	3	2
D (3)	1	0	1	1	4	0
E (4)	0	0	1	0	0	3

$D1' = [1, 0], D2' = [3, 2], D3' = [0, 0], D4' = [1, 0]$

SIG1	1	3	0	1		
SIG2	0	2	0	0		
Token	D1	D2	D3	D4	h1	h2
A (0)	1	0	0	1	1	1
B (1)	0	0	1	0	2	4
C (2)	0	1	0	1	3	2
D (3)	1	0	1	1	4	0
E (4)	0	0	1	0	0	3

Minhash (Algoritmo)

$D1' = [1, 0], D2' = [3, 2], D3' = [0, 0], D4' = [1, 0]$

J	D2	D3	D4
D1	0	0,25	0,67
D2		0	0,33
D3			0,2

J	D2’	D3’	D4’
D1’	0	0,5	1
D2’		0	0
D3’			0,5

Minhash

Com isso podemos reduzir o número de atributos textuais de milhares para menos de 100!

Mas….e se N for muito alto? Ainda temos que comparar os documentos par a par…

Locality Sensitive Hashing

Dos $N \cdot (N-1)/2$ pares de documentos, apenas alguns poucos devem ser similares.

E se filtrarmos aqueles pares que com certeza não são similares, reduzindo o número de comparações?

Locality Sensitive Hashing

Considere a matriz de minhash:

D1	D2	D3	D4
SIG1	2	0	1	0
SIG2	1	3	0	1
SIG3	0	3	1	0
SIG4	0	2	1	0
SIG5	1	3	0	1
SIG6	0	2	0	0

Locality Sensitive Hashing

Vamos dividi-la em 2 faixas de 3 linhas:

D1	D2	D3	D4
SIG1	2	0	1	0
SIG2	1	3	0	1
SIG3	0	3	1	0
SIG4	0	2	1	0
SIG5	1	3	0	1
SIG6	0	2	0	0

Locality Sensitive Hashing

Cada faixa vai gerar um número de assinaturas hash como a composição das colunas:

D1	D2	D3	D4	
SIG1	2	0	1	0
SIG2	1	3	0	1
SIG3	0	3	1	0
SIG4	0	2	1	0
SIG5	1	3	0	1
SIG6	0	2	0	0

Faixa 1: $210, 033, 101, 010$

Faixa 2: $010, 232, 100, 010$

Locality Sensitive Hashing

Dado um número suficiente de faixas e linhas, esperamos que em algumas chaves hash ocorra colisão:

D1	D2	D3	D4	
SIG1	2	0	1	0
SIG2	1	3	0	1
SIG3	0	3	1	0
SIG4	0	2	1	0
SIG5	1	3	0	1
SIG6	0	2	0	0

Faixa 1: $\{D1\}, \{D2\}, \{D3\}, \{D4\}$

Faixa 2: $\{D1, D4\}, \{D2\}, \{D3\}$

Locality Sensitive Hashing

Os documentos que colidirem deve ter uma alta probabilidade de serem muito similares.

D1	D2	D3	D4	
SIG1	2	0	1	0
SIG2	1	3	0	1
SIG3	0	3	1	0
SIG4	0	2	1	0
SIG5	1	3	0	1
SIG6	0	2	0	0

Faixa 1: $\{D1\}, \{D2\}, \{D3\}, \{D4\}$

Faixa 2: $\{D1, D4\}, \{D2\}, \{D3\}$

Locality Sensitive Hashing

A probabilidade de que a assinatura s coincida em todas as r linhas de uma faixa é s^r.

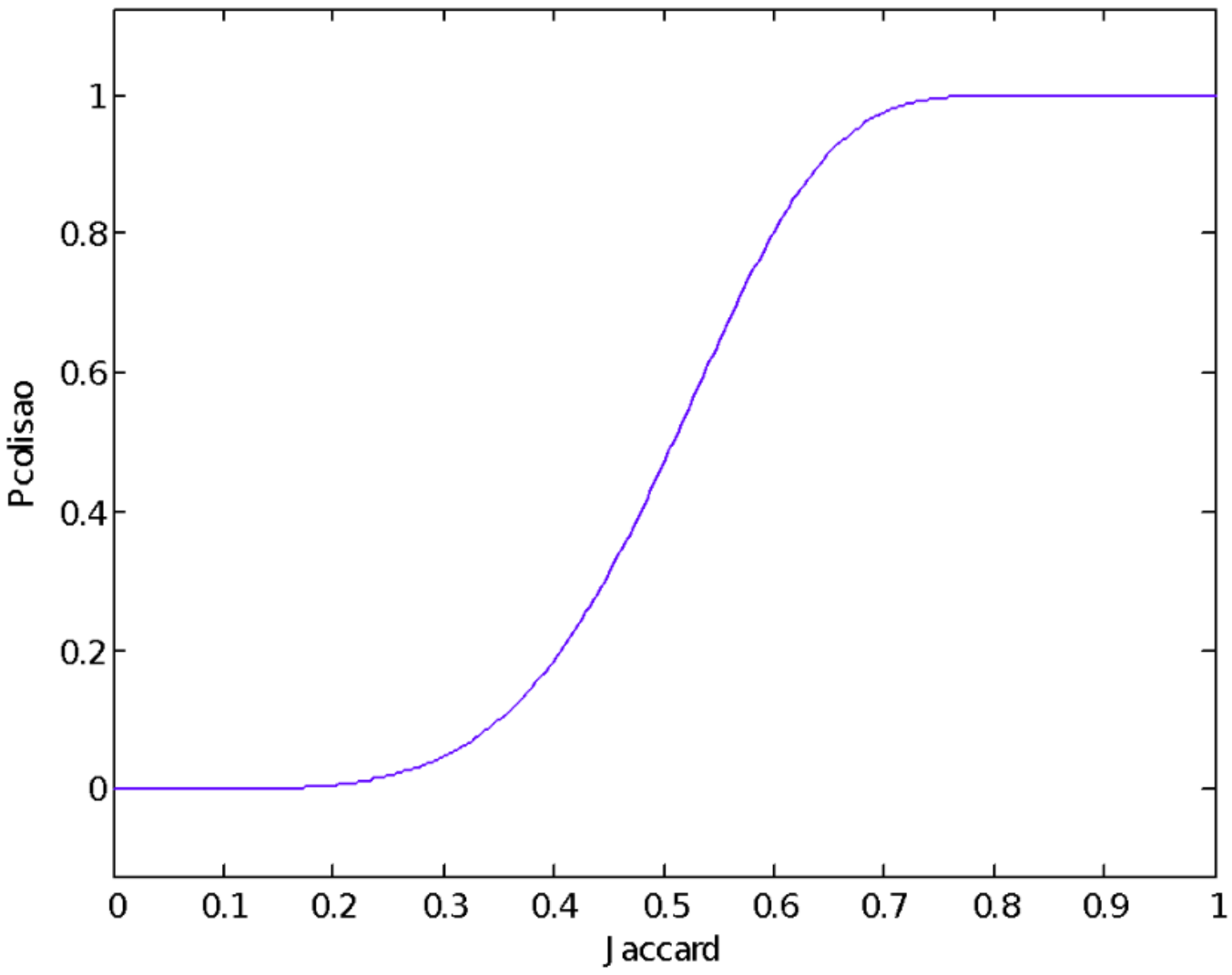
A probabilidade que não coincida em pelo menos um é $1 - s^r$.

A probabilidade que não coincida em todas as b faixas é $(1 - s^r)^b b$.

A probabilidade que coincida em pelo menos uma das b faixas é $1 - (1 - s^r)^b$.

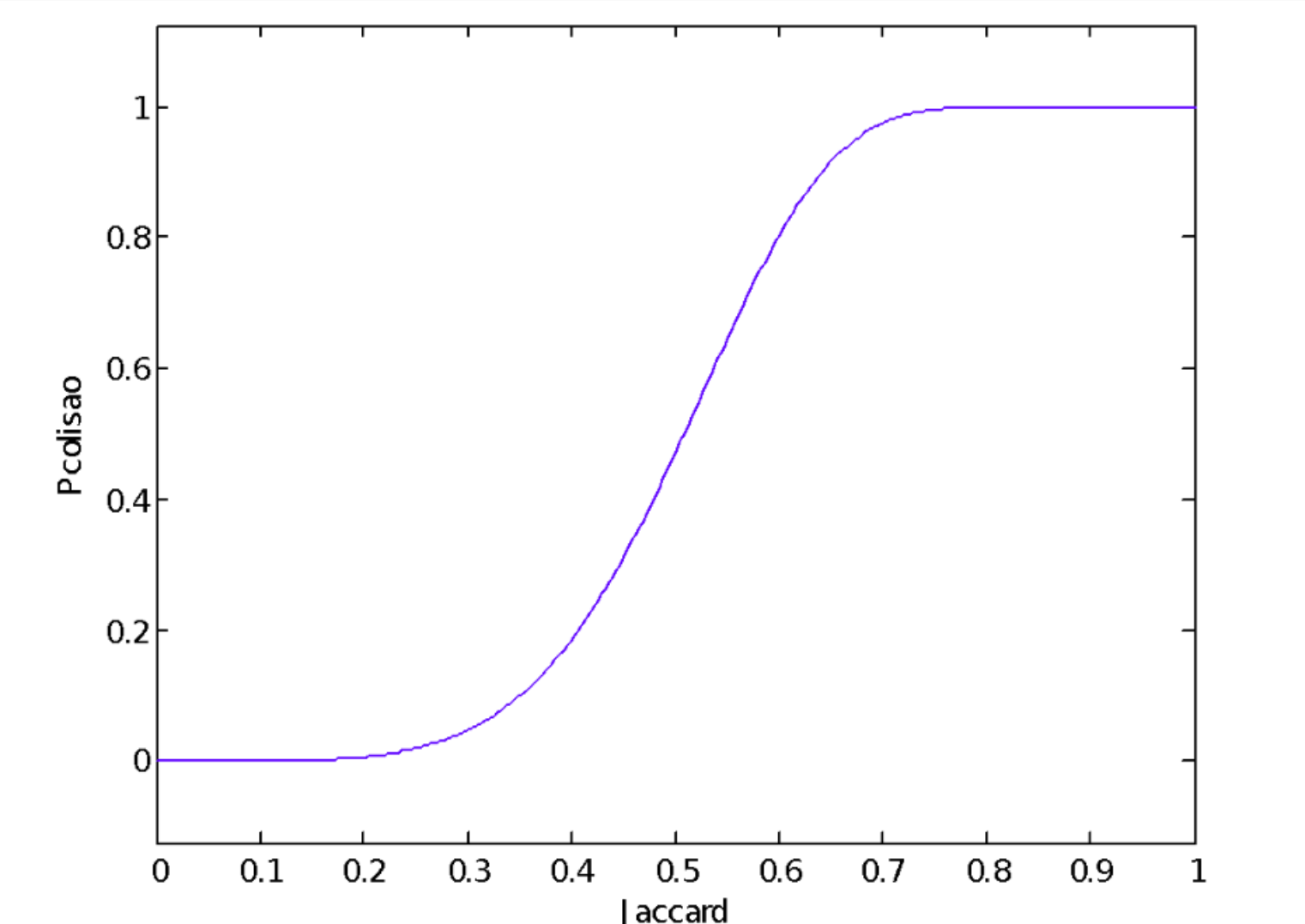
Locality Sensitive Hashing

Essa função tem o seguinte formato, independente dos valores de b e r:



Locality Sensitive Hashing

O limiar dessa curva se encontra aproximadamente em $(1/b)^{1/r}$.



Locality Sensitive Hashing

Escolha valores de b e r tal que $(1/b)^{1/r}$ resulte no limiar de similaridade desejado.

Faça $n = b \cdot r$ e gere n assinaturas hash dos documentos.

Crie as b funções de hash como a composição de r assinaturas para cada banda.

Remova todos os buckets em que não há colisão, calcule a similaridade apenas dos documentos que colidiram.

Locality Sensitive Hashing

Assuma uma função $f()$, e que $f(x) = f(y)$ signifique que x e y devem formar um par e $f(x) \neq f(y)$ indique que não devem formar um par.

Locality Sensitive Hashing

Dada uma função de distância d e dois limiares d_1 e d_2, uma família de funções F é (d_1, d_2, p_1, p_2) -sensível:

- Se $d(x, y) \leq d1$, a probabilidade de que $f(x) = f(y)$ é $p1$.
- Se $d(x, y) \geq d2$, a probabilidade de que $f(x) \neq f(y)$ é $p2$.

Locality Sensitive Hashing

O minhash é $d_1, d_2, 1 - d_1, 1 - d_2$-sensível, ou seja, se definirmos:

$d_1 = 0.3$ e $d_2 = 0.6$ temos que se a similaridade de Jaccard entre x e y for ≥ 0.7, temos uma probabilidade de 0.7 de colisão.

Da mesma forma se a similaridade for ≤ 0.4, temos uma probabilidade de 0.4 de não colidirem.

Construção-E

Dado um conjunto F de funções sensíveis, podemos compor r funções com a conjunção formando a função f':

$f'(x) = f'(y)$ se e somente se $f_i(x) = f_i(y)$ para todo $i = 1..r$

Isso transforma nossa função em $(d1, d2, (p1)^r, (p2)^r)$ -sensível

Construção-OU

Dado um conjunto F de funções sensíveis, podemos compor b funções formando a função f':

$f'(x) = f'(y)$ se $fi(x) = fi(y)$ para pelo menos um de $i = 1..r$

Isso transforma nossa função em $(d1, d2, 1 - (1 - p1)^b, 1 - (1 - p2)^b)$ -sensível.

Construções

Escolhendo adequadamente b e r podemos ajustar os valores de $p1$ e $p2$ de acordo com $d1, d2$.

Digamos que queremos maximizar a colisão para $d = 0.3$ e minimizar para $d = 0.6$, se escolhermos $b = 100$ e $r = 5$ temos:

$(0.3, 0.7, 0.99, 0.21)$ -sensível

r - reduz as probabilidade; b - aumenta

Comentários Finais

Comentários Finais

- Nem sempre a solução para tratar grande massa de dados é o uso de computação distribuída.
- O uso de funções de hash e suas probabilidades de colisão podem formar um algoritmo escalável que encontra os pares de documentos similares com alta probabilidade.