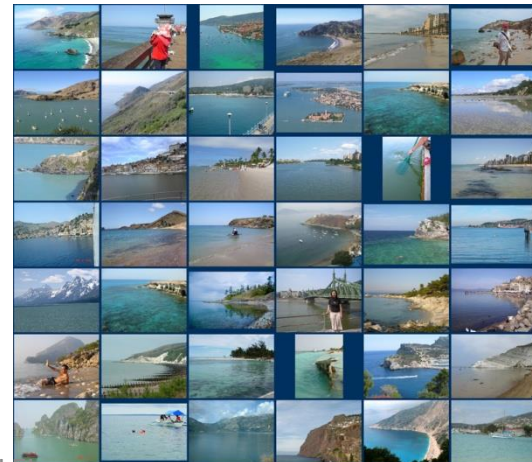


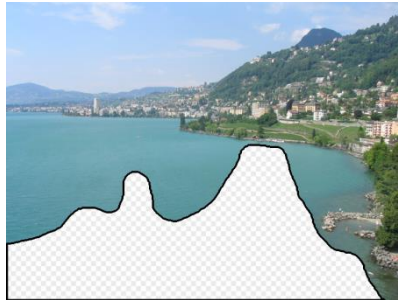
# MPEI

Solução Probabilística  
para a Procura de Similares

# Problema Exemplo – Completar cena

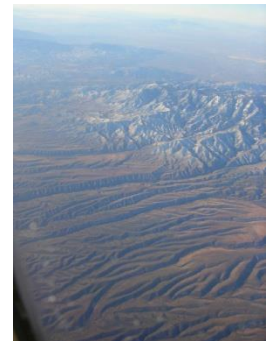
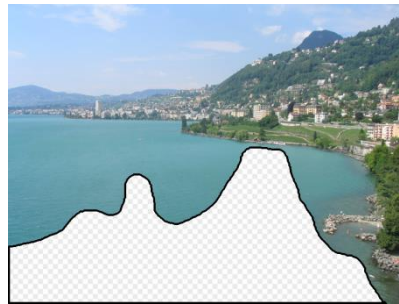
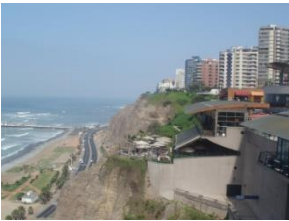


# Problema Exemplo – Completar cena



# Problema Exemplo – Completar cena

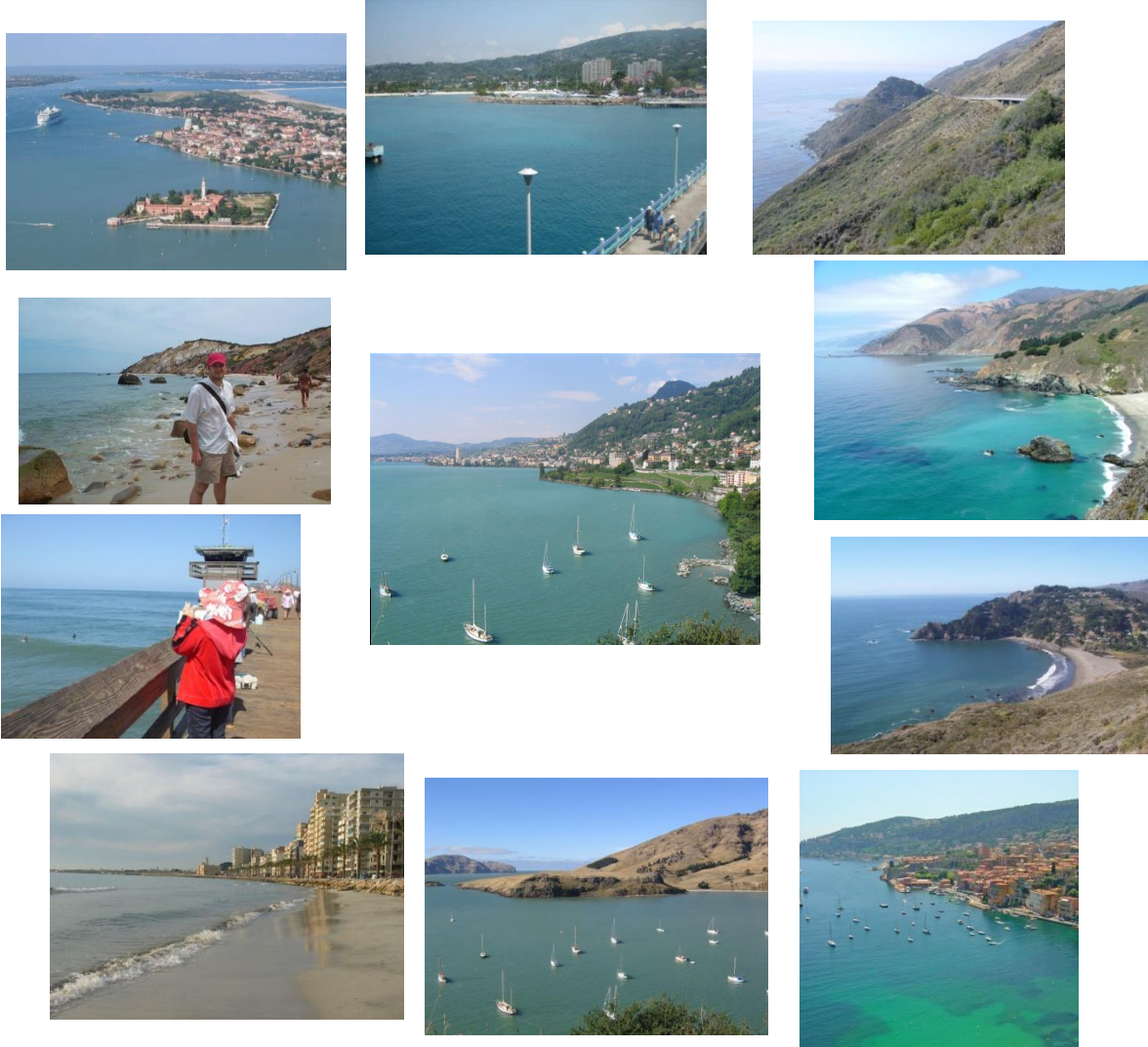
- 10 imagens mais próximas numa coleção de 20 000 imagens





# Problema Exemplo – Completar cena

10 imagens mais próximas num conjunto de 2 milhões de imagens



# Generalizando

- Muitos problemas podem ser expressos em termos da **descoberta de conjuntos similares**
- Exemplos:
  - Páginas web similares
    - para deteção de potenciais cópias
  - Clientes que compraram produtos similares
  - Imagens com características similares
  - Utilizadores que visitam sites similares
  - Clientes que compraram livros similares

# Generalizando (continuação)

- Em todos estes problemas temos entidades que podem ser representadas por um conjunto
  - páginas web, compras, imagens, utilizadores, ...
- Exemplo:
  - As páginas web podem ser representadas pelo conjunto das palavras que contêm

# Definição do Problema

- Tendo:
- pontos  $x_1, x_2, \dots$  num espaço com  $n$  dimensões
  - Exemplo: imagem é um vetor com as cores dos pixels
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow [1 \ 2 \ 1 \ 0 \ 2 \ 1 \ 0 \ 1 \ 0]$$
- uma função de distância  $d(x_1, x_2)$ 
  - Que quantifica a distância entre  $x_1$  e  $x_2$
- **Objectivo:** Determinar todos os pares de dados  $(x_i, x_j)$  com distância igual ou inferior a um determinado limiar  $s$ ,  $d(x_i, x_j) \leq s$



# Solução ingénua

- **Comparar todos os pares** possíveis
- Com  $N$  pontos, teríamos complexidade  $O(N^2)$
- Muito demorada ou mesmo impossível em tempo útil para  $N$  grande
- Exemplo:
  - 1 milhão de documentos ( $N = 10^6$ )
  - temos de calcular a similaridade para cada par
    - $N(N - 1)/2 = 5 \times 10^{11}$  comparações
  - Com 86400 s/dia e  $10^6$  comparações/s, levaria mais de 5 dias
  - Se tivermos 10 milhões é muito pior, demora mais de um ano.

# Distância

- O objectivo é determinar os vizinhos mais próximos no espaço  $n$ -dimensional
- Formalmente vizinhos próximos (near neighbors) são pontos que se encontram distanciados por uma “pequena distância”
- Em primeiro lugar tem de se definir o que significa distância

# Distância e Similaridade de Jaccard

- A **similaridade** (ou semelhança) **de Jaccard** de 2 **conjuntos** é definida pelo quociente entre a dimensão da sua **interseção** e dimensão da sua **união**:

$$\text{sim}(\mathbf{C}_1, \mathbf{C}_2) = |\mathbf{C}_1 \cap \mathbf{C}_2| / |\mathbf{C}_1 \cup \mathbf{C}_2|$$

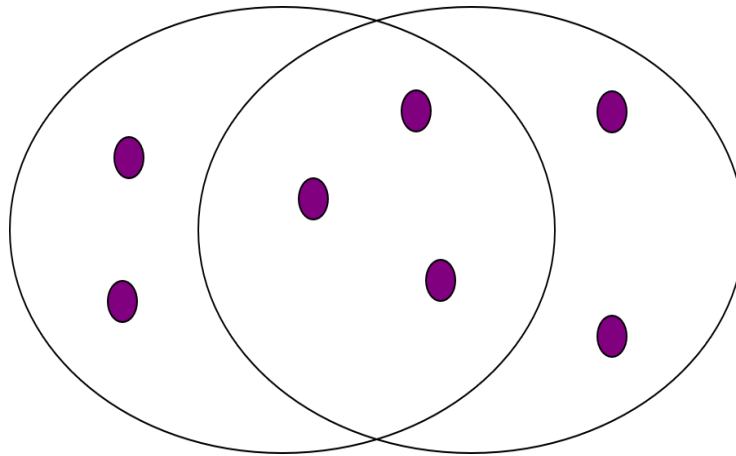
— representando  $||$  o nº de elementos do conjunto ou cardinalidade do conjunto

- A **distância Jaccard**,  $d_j$ , é obtida diretamente da similaridade:

$$d_j(\mathbf{C}_1, \mathbf{C}_2) = 1 - |\mathbf{C}_1 \cap \mathbf{C}_2| / |\mathbf{C}_1 \cup \mathbf{C}_2|$$

# Exemplo

- Qual a distância de Jaccard entre os 2 conjuntos?



- Resolução
  - Temos 3 elementos na interseção e 7 na união ...
  - $d_j = 1 - 3/7 = 4/7$

# Exemplo em Matlab

- Qual a similaridade entre as seguintes strings
- `str1='When nine hundred years old you reach, look as good you will not.'`
- `str2='You will not look as good when nine hundred years old'`
- `C1=unique(strsplit(lower(str1)));`
- `C2=unique(strsplit(lower(str2)));`
- `simJ=length(intersect(C1,C2))/ length(union(C1,C2))`
- Resultado: `simJ = 0.7692`

# Aplicação Exemplo #1 (Matlab)

- Detetar textos similares
- Exemplo para demonstração (toy example)
- Conjuntos serão as palavras (únicas) dos textos
  - Sem pós-processamento
- Aplicação direta da distância de Jaccard



# Aplicação Exemplo #1

## Tarefas principais

1. Criar Conjuntos com as palavras de todos os documentos

`Sets{1}=getSetOfWordsFromFile('texto1.txt')`

`Sets{2}=getSetOfWordsFromFile('texto2.txt')`

...

2. Calcular a distância de Jaccard para todos os pares

`distJ=calcDistancesJ(Sets);`

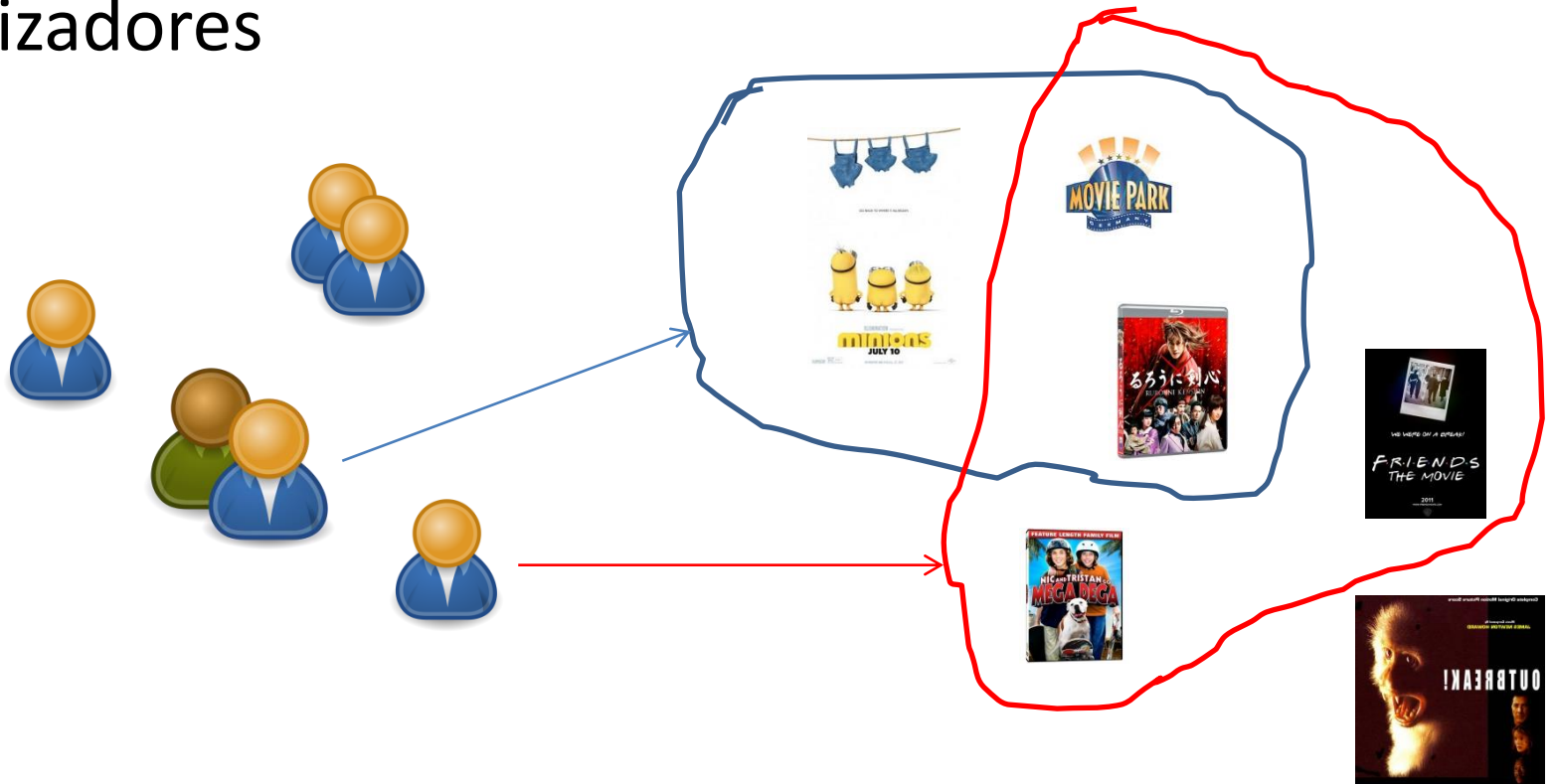
3. Determinar os pares que têm distância inferior a um certo limiar

`Similar=findSimilar(distJ,threshold,ids);`

4. Mostrar resultados

# Aplicação Exemplo #2

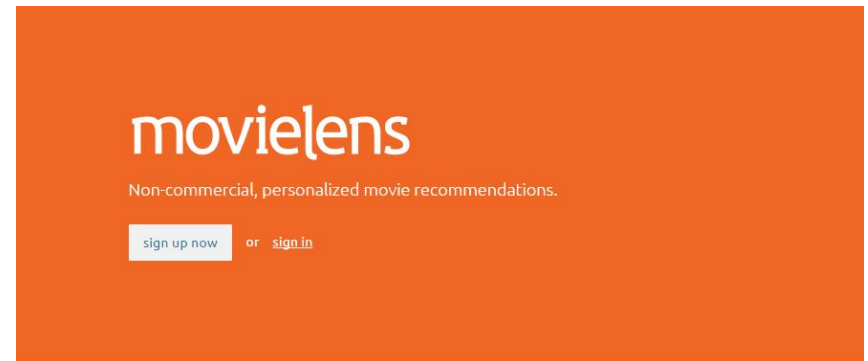
- Determinar conjuntos de filmes avaliados por utilizadores



- Assunto de um dos Guiões Práticos

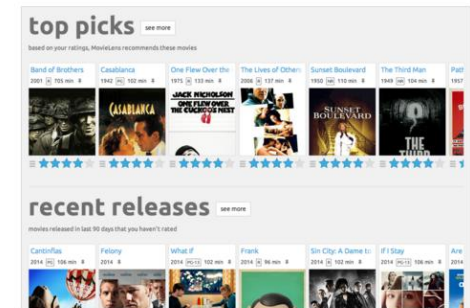
# MovieLens

- MovieLens  
(<http://movielens.org>)  
é um site que ajuda na  
selecção de filmes a ver
- Tem muitos utilizadores  
registados



## recommendations

MovieLens helps you find movies you will like. Rate movies to build a custom taste profile, then MovieLens recommends other movies for you to watch.



# Conjuntos de dados MovieLens

- GroupLens Research criou e disponibilizou **conjuntos de dados** (data sets) do site MovieLens
  - Disponíveis em: <http://grouplens.org/datasets/movielens/>
- Os dados foram recolhidos ao longo de diferentes períodos de tempo
- Existem vários conjuntos, de vários tamanhos:
  - **MovieLens 100K Dataset**
  - MovieLens 1M Dataset
  - ...

# Conjunto de dados MovieLens 100K

- Conjunto de dados estável e usado como padrão (benchmark)
  - Disponibilizado em 4/1998
- Aprox. 100 000 avaliações
  - de 943 utilizadores, sobre 1682 filmes
- [README](#)
- Link: <http://grouplens.org/datasets/movielens/100k/>

# Ficheiro **u.data**

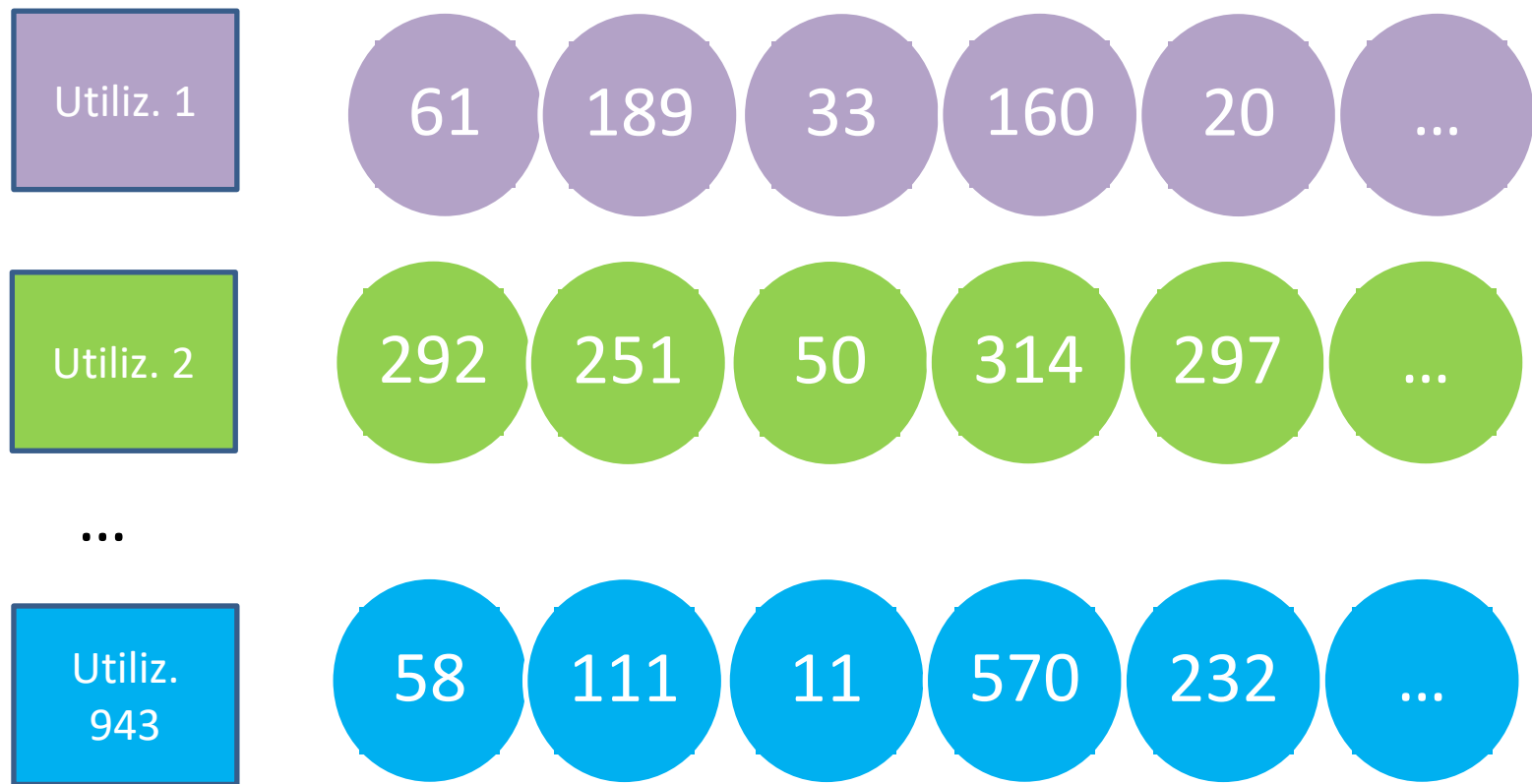
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596
298	474	4	884182806
115	265	2	881171488
253	465	5	891628467
305	451	3	886324817
...			

- 1ª coluna contém o ID do utilizador
- 2ª coluna o ID do filme
  - avaliado pelo utilizador da 1ª coluna
- Avaliação na 3ª coluna
- 4ª coluna é Informação de tempo (timestamp)



# Conjuntos de filmes

(avaliados por cada utilizador)



# Demonstração

- Primeira “solução”
  - Utilização direta da distância de Jaccard
- Muito lento 😞

# Resultado

- Results (similar sets):

328 788 distance = 0.327

408 898 distance = 0.161

489 587 distance = 0.370

# Conjuntos Grandes e Gigantes

- **Objetivo:**

Dado um **grande número de documentos (N)**,  
determinar pares “quase iguais”

–  $N =$  **milhões, milhares de milhões, bilhões, ...**

# Conjuntos Grandes e Gigantes

- **Problemas:**
- Demasiados documentos para se compararem todos os pares
- Muitas partes de um documento podem aparecer por outra ordem noutro
- Documentos são tão grandes ou número elevado que não cabem em memória

# Conjuntos Grandes e Gigantes

## Solução

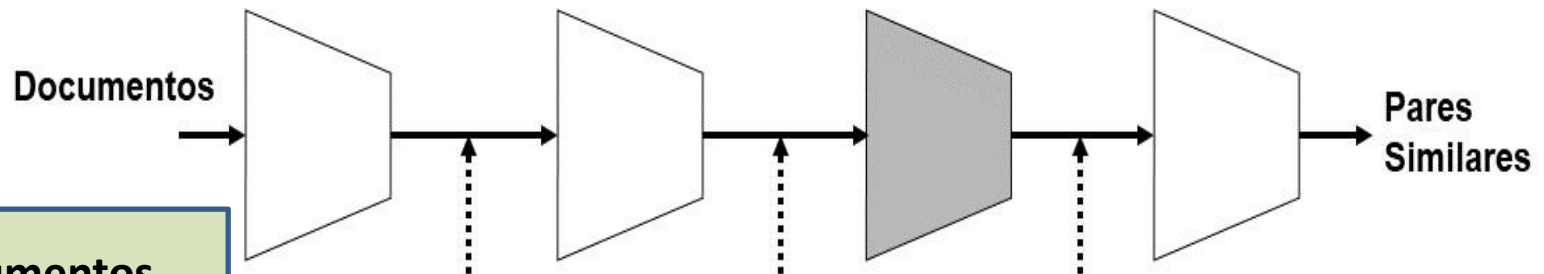
- Para  $N$  grande ou muito grande a solução passa por:
  1. Reduzir a dimensão dos conjuntos
    - mantendo a informação essencial à determinação de distância entre eles
  2. Reduzir o tempo de cálculo da distância e/ou reduzir os pares a que se tem de aplicar essa distância.
- A resolução destes problemas é habitualmente feita em sequência



# Abordagem probabilística

## Visão geral

- Processo de determinação de documentos similares:



### Conjuntos:

Representativos  
dos documentos

### Assinaturas:

Pequenos vetores  
que representam o  
conjunto e  
preservam  
Informação sobre a  
sua similaridade

### Pares candidatos:

Pares de assinaturas a  
testar em termos de  
similaridade

**Documentos**  
Designação geral  
Inclui desde  
documentos de  
texto a imagens e  
dados multimédia

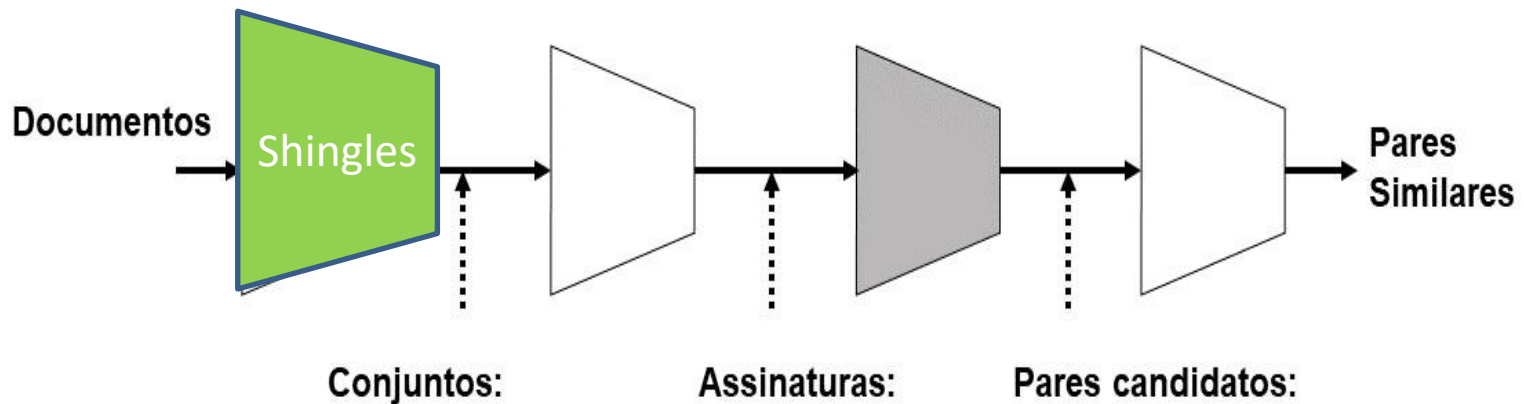
# Processo

- O processo consiste nas seguintes etapas sequenciais:
  1. Obtenção dos conjuntos representativos
  2. Redução desses conjuntos a conjuntos de dimensão fixa e pequena

O resultado é usualmente designado por assinatura

  3. (opcional) Processamento das assinaturas por forma a identificar pares potencialmente similares
  4. Cálculo de similaridade dos pares de conjuntos
    - todos ou os resultantes do passo anterior

# 1 - Conversão dos documentos em conjuntos



# Criação dos conjuntos representativos

- O objetivo desta primeira etapa é **criar os conjuntos representativos dos documentos**
  - A informação relevante a reter depende obviamente do tipo de documento
- Sem perda de generalidade, **consideraremos documentos constituídos por palavras**
  - ou para sermos mais precisos, sequências de caracteres
- A aplicação a outro tipo de documentos pode fazer-se adaptando o apresentado para sequências de caracteres
  - Por exemplo, no caso de imagens pode considerar-se como equivalente à palavra o valor de cada pixel (valor inteiro ou triplo RGB)

# Soluções

- As soluções mais simples são:
  1. conjunto de palavras que ocorrem no documento
  2. conjunto das palavras “importantes”.
- Ambas sofrem do mesmo problema:
  - não preservam informação sobre a ordem de ocorrência
- A ordem de ocorrência pode ser tida em conta utilizando sequências de palavras (ou de caracteres), ideia na base dos *k-gramas*
  - também conhecidos por *k-shingles*
  - ou simplesmente *Shingles*

# Shingles

- Um  $k$ -shingle (ou  $k$ -grama) para um documento é uma **sequência de  $k$  símbolos** que aparecem no documento
- Os símbolos podem ser caracteres, palavras ou outra informação, dependendo da aplicação
  - ex: código de cor de um pixel
- Assume-se que **documentos que têm muitos *Shingles* em comum são semelhantes**
- Utilizando *Shingles*, um **documento  $D$  é representado pelo conjunto dos seus  $k$ -gramas  $C = S(D)$**



# Exemplo

- Quais os Shingles do documento contendo a sequência de caracteres 'ab<sup>ab</sup>cab' considerando  $k=2$
- Conjunto de 2-shingles:  $S(D) = \{ab, bc, ca\}$
- Se aceitamos repetições:  $S'(D) = \{ab, bc, ca, ab\}$ 
  - ab aparece 2 vezes

# Similaridade para Shingles

- Representando um documento  $D_i$  pelo seu conjunto de  $k$ -shingles  $C_i = S(D_i)$
- Uma medida natural de similaridade é a similaridade de Jaccard
  - calculada com base nos conjuntos de Shingles representativos dos documentos

$$\text{sim}(D_1, D_2) = \text{sim}(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$

# Escolha de $k$

- A escolha de  $k$  não é trivial
- Deve escolher-se  $k$  suficientemente grande para evitar que a maioria dos documentos tenha a maioria dos *Shingles*
  - evitando desta forma que a generalidade dos documentos sejam representados pelo mesmo conjunto.
- Na prática:
  - $k = 5$  é bom para documentos curtos
  - e  $k = 10$  é mais adequado para documentos longos.

# Representação binária

- Para simplificar cálculo de interseção e união, os documentos podem ser representado por um vetor de zeros e uns no espaço de k-gramas (vetor binário)
  - em que cada Shingle/k-grama é uma dimensão
- Nesta representação a interseção e união são operações de bits (AND e OR)
- Os vetores de um conjunto de documentos formam uma matriz

# Exemplo

- 4 documentos:
  - D1='aab'
  - D2='bcd'
  - D3='cda'
  - D4='cd'
- 2-shingles (sem repetição) existentes nos 4 documentos:
  - $S(D) = \{aa, ab, bc, cd, da\}$
- Usando as linhas para os diferentes shingles e pela ordem em  $S(D)$  temos a matriz:

aa	1	0	0	0
ab	1	0	0	0
bc	0	1	0	0
cd	0	1	1	1
da	0	0	1	0
	D1	D2	D3	D4

# Exemplo (continuação)

- $d(D_2, D_3) = ?$
- $= d(C_2, C_3) = ?$
- $C_2 = 00110$
- $C_3 = 00011$
- $|Interseção| = 1$ 
  - Bitwise AND
- $|União| = 3$ 
  - Bitwise OR
- Sim Jaccard =  $1/3$
- $d(C_2, C_3) = 1 - \text{simJ} = 2/3$

aa	1	0	0	0
ab	1	0	0	0
bc	0	1	0	0
cd	0	1	1	1
da	0	0	1	0
	D1	D2	D3	D4

# Outro Exemplo

- Cada documento é uma coluna
  - $C_i$  representa o Documento  $i$
- **Exemplo:**  $\text{sim}(C_1, C_2) = ?$
- Comprimento da **interseção** = 3
- Comprimento da **união** = 6
- Similaridade de Jaccard similarity (não é a distância) =  $3/6$
- $d(C_1, C_2) = 1 - (\text{similaridade Jaccard}) = 3/6$

Documents

1	1	1	0
1	1	0	1
0	1	0	1
0	0	0	1
1	0	0	1
1	1	1	0
1	0	1	0

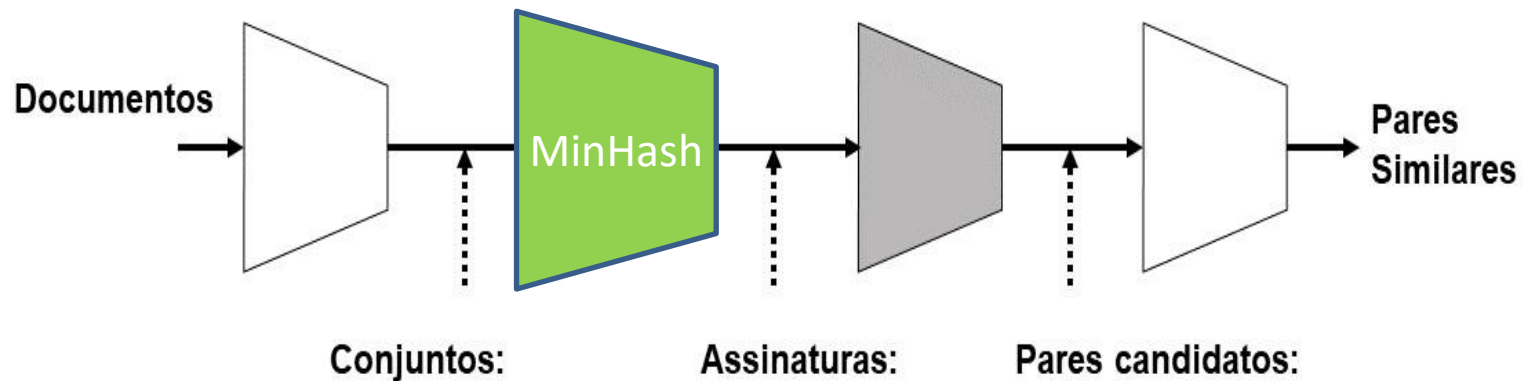
Shingles

# A seguir: Descobrir Colunas Similares

- **Até agora:**
  - Documentos → Conjuntos de Shingles
  - Representação dos conjuntos por vetores booleanos numa matriz
- **Próximo objectivo:** Descobrir colunas similares usando representações compactas (assinaturas)



## 2 - Cálculo das Assinaturas



# Assinaturas

- Uma das etapas importantes do processo é a redução da representação dos conjuntos
- Ideia base:
- Mapear cada conjunto  $C_i$  para uma pequena assinatura  $Sig(C_i)$  através de funções rápidas, tal que:
  1.  $Sig(C)$  é suficientemente pequena para que a assinatura de um número muito grande de conjuntos possa ser mantida em memória RAM
  2. A similaridade das assinaturas  $Sig(C_1)$  e  $Sig(C_2)$  é aproximadamente igual à  $sim(C_1, C_2)$

# Desafio

- Obter uma função de dispersão  $h()$  tal que:
  - Se  $\text{sim}(C_1, C_2)$  é elevada, então com elevada probabilidade  $h(C_1) = h(C_2)$
  - Se  $\text{sim}(C_1, C_2)$  é baixa, então  $h(C_1) \neq h(C_2)$  com elevada probabilidade
- A função  $h()$  depende da métrica de similaridade
- Para a similaridade de Jaccard a função Min-Hash cumpre os requisitos

# Redução do conjunto usando permutações

- Uma forma de reduzir o conjunto  $C$  representativo de um documento é considerar apenas um subconjunto
- A seleção pode ser feita usando **permutações aleatórias**:
  - Aplicação de uma permutação aleatória  $\pi$  às linhas da matriz booleana
  - Reter o valor do índice da primeira linha (na ordem permutada) correspondente a um Shingle (ou equivalente) existente

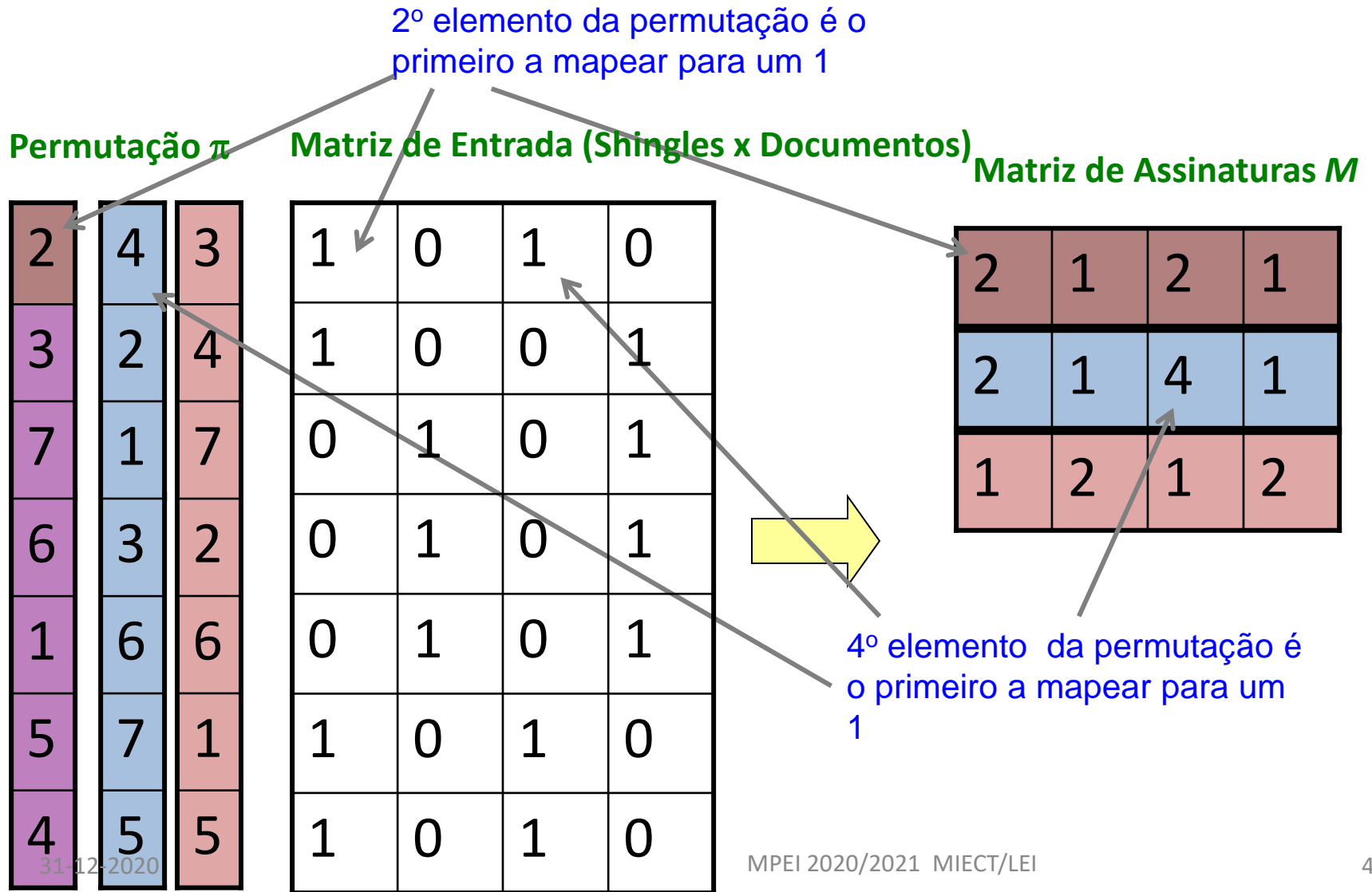
# Mínimo da função

- Esta operação pode ser vista como a aplicação de uma **função de dispersão**  $h_{\pi}(C) = \text{índice da primeira linha}$  (na ordem permutada ) **na qual a coluna C tem valor 1**

$$h_{\pi}(C) = \min_{\pi} \pi(C)$$

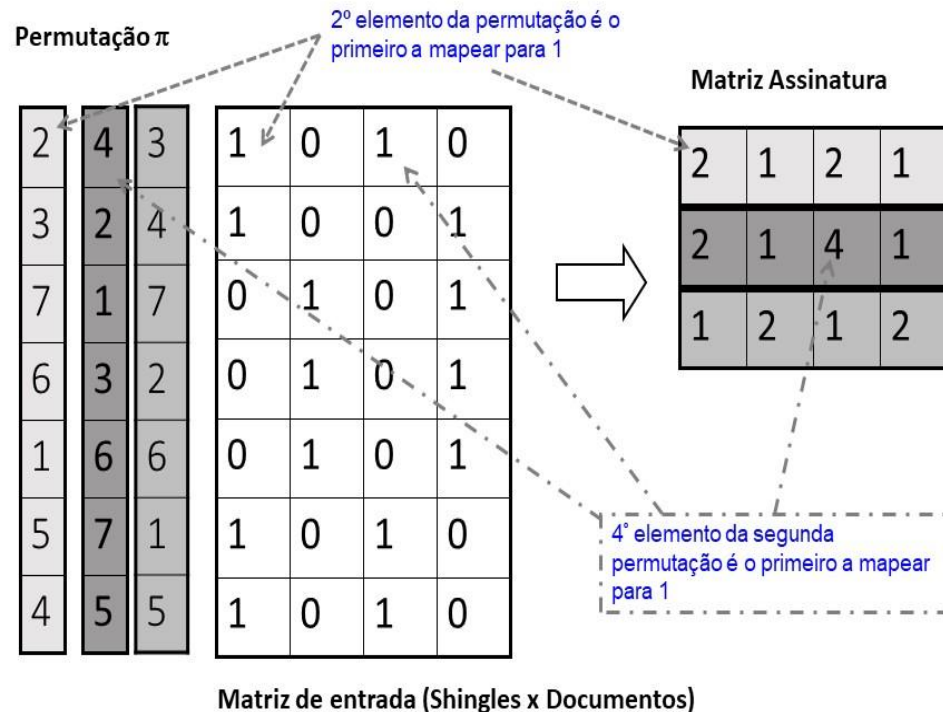
- Repetir para várias permutações independentes, por exemplo 100, para obter um vetor (assinatura)
  - Este processo de repetição com permutações independentes pode ser visto como a aplicação de várias funções  $h_{\pi}()$

# Exemplo



# Exemplo (continuação)

- 4 documentos representados num espaço de 7 *Shingles* diferentes
- Na primeira permutação, a primeira linha da matriz será a quinta, a segunda a primeira, etc.
- Em consequência, na primeira permutação, o valor a considerar para o primeiro documento será 2, primeira linha da permutação com 1 na primeira coluna da matriz dos documentos
- De forma similar se obtém o resto da primeira linha da matriz assinatura
- As outras 2 linhas da matriz assinatura são obtidas usando as outras duas permutações



# Propriedade de $h_\pi$

- A função  $h_\pi()$  permite obter assinaturas pois estas mantêm informação sobre a similaridade dos documentos devido à seguinte propriedade
- Para uma permutação aleatória,  $\pi$

$$P[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$$



# “Demonstração”

- Considere-se o documento  $D$  (conjunto de *shingles*), e  $s \in D$  um shingle
- Como é igualmente provável que qualquer  $s \in D$  seja mapeado para o mínimo
- Então:  $P[\pi(s) = \min(\pi(D))] = 1/|D|$

# “Demonstração” (cont.)

- O universo dos valores de  $\min(\pi(C_1 \cup C_2))$  é  $|C_1 \cup C_2|$
- Em consequência temos  $|C_1 \cup C_2|$  casos possíveis
  - Todos os Shingles podem ser o mínimo
- Por outro lado, em  $|C_1 \cap C_2|$  casos temos  $\min(\pi(C_1)) = \min(\pi(C_2))$ 
  - Ou seja  $|C_1 \cap C_2|$  casos favoráveis
- Tendo em conta a equiprobabilidade, o que interessa é a dimensão dos conjuntos, temos:

$$\Pr[\min(\pi(C_1)) = \min(\pi(C_2))] = |C_1 \cap C_2| / |C_1 \cup C_2|$$

- A Similaridade de Jaccard

# Propriedade de $h_\pi$

- Para duas colunas A and B:
- $h(A) = h(B)$  quando o mínimo da função de hash faz parte da interseção  $A \cap B$ .

A	B
0	0
0	0
<b>1</b>	<b>1</b>
0	0
0	1
1	0

# Confirmação por Simulação

- A propriedade do Min-Hash anterior pode ser comprovada através de simulação

```
%% criar dois conjuntos C1 e C2 (colunas de bits)
limiar1=rand(); limiar2=rand();
C1=rand(NS,1)>limiar1;
C2=rand(NS,1)>limiar2;

%% calcular distância de Jaccard
% (com operações com bits)

I= C1 & C2; % interset
U= C1 | C2;
dJaccard=1-(sum(I)/sum(U));
fprintf(1,'distância Jaccard=%.4f\n',dJaccard);
```

# Confirmação por Simulação

```
% estimar com permutações
EXPERIMENTS=10000; cfav=0;      p=zeros(1,EXPERIMENTS);

for i=1:EXPERIMENTS
    % permutar
    pi1=C1(randperm(NS));      pi2=C2(randperm(NS));

    % mínimos
    min1=min(find(pi1>0));      min2=min(find(pi2>0));

    % se minimos iguais
    if (min1==min2) cfav=cfav+1; end
    p(i)=cfav/i;
end

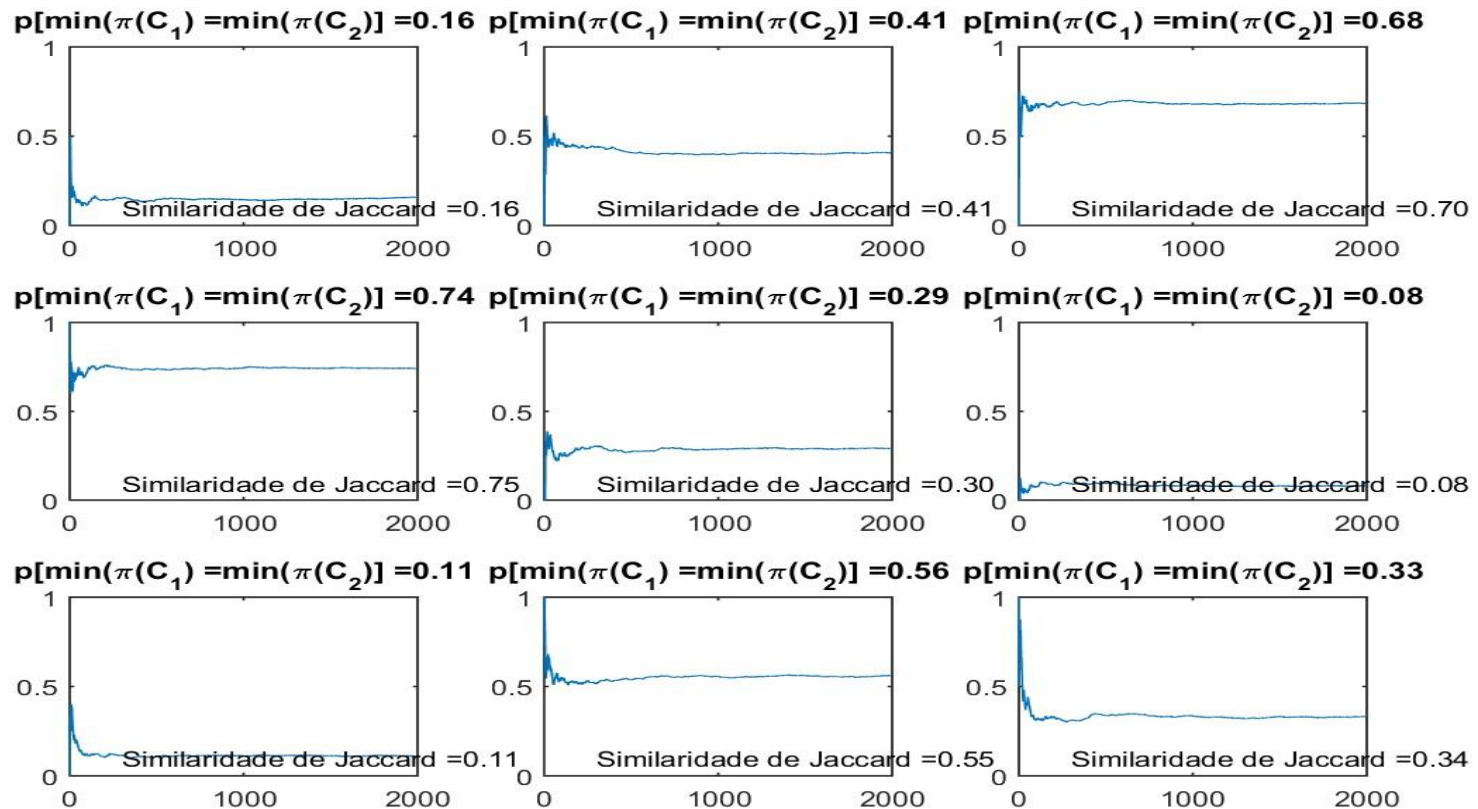
figure(1);
plot(1:i,p(1:i)); title(['p[min(\pi(C_1) =min(\pi(C_2))] =' ...
                        sprintf('%.3f',p(i))]);

sim=cfav/EXPERIMENTS;      distancia=1-sim;

fprintf(1,'Distância por simulação=%.4f\n',distancia);
```

# Confirmação por Simulação

- Exemplo de resultados



# Assinaturas

- Selecione **k permutações aleatórias** das linhas
- Pense na assinatura  **$\text{sig}(\mathbf{C})$**  como um vetor
- **$\text{sig}(\mathbf{C})[i]$  = índice da primeira linha** da coluna **C** **que contém 1** de quando **aplicada a permutação  $i$**  a essa linha
$$\text{sig}(\mathbf{C})[i] = \min (\pi_i(\mathbf{C}))$$
- **Notq:** A **assinatura de um documento é pequena**
  - **$\sim 100$  bytes para  $k=100$**  😊
- **Atingimos um dos nossos objetivos!**
  - **“Comprimimos” vetores de bits longos em pequenas assinaturas**

# Similaridade das Assinaturas

- Vimos que:  $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$
- Generalizando para várias funções
- A similaridade de 2 assinaturas é a fração de funções em que concordam
- **Nota:** Devido à propriedade de  $h_{\pi}$ , a similaridade das colunas é a mesma que a similaridade esperada das suas assinaturas



# Exemplo de aplicação

Permutações  $\pi$

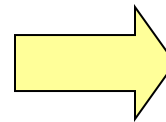
2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Matriz (Shingles x Documentos)

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Matriz de assinaturas  $M$

2	1	2	1
2	1	4	1
1	2	1	2



Similaridades:

Col/Col

Assin

	1-3	2-4	1-2	3-4
Col/Col	0.75	0.75	0	0
Assin	0.67	1.00	0	0

# Na prática...

- Permutar linhas para conjuntos de dimensão elevada é demorado e em geral **proibitivo**
- No entanto, o que necessitamos é de uma função que determine os índices e calcular o seu mínimo
- Uma solução eficiente **é utilizar uma função de dispersão**
  - evitando ao máximo colisões
  - O mapeamento efetuado pela função dá-nos a permutação aleatória
- Como a solução consiste na determinação **do mínimo dos valores de uma função de dispersão (hash)** é conhecido por **Min-Hash**

# A propriedade mantém-se ?

- Mantemos a propriedade usando as funções de dispersão?
- Sim.
- Seja  $h()$  essa **função de dispersão** que mapeia os membros de  $C_1$  e  $C_2$  para inteiros distintos
- Seja  $h_{min}(C)$  o membro  $x$  de  $C$  com o valor mínimo
- Se aplicarmos  $h_{min}()$  a  $C_1$  e  $C_2$ , **obteremos o mesmo valor** exatamente quando o elemento da união  $C_1 \cup C_2$  com valor **mínimo da função de dispersão estiver na intersecção**  $C_1 \cap C_2$
- A probabilidade disso ser verdade é

$$|C_1 \cap C_2| / |C_1 \cup C_2|$$

- e portanto:

$$P[h_{min}(C_1) = h_{min}(C_2)] = sim_J(C_1, C_2)$$

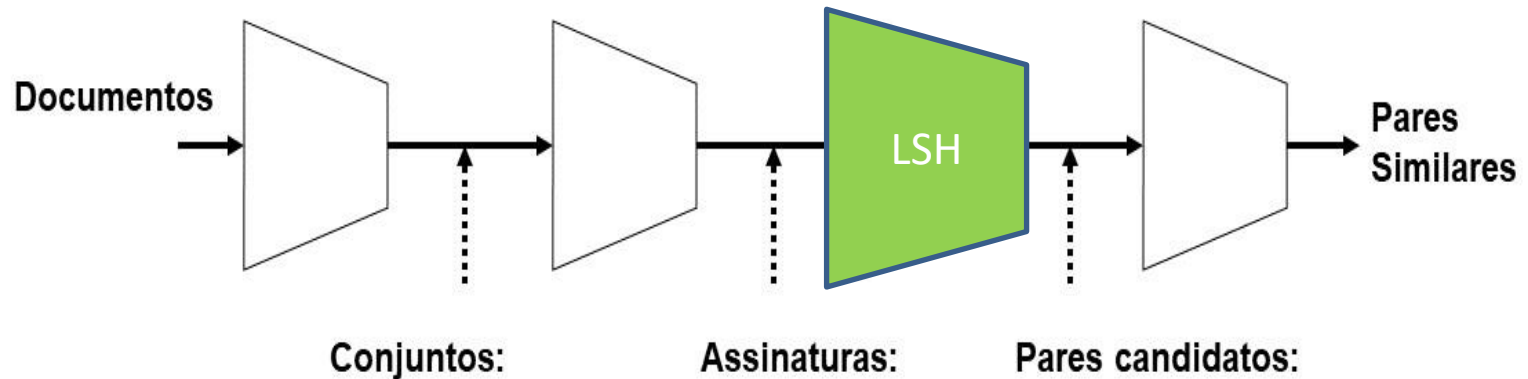
# Múltiplas funções de dispersão

- Definindo a variável aleatória  $X$  como um quando  $h_{min}(C_1) = h_{min}(C_2)$  e como zero caso contrário, então  $X$  é um **estimador não enviesado** (unbiased) da similaridade de Jaccard
- Por assumir apenas os valores zero ou um,  $X$  tem uma **variância muito elevada** para ser útil na prática
- Como reduzir a variância?
- **Utilizando várias variáveis** contruídas da mesma forma
- A **estimativa para a distância é igual à fração de funções de dispersão que concordam**

# Aplicação exemplo

- Não está convencido?
- Apliquemos ao exemplo das avaliações de filmes (MovieLens)
  - Estimando a similaridade usando o mínimo dos hash codes

# 3 - Determinação de pares candidatos



# Pares candidatos

- Mesmo com a redução drástica obtida com as assinaturas, ter de **comparar todos os pares é algo que tem de ser evitado**
- Precisamos de **reduzir o número de pares a comparar**
- O nosso objetivo é encontrar documentos com similaridade, de Jaccard, superior a um determinado limiar
  - por exemplo,  $s = 0.8$  (ou distância inferior a 0.20).

# Candidatos Segundo o MinHash

- Selecionar um limiar de semelhança  $s$  ( $0 < s < 1$ )
- As colunas  $x$  e  $y$  da matriz de assinaturas são um **par candidato** se as suas assinaturas concordarem em pelo menos uma fração  $s$  das suas linhas

$M(i, x) = M(i, y)$  para pelo menos a fração  $s$  valores de  $i$

- Espera-se que os documentos  $x$  e  $y$  tenham a mesma similaridade (de Jaccard) que as suas assinatura



# Ideia base do método LSH (*Locality-Sensitive Hashing*)

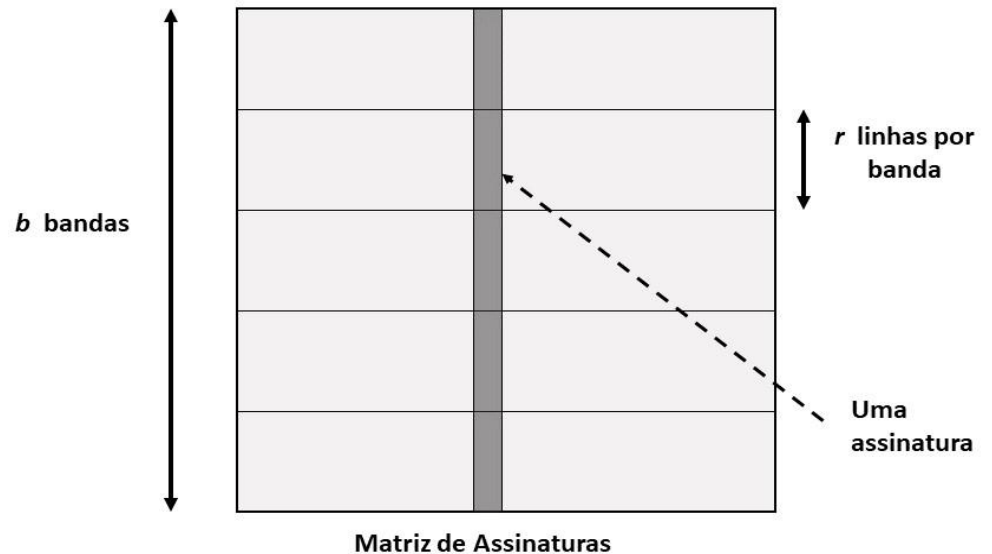
- **Objectivo:** Encontrar documentos com similaridade de Jaccard de pelo menos  $s$ 
  - Para um determinado limiar, por exemplo  $s = 0.8$ .
- **Ideia base:**
  - Utilizar uma função de dispersão  $f(x,y)$  que indica se  $x$  e  $y$  constituem um par candidato
    - Par de elementos cuja similaridade tem de ser avaliada
  - Aplicar a função às colunas da matriz de assinaturas
    - Cada par de colunas que resultam no mesmo valor da função de dispersão é um par candidato

# LSH aplicado à matriz de MinHash

- **Grande ideia:** Aplicar funções de dispersão às colunas da matriz várias vezes
- Fazer com que (apenas) colunas similares tenham elevada probabilidade de terem o mesmo hash code
- Pares candidatos são aqueles que resultam no mesmo hash code

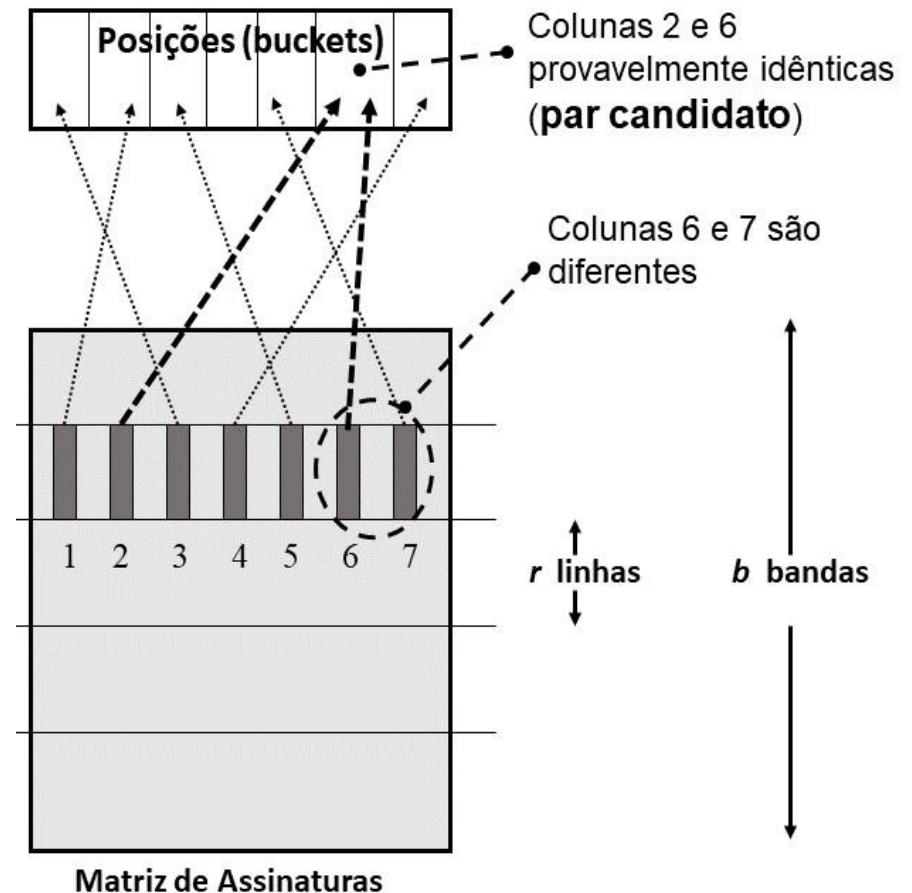
# LSH na prática

- Na prática aplica-se a cada coluna **várias funções de dispersão**
- Divide-se a matriz de assinaturas em  **$b$  bandas**
  - de  $r$  linhas



# LSH na prática (continuação)

- Aplica-se a função de dispersão a cada banda
  - Que mapeia numa de  $k$  posições
    - com  $k$  suficientemente grande
- **Pares candidatos** são mapeados para a **mesma posição** pela função de dispersão para **pelo menos uma das bandas**
  - No nosso exemplo: (2,6)



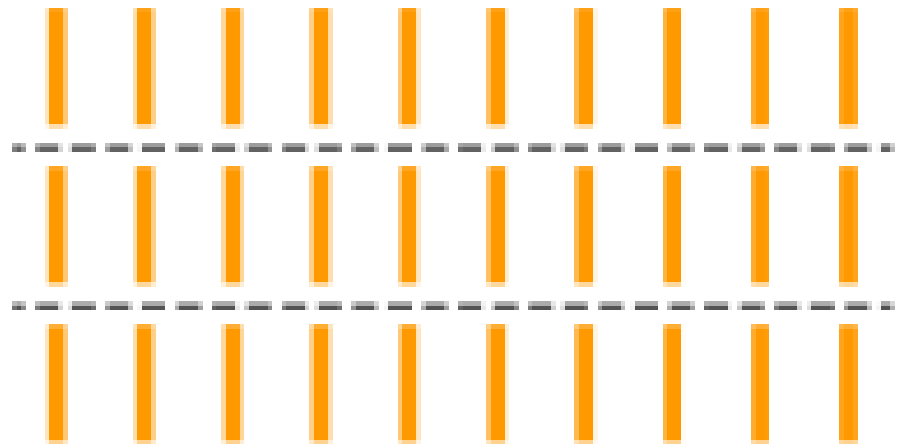
# Análise do LSH

# Análise do processo

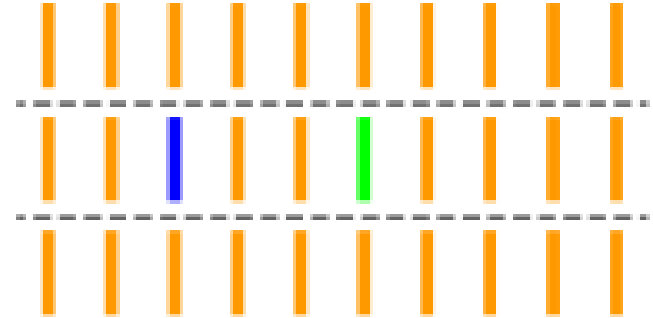
- Assumimos que:
  - existem **posições** (*buckets*) **suficientes** para que as colunas não sejam suscetíveis de mapeamento para a mesma posição a menos que sejam **idênticas** num banda específica
- Em consequência, assumimos que “**mesma posição**” significa “**idêntico nessa banda**”
- Esta simplificação é necessária **apenas para simplificar a análise**, não para a correção do algoritmo

# b bandas, r linhas/banda

- É conveniente representar as bandas de forma abreviada
- A Figura refere-se a 10 documentos (10 colunas) e 3 bandas.



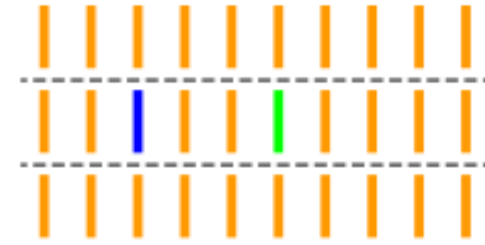
# Análise (continuação)



- Consideremos os dois blocos marcados com coloração diferente na figura
- A probabilidade de todos os elementos do bloco a azul serem iguais aos elementos do bloco verde é  $J^r$ 
  - $J$  é a similaridade de Jaccard dos dois
    - resultante de as colunas  $C_1$  e  $C_2$  terem semelhança  $J$
  - $r$  é o número de linhas de uma banda



# Probabilidades



- A probabilidade de que nem todos os elementos nos dois blocos sejam iguais é

$$1 - J^r$$

- Nem todos serem iguais em  $b$  bandas tem probabilidade

$$(1 - J^r)^b$$

- Probabilidade de pelo menos uma banda idêntica:

$$P = 1 - (1 - J^r)^b$$



# Exemplo de aplicação

- Consideremos o seguinte caso:
- 100 000 documentos ( $\Rightarrow$  100 000 colunas de  $M$ )
- Assinaturas de 100 inteiros (linhas)
  - Ocupam 40 Mb
- Com  $b = 20$  bandas de  $r = 5$  inteiros/banda
- **Objetivo:** encontrar pares de documentos que apresentem  $s \geq 0.8$

# Caso 1: $\text{sim}(C_1, C_2) = 0.8$

- $b = 20$  bandas de  $r = 5$  inteiros/banda.
- Vejamos primeiro o que acontecerá a dois documentos com similaridade elevada (0.8)
- Como  $\text{sim}(C_1, C_2) \geq s$  pretendemos que  $C_1, C_2$  seja um par candidato
  - Pretendemos que haja pelo menos uma banda idêntica
- Probabilidade de  $C_1, C_2$  idênticas numa banda específica:
$$J^r = (0.8)^5 = 0.329$$
- Probabilidade  $C_1, C_2$  não serem semelhantes em todas as 20 bandas:
$$(1 - 0.328)^{20} = 0.00035$$
  - Cerca de 1/3000 dos pares de colunas semelhantes a 80% são falsos negativos
- Encontraríamos 99.97% de pares de documentos verdadeiramente semelhantes

# Caso 1: $\text{sim}(C_1, C_2) = 0.8$

- $b = 20$  bandas de  $r = 5$  inteiros/banda.

- Considerando agora documentos com baixa similaridade

$$\text{sim}(C_1, C_2) = 0.3$$

- Pretendemos que todas as bandas sejam diferentes

- Probabilidade de  $C_1, C_2$  idênticas numa banda:

$$J^r = (0.3)^5 = 0.00243$$

- Probabilidade de termos  $C_1$  e  $C_2$  idênticos em pelo menos uma das 20 bandas:

$$1 - (1 - 0.00243)^{20} = 0.0474$$

- Aproximadamente 4.74% pares de documentos com similaridade de 0.3% acabam como pares candidatos, aparecendo como falsos positivos,
  - que terão de ser analisados mas não são de fato pares similares.

# Otimização do processo

- A aplicação do LSH obriga a um **compromisso entre os falsos positivos e falsos negativos**
- Este compromisso pode ser efetuado através de:
  1. número de funções de dispersão
    - que resultam no número de linhas da matriz de assinaturas;
  2. número de bandas
  3. número de linhas por banda

# Exemplo - efeito de número de bandas

- Quais as alterações na probabilidade de falsos positivos e falsos negativos se utilizarmos apenas 15 bandas de 5 linhas, considerando os casos de  $\text{sim}(C_1, C_2) = 0.8$  e  $\text{sim}(C_1, C_2) = 0.3$  ?

- Falsos positivos

- $P[C_1, C_2 \text{ idênticos numa faixa}]: (0.3)^5 = 0.00243$ .
- $P[C_1, C_2 \text{ idênticos em pelo menos 1 das 15 bandas}]:$

$$1 - (1 - 0.00243)^{15} = 0.0358$$

- Aprox. 3.6% dos pares de documentos com similaridade 0.3 falsos positivos

Diminuiu

(4.74%  
para b=20)

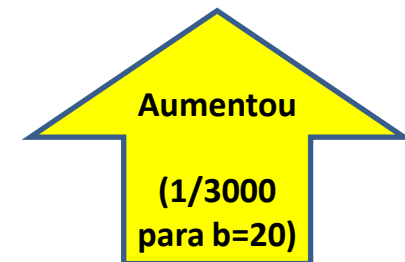
# Exemplo - efeito de número de bandas

- Falsos negativos

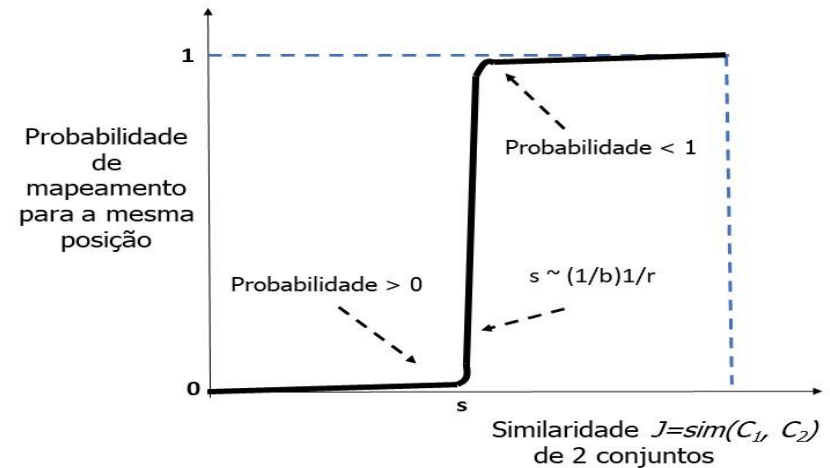
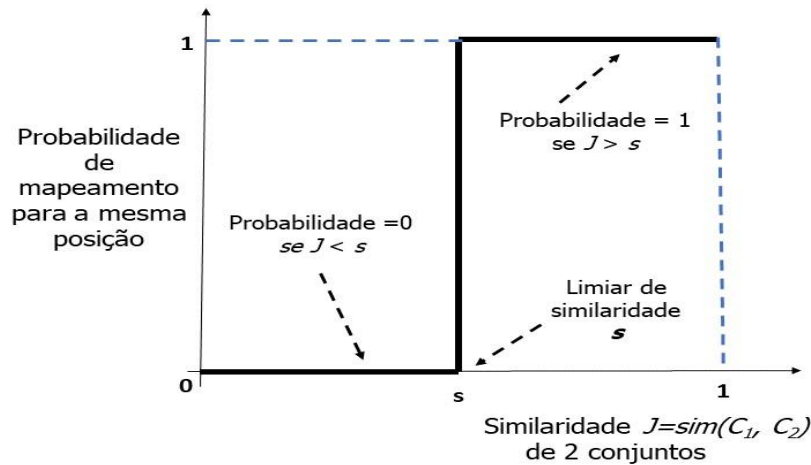
- $P[C_1, C_2 \text{ idênticos numa faixa específica}]: (0.8)^5 = 0.328.$

- $P[C_1, C_2 \text{ não semelhantes em todas as 15 bandas}]: (1 - 0.328)^{15} = 0.0026,$

- cerca de 1/400 dos pares semelhantes a 80% são falsos negativos



# Realidade versus situação ideal



- O que gostaríamos de ter
- Probabilidade nula para similaridades até ao limiar e probabilidade igual a 1 para pares com similaridade superior a esse limiar

- Desempenho real do processo LSH
- Temos probabilidades não nulas para valores de similaridade baixos
- Não temos valores iguais a 1 para similaridades elevadas



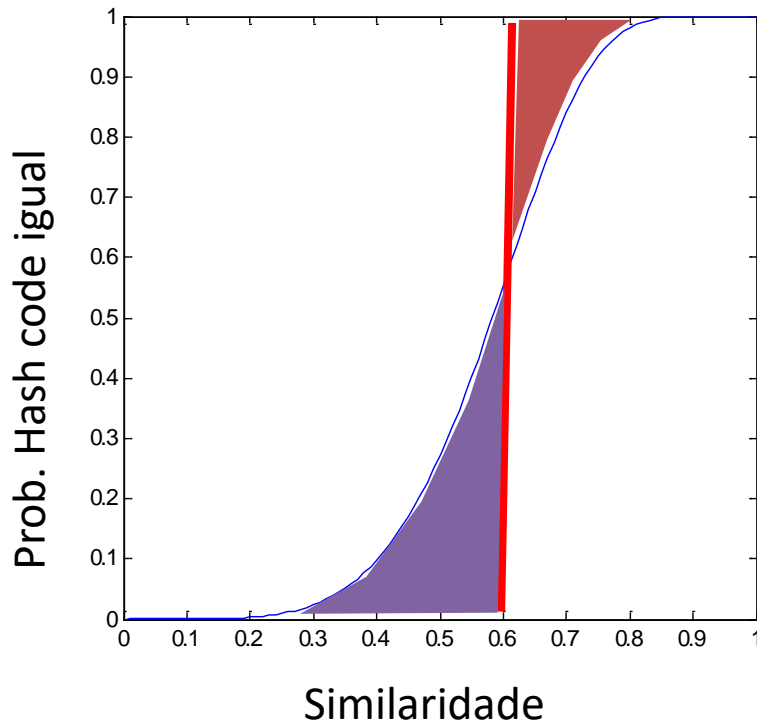
# Exemplo

- $b = 20$  e  $r = 5$
- Prob. de pelo menos uma banda idêntica

$s$	$1-(1-s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

# Valores para $r$ e $b$

- Seleccionar  $r$  e  $b$  para ter a melhor curva
  - Ex: 50 funções de dispersão ( $r=5$ ,  $b=10$ )



Área azul: Falsos Negativos

Área vermelha: Falsos Positivos

# Obtenção dos parâmetros $b$ e $r$

- $b$  e  $r$  podem ser obtidos através da resolução de um conjunto de equações
  - definido em termos de verdadeiros e falsos positivos considerados adequados à aplicação
    - e/ou verdadeiros e falsos negativos

# Exemplo de obtenção de b e r

- Pretendemos:
  1. Falsos positivos < 1% para todos os elementos com similaridade de Jaccard  $\leq 0.6$
  2. Verdadeiros positivos >99% para elementos com similaridade  $\geq 0.9$

# Exemplo de obtenção de b e r

- Sistema de equações:

$$\begin{cases} 1 - (1 - 0.6^r)^b = 0.01 \\ 1 - (1 - 0.9^r)^b = 0.99 \end{cases}$$

- Como resolver?

# Resolução do sistema de equações

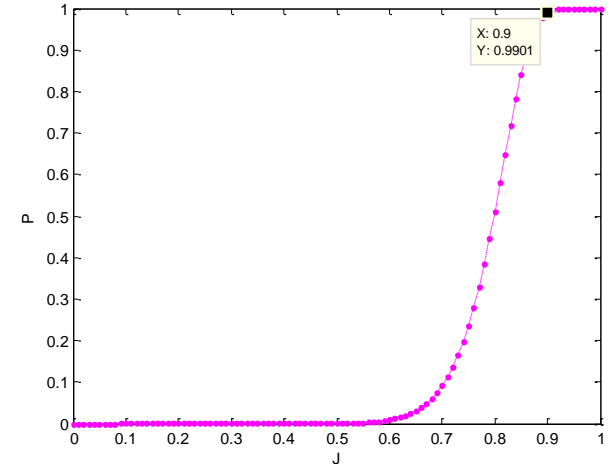
O sistema pode, por exemplo, ser resolvido em **Matlab usando a Optimization Toolbox**

```
fun = @probabilidade;  
  
x0 = [randi(10),randi(10)];  
x = fsolve(fun,x0);  
  
r=fix(round(x(1)))  
b=fix(round(x(2)))  
  
%% -----  
function F = probabilidade(x)  
  
r=x(1);  
b=x(2);  
  
F(1)= 1-(1-0.6^r)^b -0.01;  
F(2)= 1-(1-0.9^r)^b - 0.99;
```

# Exemplo (continuação)

- Confirmemos os resultados...

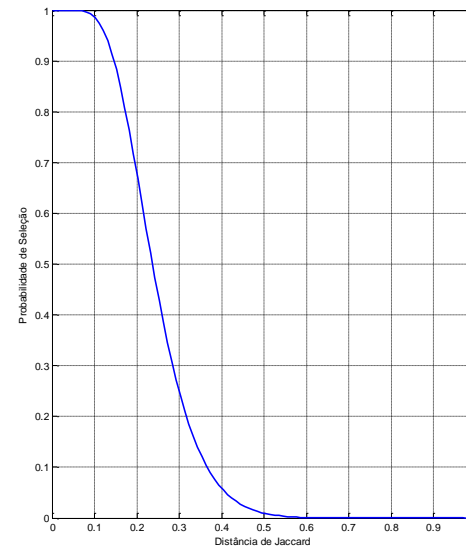
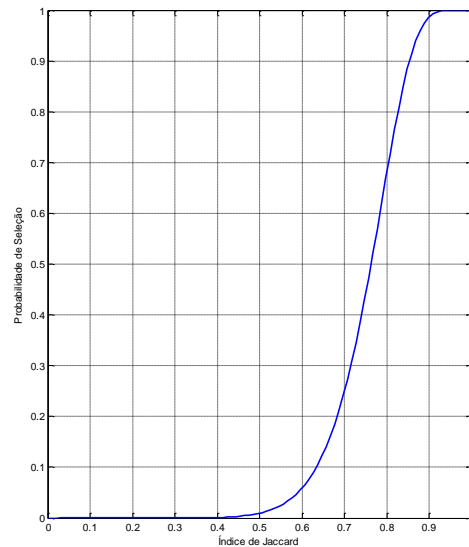
- Curva  $P(J) = 1 - (1 - J^r)^b$ 
  - $r=15$
  - $b=20$



- Probabilidade  $< 0.01$  para similaridade de Jaccard  $\leq 0.6$
- $1 - (1 - 0.6^{15})^{20} \approx 0.0095 < 0.01$ 
  - OK
- Probabilidade  $> 0.99$  para similaridade de Jaccard  $\geq 0.9$
- $1 - (1 - 0.9^{15})^{20} \approx 0.9901 > 0.99$ 
  - OK

# Aplicação ao MovieLens

- Processar a matriz Min-Hash
  - Já calculada
- Usemos, por exemplo:
  - $r=10$     $b=\text{NumHashFunctions} / r$





# Comentários finais

- Afinar  $M$ ,  $b$ ,  $r$  para obter quase todos os pares com assinaturas similares
  - Mas eliminando a maior parte dos pares que não têm assinaturas similares
- Testar se os pares candidatos têm, de facto, assinaturas similares
- Opcional: Num outro passo, verificar se os pares candidatos remanescentes representam mesmo documentos similares

# Recomendação

- Para informação adicional sobre este tópico recomenda-se a consulta de *Mining of Massive Datasets*, da autoria de Jure Leskovec, Anand Rajaraman e Jeff Ullman

**Note to other teachers and users of these slides:** We would be delighted if you found this our material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. If you make use of a significant portion of these slides in your own lecture, please include this message, or a link to our web site: <http://www.mmds.org>

# Parte dos slides adaptados de: Finding Similar Items: Locality Sensitive Hashing

Mining of Massive Datasets

Jure Leskovec, Anand Rajaraman, Jeff Ullman Stanford  
University

<http://www.mmds.org>

