
# Trabalho Prático

Métodos Probabilísticos para Engenharia Informática

## Implementação de algoritmos probabilísticos


| | |
|---|---|
| *Manuel Costa* | *67849* |
| *Henrique Manso* | *65308* |

Professor Carlos Bastos

# *Bloom filter*

Um *bloom filter* é uma estrutura de dados probabilística, sob a forma de vetor, que permite testar se um elemento pertence a um conjunto de dados usando hashing. Não tem precisão 100% pois por vezes retorna falsos positivos, mas nunca retorna falsos negativos tornado a sua utilização bastante viável.

**Implementação:**

- Utilizou-se a função de *hashing* universal em que os três parametros inteiros a,b e c são calculados utilizando a libraria Random do Java.
- *Fudge factor* é sempre 1000.
- O *array* é um objeto do tipo BitSet (array de bits).
- Uma *hash function* é apenas um objeto da classe “parameters” que contém os parametros para a dita *hash function* facilitando a manipulação de várias *hash functions*.

**Resultados:**

- Para um *set* de 2766 palavras , testámos outras 1714 obtendo entre 8-15 falsos positivos.

# *Shingling*

O processo de shingling permite criar subconjuntos de *Strings* de um determinado documento. Podem-se agrupar n a n palavras ou caracteres para formar os subconjuntos.

**Implementação:**

- O módulo *Shingles* recebe um conjunto de documentos. Cada documento é representado por uma *ArrayList* de *Strings*.
- O módulo tem duas opções, gerar os *Shingles* palavra a palavra ou carater a carater. Esta opção é explicada na secção “Uso” do módulo neste relatório e no construtor do Módulo “*MainMinHash*”.
- Os *Shingles* são gerados N a N palavras ou N a N carateres.
- No fim o módulo produz uma *ArrayList* de Strings (conjunto de *shingles*) para todos os documentos.

# *Min-hash*

Min-hash é uma técnica que permite estimar o grau de similaridade entre conjuntos de forma relativamente rápida. Isto deve-se ao facto de converter pares de Documento/*Shingles* numa matriz de inteiros para mais fácil e rápida estimação da distância de Jaccard entre vários Documentos.

**Implementação:**

- O módulo opera sobre a matriz de Documento/*Shingles* gerada pelo módulo "*Shingles*" . Para todos os *Shingles* do set de *Shingles* são calculados N *hash codes* (N sendo o número de *hash functions*) mantendo sempre o valor mínimo. Por exemplo: Documento0 = "O bom dia" tem os *Shingles* {"O bom","bom dia"}. Seriam então calculados h1("O Bom), h1("bom dia").De todos esses valores, o mínimo entra na matriz de signaturas na posição [0][0] (Documento 0 ,Função 0). Após este processo estar terminado para todas as *hash functions* teremos a coluna de signaturas do Documento0. Repete-se o processo para os outros documentos até obtermos a matriz signatura em que cada linha representa uma *hash function*, cada coluna um documento e vada valor é o valor mínimo de *hash* dos *shingles* todos do documento para aquela *hash function*.
- A função de *hash* utilizada foi a Universal, como no *Bloom Filter*.
- No fim o módulo terá calculada a matriz signatura.

# *Local-sensitive hashing*

Se o número de documentos for relativamente pequeno, o cálculo da similaridade é fácil de se calcular a partir da matriz de signaturas. Com 5 documentos e 50 *hash functions*, seriam feitas 500 comparações. Bastava comparar os documentos um a um através das suas colunas de signaturas. Para um processador moderno isto não é um desafio. O desafio aparece quando temos milhares de documentos e centenas de *hash functions*. É neste contexto que aparece o módulo "*LSH"*. A matriz de signaturas é divida em bandas de n linhas por banda. Cada banda é mapeada numa *HashTable*. Se duas bandas de documentos diferentes são mapeadas para a mesma posição significa que houve uma colisão e temos documentos candidatos a serem analisados. Assim sendo, diminui-se o número de documentos a serem

comparados logo podemos analisar milhares de documentos de forma eficiente.

**Implementação:**

- O módulo opera sobre a matriz de signaturas proveniente do módulo "*MinHash*".
- As bandas são mapeadas para uma *HashTable*. Se vários documentos distintos forem mapeados para a mesma *key* na *table*, são agrupados dois a dois. Cada par de documentos é um objeto do tipo *Pair*.
- São usadas o mesmo número de *hashfunctions* que no módulo *MinHash* e as função são Universal.
- A *HashTable* tem como tamanho (número de *buckets*) *numBuckets* que é decidido no módulo "*MainMinHash*" pelo utilizador.
- O número de bandas é o número de documentos.
- O número de linhas por banda é cedicido no módulo "*MainMinHash*" pelo utilizador.

**Resultados:**

- Para testes feitos sobre poucos documentos reais de dimensões pequenas com ligeiras modificações, os resultados foram 100% acertados.
- Para testes feitos sobre sensivelmente 1300 documentos em que 1000 são únicos e 300 são cópias idênticas, os resultados foram 100% acertados.
- Para tetes feitos sobre poucos documentos com grandes dimensões (livros) os resultados foram parcialmente corretos. Para livros idênticos, os resultados foram os que tínhamos previsto assim como para livros parecidos. Já para livros distintos, os resultados parecem estar errados. Não sabemos contudo a origem desta falha.

# *Módulos de Suporte*

Como referido noutros módulos, foram criados módulos para dar suporte a:

- *Hash functions* – Módulo "*parameters*"
- Pares de documentos – Módulo "*Pair*"
- Readers:
    - TxTReader – Leitor de ficheiros .txt. Transforma ficheiros .txt numa lista de palavras
    - CSVReader – Leitor de ficheiros .csv. Transforma uma determinada coluna dum ficheiro .csv numa lista de palavras. Não utilizado.
    - rFiles – Gerador de ficheiros .txt com conteúdo aleatório (10000 carateres).
    - DictReader – Leitor do diccionário para o teste do bloom filter. Usado apenas uma vez.

# *Módulos Main*

---

Instruções de uso

---

Os módulos "*Main*" têm instruções de uso e todas as funções se encontram devidamente comentadas.
Todos os ficheiros .txt para serem testados devem ser colocados no workspace sobre as pastas:

- /docs/Apresentacao/BloomFilter para o bloom filter.

O nome dos ficheiros deve ser especificado e os campos para os módulos devem ser especificados da seguinte forma:

**MainMinHash:**

```
/*
 * USAGE:   colocar em pathToFiles a pasta onde os ficheiros .txt estão alocados
 *          colocar em agrupar o número que queremos agrupar os shingles em
 *          colocar em numRowsPerBand a quantidade de linhas por banda
 *          colocar em factor o fator que queremos para calcular o número de buckets (factor * numDocs)
 *          colocar em minCollisions o número mínimo de colisões para calcular a distância de Jaccard
 *          colocar em numhashfuncs o número de hash functions a utilizar
 *          colocar em test_books,test_aula,test_big o tipo de teste a fazer
 *          colocar em group_by_words caso os shingles sejam para agrupar em words em vez de chars
 *          DEBUG : printMatrix para mostrar o estado da matriz de signaturas nos diferentes módulos
 */
```

Se o pathToFiles não for especificado, deve-se assinalar uma das flags test_books,test_aula ou test_big. Cada uma destas irá procurar pelos ficheiros nas pastas Books, AulaTest e BigTest respetivamente. Se nada for assinalado nem existir um path, será lançada uma exceção.

**mainBloomFilter:**

```
/*
 * Usage:   colocar em k o número de hash functions a serem utilizadas
 *          colocar em size o tamanho do bloom filter
 *          colocar na pasta src/docs/Apresentacao/BloomFilter os ficheiros
 *              de membros e de elementos a testar.
 *          colocar nos campos MemberFileName e TestFileName os nomes
 *              dos ficheiros de Membros e de Teste respetivamente
 *
 */
```

# *Comentários finais*

Foram visualizados os seguintes vídeos para melhor entendimento dos conteúdos:

- 3 1 Finding Similar Sets 13 37
- 3 2 Minhashing 25 18
- 3 3 Locality Sensitive Hashing 19 24

As notícias reias apresentadas foram retiradas do seguinte link:

- Notícias

Os outros documentos de teste foram retirados das páginas:

- Wikipedia - Ásia
- Wikipedia - The_Binding_of_Isaac

Os livros foram retirados das páginas:

- TxT Books - TextFiles

<FIM>