Projeto MPEI - PL4

Pedro Pinto nº115304 ; Jorge Domingues nº113278

---

1. Introdução

No contexto da unidade curricular de Métodos Probabilísticos para Engenharia Informática, este projeto conduziu ao desenvolvimento de uma pequena aplicação onde colocámos em prática conhecimentos adquiridos ao longo do semestre, nomeadamente, funções de dispersão, Counting Bloom Filter, assinaturas e similaridade combinando Shingles e MinHash.

2. Leitura e processamento - readData.m

Este script tem como objetivo ler todos os ficheiros descarregados do Moodle e, a partir deles, gerar as matrizes de assinatura, as matrizes das distâncias, o Counting Bloom Filter e algumas variáveis importantes que são salvas em ficheiros com a extensão *.mat*, armazenados na pasta 'data', para serem utilizados quando necessário no script *main.m*, sem que o utilizador da aplicação precise de esperar pelo processamento. Para simplificar a leitura deste relatório, iremos introduzir três conceitos importantes em que nos iremos basear neste projeto.

Shingles: Um k-shingle para um conjunto de caracteres é uma sequência de k símbolos que aparecem no conjunto. Assume-se que conjuntos que têm muitos Shingles em comum são semelhantes. Neste projeto iremos considerar conjuntos de Shingles sem repetição.

Assinaturas e MinHash: Assinaturas é uma etapa importante para a redução da representação dos conjuntos. Neste projeto, iremos nos referir a matrizes de assinaturas como matrizes $[k, C_i]$, onde as k linhas representam as k funções de dispersão a serem aplicadas aos conjuntos C_i. A aplicação das funções de dispersão baseia-se em MinHash, sendo apenas considerado para a assinatura o menor valor de todos os valores de hash resultantes da k_i função de dispersão, baseando-nos na propriedade $P[h_{min}(C_1) = h_{min}(C_2)] = Sim_J(C_1, C_2)$. Para além disso, se o cálculo das assinaturas for bem feito, a assinatura, $Sig(C_i)$, poderá ser utilizada para a aproximação à similaridade de Jaccard $Sim_J(C_1, C_2) \approx Sim_J(Sig(C_1), Sig(C_2))$. E assim recorremos à aproximação $P[h_{min}(C_1) = h_{min}(C_2)] \approx Sim_J(Sig(C_1), Sig(C_2))$ para o cálculo das similaridades de Jaccard neste projeto.

Counting Bloom Filter: Com semelhanças com o *Bloom Filter*, o *Counting Bloom Filter* é um processo que começa por se criar um vetor com N posições à qual nós chamamos inicialização. De seguida, são aplicadas k funções de dispersão a cada elemento do conjunto, incrementando os resultados nas posições correspondentes do vetor. O que difere entre os dois métodos é a capacidade de contagem, ou seja, ao invés de simplesmente colocar a posição do elemento a 1, incrementamos o valor nas posições do vetor, $B[h_i(elemento) = B[h_i(elemento) + 1]$. Neste método, após adicionar todos os elementos, somos obrigados a guardar todas as funções de dispersão utilizadas, para poderem ser utilizadas também na função de contagem para obtermos a correspondência correta.

Tipos de funções de dispersão: Para este trabalho, utilizámos as funções de dispersão propostas por *Carter* e *Wegman*, que são um tipo de funções de dispersão universal para lidar com números inteiros. Elas consistem em escolher um número primo $p \geq M$ definir $h_{ab}(x) = ((ax + b)\ mod\ p)\ mod\ M$, a e b são inteiros aleatórios módulo p ($a \neq 0$). No caso de strings e shingles, utilizamos a função "*DJB31MA*", e a chave introduzida, do tipo string, vai variando para obter novos valores de hash para a mesma string inicial, garantindo sempre que as funções não estão correlacionadas. Simulando assim k funções de dispersão.

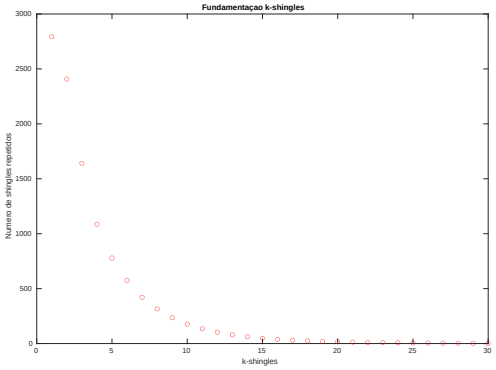
3. Fundamentação das opções tomadas - métodos probabilísticos

k-shingles, opção 3: Analisando a figura 1a, que considera o número médio de shingles repetidos nas descrições consideradas, e tendo em consideração o espaço em memória, concluímos que qualquer valor $k \in \{8, 9, 10, 11, 12, ...\}$ seria uma boa opção. Por isso, para este projeto optámos por considerar $k = 10$ para os k-shingles utilizados na opção 3.

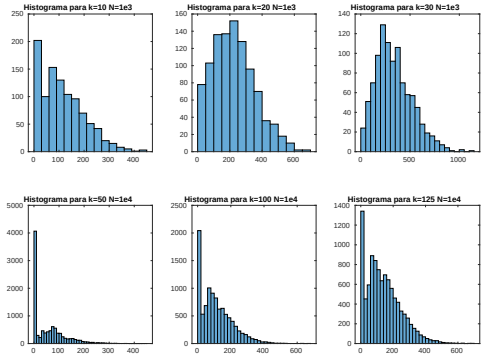
k Funções de dispersão e $\mathbb{Z}_N$, opções 2,3,4: Analisando para cada caso, figuras como a 1c e analisando também a diferença entre as distâncias de Jaccard aproximadas e as distâncias reais com valores exatos, concluímos que para este projeto seria indicado utilizar os seguintes valores, k funções de dispersão, e $H_c \in \mathbb{Z}_N = \{0, 1, 2, ..., N - 1\}$ para a gama de valores de hash possíveis:

- **Opção 2:** k=150, N=10^6
- **Opção 3:** k=200, N=10^8 (demora bastante tempo para processar, são conjuntos grandes para um k grande)
- **Opção 4:** k=150, N=10^6

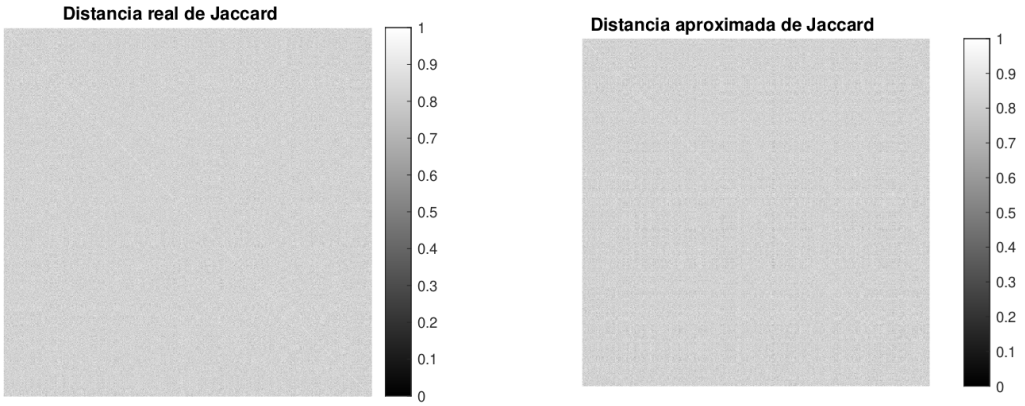
Parâmetros Counter Bloom Filter, opção 5: Analisando a figura 1b, que apresenta vários histogramas variando o tamanho do Bloom Filter e o número de funções de hash, concluímos que uma boa escolha seria $k = 125$ funções de dispersão para um Bloom Filter de tamanho $N = 10000$.



(a) Repetições variando os k-shingles



(b) k f_h para um Bloom Filter de tamanho N



(c) k=150 e N=10^6 (opção 4) comparação real/aproximada

4. Interface do utilizador - main.m

Este script foi concebido com o propósito de apresentar uma interface ao utilizador, solicitando a inserção de um identificador ID correspondente a um turista pertencente ao *travels9.data*. Com este ID, o utilizador terá acesso a um menu composto por seis opções distintas, cada uma representando a aplicação de algoritmos diferentes. Este script utiliza como suporte os ficheiros armazenados na diretoria 'data' desenvolvidos no *readData.m* , que são lidos logo no início do script.

4.1. Opção 1

Na opção 1, o ID inserido pelo utilizador será usado como referência para acessar a estrutura *travelSets*. Esta estrutura contém informações sobre os países visitados por cada turista. Depois é apresentado pela aplicação a quantidade de países avaliados, assim como, a lista desses países, indicando o *ID* e o respetivo *nome*.

4.2. Opção 2

Na opção 2, com a matriz das distâncias, *MdistOption2*, calculada previamente através das respetivas assinaturas dos conjuntos de países visitados por cada turista (conjunto de ID's), *travelSets*, obtivemos uma estimativa aproximada das similaridades de Jaccard com o turista atual. Depois identificamos os dois turistas que têm a maior similaridade e fazémos uma correspondência com a estrutura *travelNames* para recuperar os nomes associados a esses turistas e apresentar essa informação, pois *travelNames* tem a informação de todos os turistas, mesmo aqueles que não visitaram nenhum país. Por fim, calculámos a união dos conjuntos de países pertencentes aos dois turistas mais similares e apresentamos esses países indicando o *ID* e o respetivo nome.

4.3. Opção 3

Na opção 3, utilizando a estrutura *travelSetsMoreThan3Days*, previamente calculada, conseguimos identificar os países nos quais o turista especificado visitou mais de 3 dias. Posteriormente, utilizando a matriz de distâncias, *MdistOption3*, calculada previamente através das respetivas assinaturas derivadas dos k-shingles de cada descrição, retirámos para cada país no *tavelSetsMoreThan3Days* o país mais similar e a respetiva distância. Ao mesmo tempo, verificámos se esses países correspondiam a países já visitados, caso acontecesse era ignorado. Com essas informações, e considerando apenas os países com distâncias inferiores à média de todas as distâncias obtidas apresentamos ao utilizador o ID e os respetivos nomes, assim como a respetiva distância.

4.4. Opção 4

Na escolha 4, usámos uma matriz de distâncias, *MdistOption4*, calculada previamente através das respetivas assinaturas derivadas dos interesses de cada turista, para conseguirmos encontrar o turista com maior similaridade de interesses. Com estes dados, e fazendo a correspondência já referida anteriormente com *travelNames* e *travelInterests*, apresentamos ao utilizador o nome, os respetivos interesses e a similaridade com o turista atual.

4.5. Opção 5

Na Opção 5, com base na estrutura *BloomFilterContagem* previamente calculada, a qual armazena em formato de Filtro de Bloom a aproximação das contagens de visitas a cada país, procedemos à extração dessas contagens através da função *contagemElemento*. Esta função utiliza as mesmas funções de hash utilizadas na criação do *Bloom Filter*, assegurando assim que os valores de hash são idênticos. Isso permite verificar, a partir do menor valor correspondente, a aproximação da respetiva contagem.

4.6. Opção 6

A opção 6 é destinada a encerrar o programa.

5. Conclusão

Em suma, o trabalho realizado está de acordo aos requisitos estabelecidos no enunciado. O script "*readData.m*" foi criado de forma consistente permitindo a adequada preservação dos dados essenciais para a resolução da interface do utilizador. Da mesma forma, o script "*main.m*" encontra-se bem organizado e implementado, capaz de atender a todas as opções desejadas pelo utilizador. Também criámos funções auxiliares para ajudar na visualização do trabalho e também nos permitir uma melhor organização.

É importante destacar algumas dificuldades que tivemos ao longo do desenvolvimento do trabalho como por exemplo a escolha do número de funções de dispersão a usar numa determinada função assim como o seu tipo pois tivemos de experimentar várias diferentes para chegar à que consideramos 'melhor'. Por outro lado, estes desafios ajudaram-nos a entender melhor os conceitos deste módulo e saber o que fazer em determinadas situações diferentes para conseguirmos chegar ao melhor resultado possível.