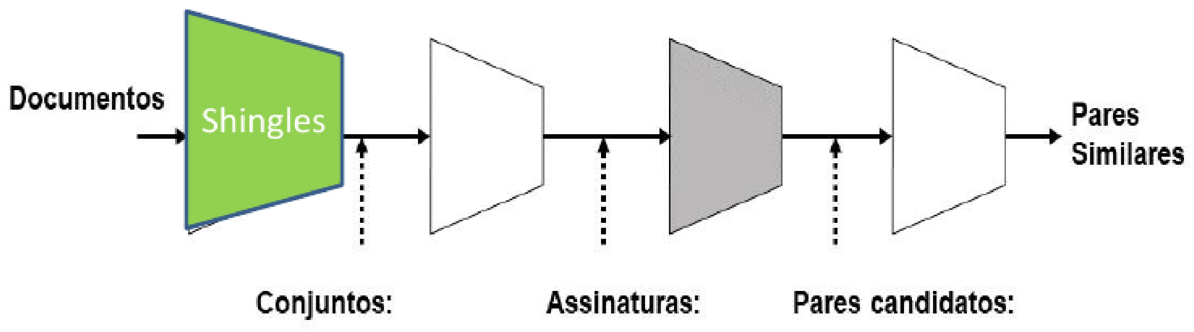
1 - Conversão dos documentos em conjuntos



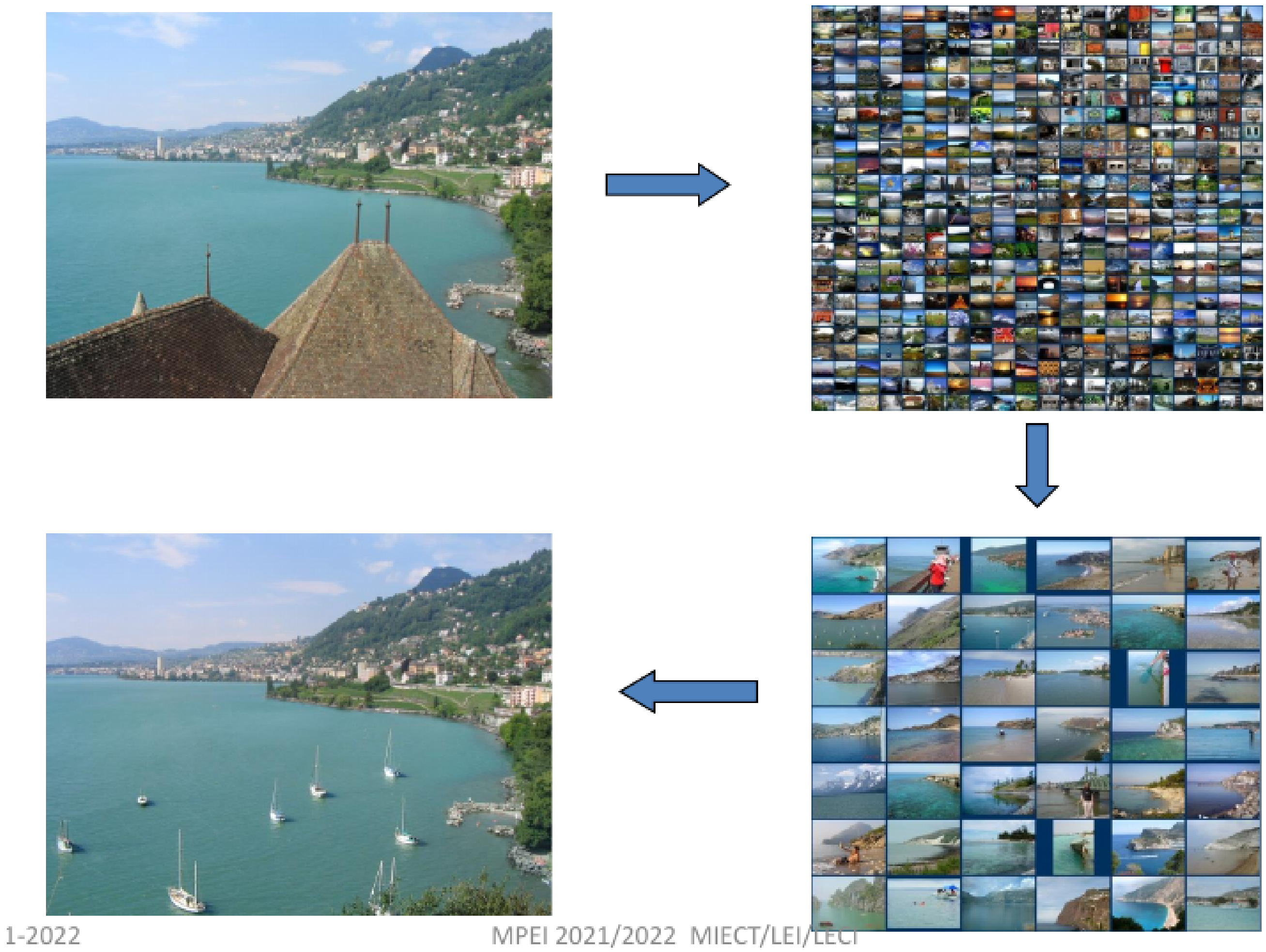
> Solução probabilística para a procura de similaridades

Encontrar similaridades é útil para estudar um grupo de interesses ou comportamentos assim como se interligam além disso tbm pode ser útil para ensinar inteligências artificiais a reconhecer padrões.

Exemplo de aplicação:

Preenchimento de uma imagem

Temos uma imagem onde não conseguimos ver parcialmente a paisagem pois está escondida atrás de um edifício, tendo um conjunto de imagens poderia-mos preencher esse lugar com outro lugar representativo da cena .



Para isso queremos as 10 imagens mais parecidas de um conjunto de ao todo 20 000 imagens

Generalizando

Além dos mencionados muitos problemas **podem ser expressos em termos de descobertas de conjuntos similares**

Em todos estes problemas temos entidades que podem ser representadas por um **conjunto**

Exemplo:

As páginas web podem ser representadas pelo conjunto das palavras que contêm

Definição do Problema

Tendo:

- pontos **x1,x2, ...** num espaço com **n** dimensões

- uma função de **distancia d(x1,x2)**
- **Objetivo**: Determinar todos os pares de dados (xi,xj) com distância igual ou inferior a um determinado limiar s, d(xi,xj) ≤ s

Solução ingénua

- Comparar todos os pares possiveis
- Cm N pontos teríamos complexidade O(N^2)
- Muito demoreda ou mesmo impossível em tempo útil para N grande

Distância

O objectivo é determinar os vizinhos mais próximos no espaço n-dimensional

Portanto os **near neighbours**

Antes de começarmos a calcular distancias temos que defenir o que é a 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:

$$sim(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$

A distância Jaccard, **dJ**, é obtida diretamente da similaridade:

$$d_j(C_1, C_2) = 1 - sim(C_1, C_2)$$

Exemplo em matlab

Calcular a similaridade de 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))
```

Ver a similaridade entre dois textos

```
Sets{1}=getSetOfWordsFromFile('texto1.txt')
Sets{2}=getSetOfWordsFromFile('texto2.txt')
% ...
% Calcular a distância de Jaccard para todos os pares
distJ=calcDistancesJ(Sets);
% Determinar os pares que têm distância inferior a um certo limiar
Similar=findSimilar(distJ,threshold,ids);
```

Problema deste algoritmo

- Muito lento

Conjuntos Grandes e Gigantes

Dado um grande número de documentos (N), queremos determinar pares “quase iguais” para N = milhões, milhares de milhões, biliões, ...

Problemas

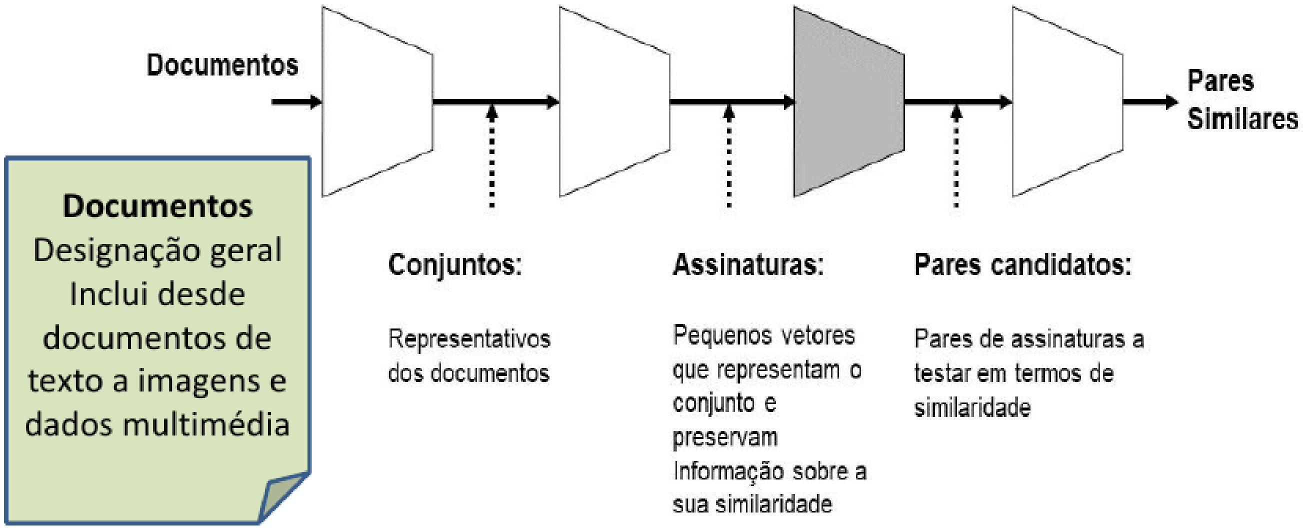
- Demasiados documentos para se compararem todos os pares
- Muitas partes de um documento podem aparecer por outra ordem noutro
- Documentos são tão grandes ou número elevado que não cabem em memória

Solução

1. Reduzir a dimensão dos conjuntos mantendo a informação essencial
2. Reduzir o tempo de cálculo da distancia ou reduzir os pares

Abordagem probabilistica

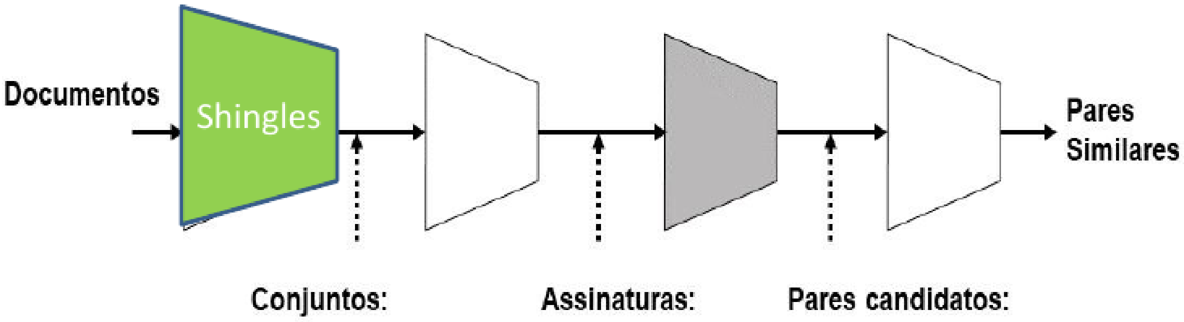
Processo de determinação de documentos similares:



Processo

1. Obtenção dos conjuntos representativos
2. Redução desses conjuntos a conjuntos de dimensão fixa e pequena - **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



O objetivo desta primeira etapa é criar os conjuntos representativos dos documentos, sem perda de generalidade considerando os documentos por aquilo que são compostos 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

Simples

1. Conjunto de palavras que ocorrem num documento
2. Conjunto de palavras "importantes"

No entanto sofrem do mesmo problema não preservam informação sobre a ordem de ocurrencia

A **ordem** de ocorrência pode ser tida em conta utilizando sequências de palavras, ideia base dos **k-gramas**

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.

Assume-se que documentos que têm muitos Shingles em comum são semelhantes

Utilizando Shingles, um documento **D** é representado pelo conjunto dos seus **k-shingles** C = S(D)

Exemplo

Quais os Shingles do documento contendo a sequência de caracteres **‘abcab’** considerando **k=2**

S(D) = {ab,bc,ca}

ou se aceitarmos repetições

S’(D)={ab,bc,ca,ab}

Similaridade para Shingles

Representando um documento **Di** pelo seu **conjunto de k-shingles Ci=S(Di)**

Uma medida de similaridade natural é , o processo que vimos anteriormente **similaridade de Jacard**

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

Escolha de k

A escolha de **k** pode por vezes ser complicada o **k** têm de ser sufciente grande para evitar que a maioria dos documentos tenha a maioria dos Shingles

Na prática

- k = 5 é bom para documentos curtos
- k = 10 é mais adequado para documentos longos

Representação binária

Para simplificar cálculo de intersecção e união, os documentos podem ser representado por um vetor de zeros e uns no espaço de k- gramas (vetor binário)

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 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:

$$
\begin{array}{cc}
\begin{array}{c} \text{aa} \\ \text{ab} \\ \text{bc} \\ \text{cd} \\ \text{da} \end{array} &
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \\
 & \begin{array}{cccc} \text{D1} & \text{D2} & \text{D3} & \text{D4} \end{array}
\end{array}
$$

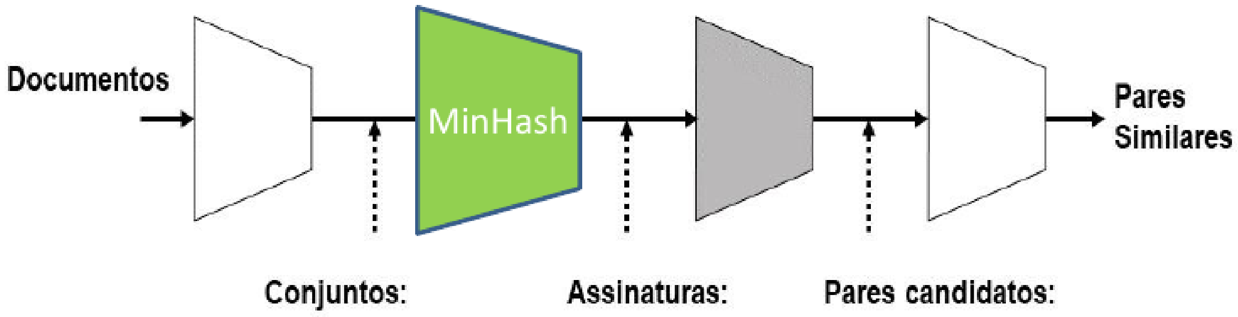
$$d(D_2, D_3) = ?$$

$$
\begin{aligned}
D_2 &= 00110 \\
D_3 &= 00011
\end{aligned}
$$

$$
\begin{aligned}
|C_2 \cap C_3| &= 00010 = 1 \\
|C_2 \cup C_3| &= 00111 = 3
\end{aligned}
$$

- Sim Jaccard = 1/3
- d(C2,C3) = 1 – simJ = 2/3

2 - Cálculo das Assinaturas



Nesta etapa procedemos á redução da representação dos conjuntos

Procedemos a este passo mapeado cada conjunto **Ci** para uma assinatura **Sig(Ci)** 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(C1)** e **Sig(C2)** é aproximadamente igual à **sim(C1,C2)**

Desafio

Obter uma hash function tal que h():

- Se sim(C1, C2) é elevada, então com elevada probabilidade h(C1) = h(C2)

- Se sim(C1, C2) é baixa, então h(C1) ≠ h(C2) 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 π às linhas da matriz booleana

- Reter o valor do índice da primeira linha (na ordem permutada) correspondente a um Shingle (ou equivalente) existente

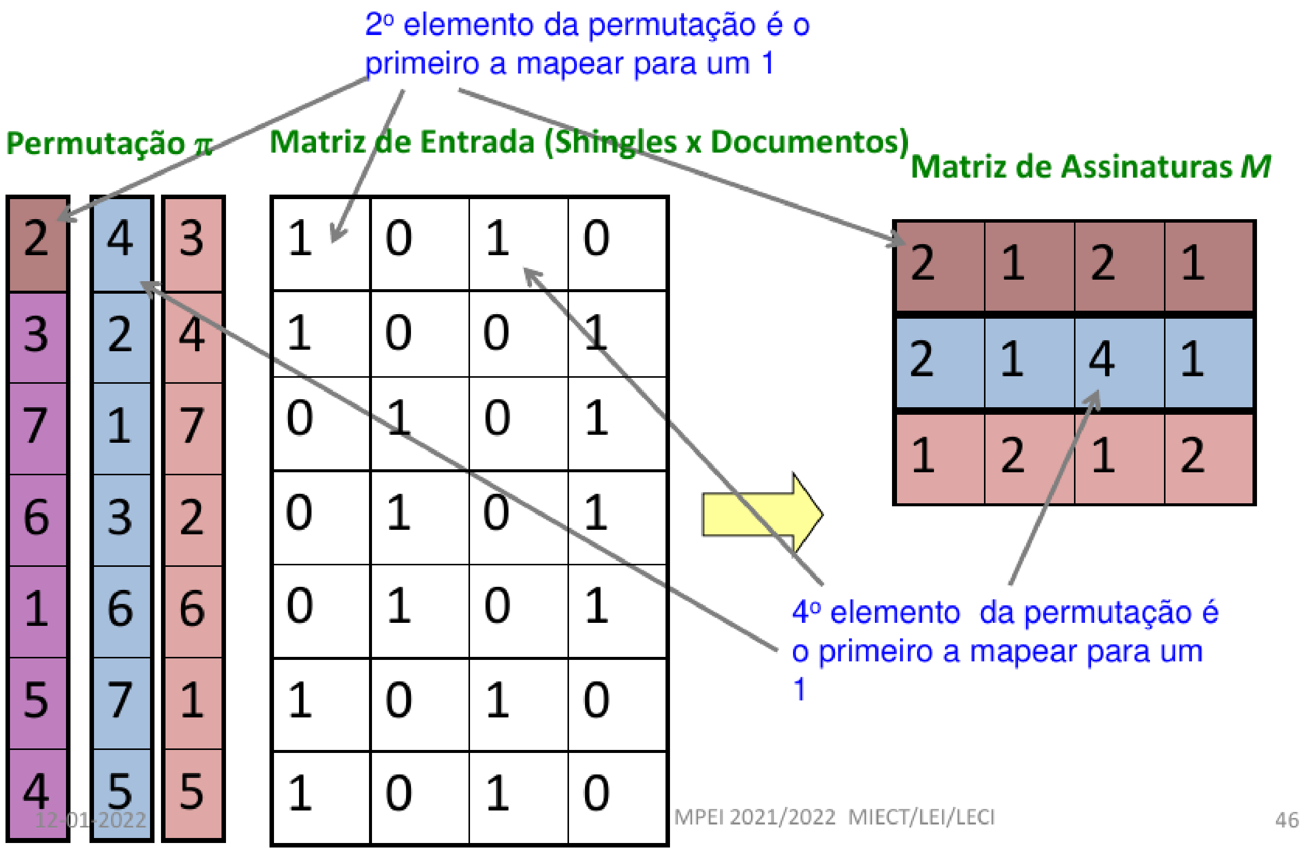
Minimo da função

Esta operação pode ser vista como a aplicação de uma **hash function h_pi(C) = inidice da primeira linha onde C tem valor 1**

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

Repetir para várias permutações independentes, por exemplo 100, para obter um vetor (assinatura)

Exemplo



- Portanto temo 4 documentos
- E 3 permutações que nos vão gerar as 3 linhas da matriz de Assinaturas M

Calcular uma linha

1. Dispomos a matriz de entrada segundo a permutação

1º linha' = 5ª linha

2º linha' = 1ª linha

3º linha' = 2º linha

4º linha' = 7º linha

....

2. Vemos o menor índice onde uma shingle é um e adicionamos á linha

   Neste caso ficaria 2,1,2,1

Por cada permutação calculamos uma linha

Propriedade de h_π

- A função hπ() permite obter assinaturas pois estas mantêm informação sobre a similaridade dos documentos devido à seguinte propriedade:

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

- Para duas colunas **A** e **B** com as shingles relativas a 2 documentos interceptarem-se no shingle de menor indice

$$h(A) = h(B)$$

Em suma para calcular as assinaturas

- Seleccione k permutações aleatórias das linhas
- Pensa na assinatura sig(C) como um vetor
- sig(C)[i] = índice da primeira linha da coluna C que contém 1 de quando aplicada a permutação i a essa linha
  - sig(C)[i] = min (πi(C))

Na prática

Permutar linhas para conjuntos de dimensão elevada é demorado e em geral proibitivo, o que podemos fazer é uma função com base nos nas permutações determine os índices e calcule o mínimo

Uma solução eficiente é utilizar uma hash function

- evitando ao máximo colisões
- o mapeamento efectuado 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 ?