

Assignment 2

Recuperação de Informação

Realizado por:

- Borys Chystov            nº mec. 78198

- Isaac dos Anjos          nº mec. 78191

**ÍNDICE**

[INTRODUÇÃO 2](#__RefHeading___Toc503_1444915559)

[1. CONCEITOS TEÓRICOS 4](#__RefHeading___Toc505_1444915559)

[1.1 INDEXAÇÃO 4](#__RefHeading___Toc507_1444915559)

[1.2 ÍNDICE INVERTIDO 4](#__RefHeading___Toc509_1444915559)

[1.3 STOP WORDS 5](#__RefHeading___Toc511_1444915559)

[2. DESENVOLVIMENTO DO PROJETO 6](#__RefHeading___Toc513_1444915559)

[2.1 TOKENIZADOR SIMPLES 6](#__RefHeading___Toc515_1444915559)

[2.2 TOKENIZADOR MELHORADO 6](#__RefHeading___Toc517_1444915559)

[2.3 INDEXER 7](#__RefHeading___Toc519_1444915559)

[2.4 SINGLE-PASS IN-MEMORY INDEXING (SPIMI) 7](#__RefHeading___Toc521_1444915559)

[2.5 WEIGHTED TERM 8](#__RefHeading___Toc523_1444915559)

[2.6 INVERSE DOCUMENT FREQUENCY (IDF) 8](#__RefHeading___Toc527_1444915559)

[2.7 GARBAGE COLLECTOR 9](#__RefHeading___Toc529_1444915559)

[3. CONCLUSÃO 10](#__RefHeading___Toc531_1444915559)

[REFERÊNCIAS BIBLIOGRÁFICAS 11](#__RefHeading___Toc533_1444915559)

APÊNDICE I - Diagrama de classes

APÊNDICE II - *WorkFlow*

APÊNDICE III - Respostas às questões do *Assignement* 1

# INTRODUÇÃO

O presente relatório enquadra-se no âmbito da componente prática da unidade curricular de Recuperação da Informação, inserida no 1º semestre do 1º ano de Mestrado em Engenharia Informática, da Universidade de Aveiro, no ano letivo 2019/2020, sob a orientação do docente Sérgio Matos. Neste contexto, foi-nos proposta a realização de um projeto, com uma simples indexação de um documento que consiste num processador, *tokenizador* e um *indexer*.

A realização deste relatório torna-se pertinente na aquisição de conhecimentos do funcionamento de motores de busca como por exemplo, o google. Desta forma, a componente prática deste trabalho irá permitir obter bases sobre os vários conceitos que motores de busca utilizam no processo de indexação.

Deste modo, a realização deste relatório constitui, simultaneamente um momento de aprendizagem e um documento de avaliação, tendo como principais objetivos: adquirir conhecimentos relativos a vários processos realizados por indexadores e adquirir conhecimento teórico subjacente ao funcionamento dos mesmos.

Relativamente à estrutura, o relatório inicia com o capítulo – “Conceitos teóricos”, onde será realizada uma breve introdução teórica acerca dos conceitos “Indexação”, “Índice invertido” e “*Stop Words*”, para que, posteriormente, no capítulo seguinte – “Desenvolvimento do Projeto” seja efetuada a explicação de todo o desenvolvimento do projeto prático. Este capítulo encontra-se dividido em vários subcapítulos “Tokenizador simples”; “Tokenizador melhorado”; “Indexer”; ”Single-pass in-memory indexing”; “Weighted Term” e “Inverse Document Frequency”, para uma melhor compreensão de todo o trabalho desenvolvido.

# CONCEITOS TEÓRICOS

## 

## 1.1 INDEXAÇÃO

A indexação analisa e armazena dados para facilitar a recuperação rápida e precisa de informações. Apenas é necessário introduzir o nome do arquivo ou algum termo a ele relacionado no motor de busca e, de seguida, o sistema apresentará resultados segmentados e otimizados.

O objetivo central de um índice é otimizar a velocidade e o desempenho na procura de documentos relevantes para uma pesquisa. Sem este, o mecanismo de pesquisa ria verificar todos os documentos disponíveis no seu contexto, o que exigiria muito tempo e grande poder computacional.

## 

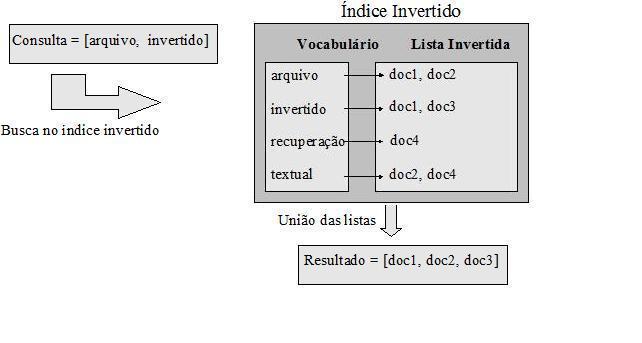
## 1.2 ÍNDICE INVERTIDO

O índice invertido consiste numa lista de palavras e nos respetivos documentos nas quais elas emergem. Neste sentido, é possível localizar rapidamente documentos que contenham as palavras de uma determinada pesquisa. Por outras palavras, o índice invertido é uma estrutura de dados de índice que armazena um mapeamento do conteúdo para os seus respetivos locais, num documento ou conjunto de documentos.

Contudo, para que exista, de facto, um verdadeiro ganho de tempo no processo de recuperação e indexação, é necessário que o índice invertido seja construído previamente ao processo que envolve a consulta do utilizador.

Assim, o índice invertido caracteriza-se por um processo de pré-processamento, sendo constituído por duas componentes principais: uma estrutura de procura, designada por “vocabulário”, que contém todos os termos distintos existentes nos textos indexados e, para cada termo, uma lista invertida, que armazena os identificadores dos registos que contêm o termo, ou seja, os documentos onde a palavra ocorre.

Após escolher a coleção de documentos a ser indexada, é necessário que cada documento tenha um identificador único na coleção, sendo que, a cada documento encontrado é atribuído um *DocID*. Os dados inseridos no processo de indexação são uma lista de termos contidos no documento.



## 1.3 STOP WORDS

As *stop words* são utilizadas com frequência nos mecanismos de pesquisa para reduzir o número de termos armazenados no índice invertido.

A maioria dos mecanismos de pesquisa usa o princípio TF-IDF, que se refere a “*term frequency–inverse document frequency*", ou seja, as palavras que ocorrem com muita frequência tendem a ser menos significativas do que as palavras que ocorrem raramente.

Assim, as *stops words* são importantes, uma vez que, quando as removemos de um documento, facilita a compreensão do significado do mesmo e o contexto que lhe está subjacente.

# DESENVOLVIMENTO DO PROJETO

Na fase inicial do projeto procedeu-se ao carregamento do ficheiro pretendido através de um *input* *stream* para que o mesmo pudesse ser lido linha a linha. O ficheiro contém *tags* que indicam o seu início “***PMID***” e o seu fim “***PG***”. Para além destas existe a *tag* “***TI***” que corresponde ao título do documento.

Na fase de processamento do documento é criada uma instância da classe do mesmo, tendo este um identificador único para que seja possível distinguir e facilmente identificá-lo, com o intuito de guardar os parâmetros ***TI*** e o ***PMID***. É de salientar que houve um cuidado especial ao processar o título, uma vez que este poderá encontrar-se em múltiplas linhas.

De seguida, é possível escolher o *tokenizador* que será utilizado no processamento dos documentos, através de um parâmetro de entrada. Poderá ser escolhido um *tokenizador* simples ou um *tokenizador* melhorado.

## 

## 2.1 TOKENIZADOR SIMPLES

O objetivo do tokenizador simples é o de substituir todos os carateres que não pertencem ao alfabeto por um espaço em branco e transformar todas as palavras para minúsculas. Após este processo, é feito um *split* para retirar palavras com menos de três carateres. As palavras duplicadas são retiradas e é adicionada uma contagem com o intuito de contar as repetições do *token*.

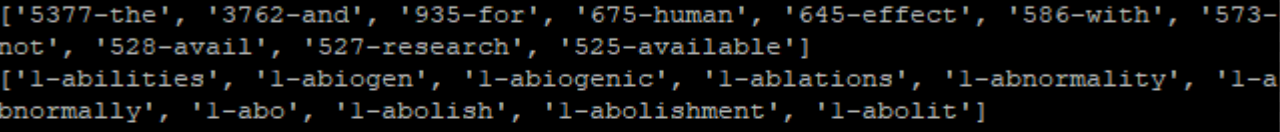
## 2.2 TOKENIZADOR MELHORADO

O segundo tokenizador também tem um comportamento similar ao tokenizador simples, uma vez que este substitui todos os carateres que não pertencem ao alfabeto por um espaço em branco. De seguida percorre todos os *tokens* e compara cada um com o ficheiro para remover as *stop word.* Por último, antes de a palavra ser adicionada ao dicionário é feito o *steeming* dessa palavra.

## 2.3 INDEXER

Antes de proceder à leitura dos ficheiros, é instanciado um objeto da classe i*ndexer*para que, no fim do processamento de cada documento, seja adicionado ao dicionário do *indexer* os *tokens*, *document frequency* e o ID do documento em que os *tokens* ocorrem.

Todo o processo anteriormente mencionado é repetido para todos os documentos do ficheiro. Posteriormente é chamada a função “*orderTokens*” da classe *indexer para ordenar* numericamente e alfabeticamente os *tokens.* Em seguida encontra-se uma imagem do resultado da indexação e da ordenação do ficheiro.

****

## 2.4 SINGLE-PASS IN-MEMORY INDEXING (SPIMI)

Ao invés de escrever os resultados da indexação para um único ficheiro, utilizou-se o algoritmo SPIMI para gravar o resultado num ficheiro diferente quando passa de um certo *threshold* de memória. Á medida que o algoritmo vai sendo processado, vai gravando os termos e as respetivas *posting-list* para um dicionário. Quando a memória atinge um certo limite definido pelo user, o dicionário é gravado para um ficheiro e, em seguida, é criado um dicionário que é utilizado para guardar os termos que ainda estão por indexar. Assim, com este método, os termos são armazenados em múltiplos ficheiros libertando a memória.

Após a fase de indexação, obtém-se inúmeros blocos, cada um com um dicionário. Desta forma, foi implementado um algoritmo de MERGE para unir todos os blocos, contendo o termo e o respetivo *posting-list*.

De seguida será explicado o algoritmo MERGE.

1. É criado um objeto *reader* para cada bloco e, de seguida, é guardado dentro de um *array* dinâmico;
2. É lida a 1ª linha de cada bloco e, seguidamente é guardada num dicionário juntamente com o correspondente índice;
3. De cada uma das linhas é extraído o termo e o índice do bloco (do qual a linha foi extraída), sendo guardados no mesmo dicionário;
4. É ordenando o dicionário e realizado *skip* dos índices, por forma a obter o termo alfabeticamente menor;
5. Iterar sobre as *posting-list* do termo alfabeticamente menor (obtido no ponto anterior) e concatenar os *postings* para uma variável denominada por *output*;
6. O *output* é ordenado pelo *docId* e de seguida é gravado num ficheiro, sendo realizado o *pop* do termo alfabeticamente menor do dicionário e, de seguida, é feito um *readline* do bloco que contém o termo;
7. Caso o termo seguinte exista no dicionário, o índice do bloco é concatenado ao *value* do termo e o processo repete-se (a partir do ponto 4);
8. Caso o termo seguinte não conste no dicionário, o termo é adicionado ao mesmo e o processo repete-se (a partir do ponto 4).

## 2.5 WEIGHTED TERM

Quando um documento é lido, é guardado o termo, docID e um *array* com as posições em que o termo se encontra. Com base nessa informação, é possível extrair o WT de cada documento com a seguinte fórmula:

WT: (log10(termFrequency) + 1) \* IDF

Resultado obtido após o cálculo do WT para cada documento:

Termo;docID:WT:position1, position2, position3;docID:WT: position22, position33

## 2.6 INVERSE DOCUMENT FREQUENCY (IDF)

O cálculo do IDF é realizado em simultâneo com o WT e é utilizada a seguinte fórmula:

log10(numberTotalDocuments/documentFrequency)

Resultado obtido após o cálculo do WT para cada documento:

Termo:IDF;docID:WT:position1, position2, position3;docID:WT: position22, position33

## 2.7 GARBAGE COLLECTOR

No processo de indexação deparámo-nos com um problema de memória ao processar o algoritmo *SPIMI*. Para tentar contornar o problema e libertar a memória, optamos por limpar o dicionário que guardava os termos, sempre que é atingido uma percentagem de utilização de memória. Contudo essa solução não foi muito eficiente devido ao facto da memória continuar ocupada. Utilizou-se a ferramenta do *Gargage Collector* para forçar a libertação da mesma e interrompeu-se o ciclo para dar a oportunidade de iniciar uma *thread* com o processo do *Garbage Collector.* Deste modo a memória é libertada e é resumido o processo de indexação.

# CONCLUSÃO

Através da realização deste trabalho foi possível compreender e desenvolver alguns dos conceitos lecionados nas aulas teórico-práticas, uma vez que foi possível aplicá-los no desenvolvimento deste projeto.

Durante a primeira fase de desenvolvimento do projeto deparámo-nos com dificuldades na componente teórica acerca da indexação dos *tokens*. Com o decorrer das aulas e, através de estudo autónomo, o conhecimento sobre a matéria foi aprofundado, o que levou a superar essas mesmas dificuldades. Após compreender os conceitos de “indexação”; “índice invertido”; “*stop words*”; “*tokenizador*” e “*indexer*” e os requisitos pretendidos para o projeto, a componente prática tornou-se acessível.

Numa segunda fase foi possível compreender melhor o algoritmo *Single-pass in-memory indexing* (*SPIMI*), que deu origem à implementação do *merge* dos ficheiros. Também foi possível aplicar as fórmulas de *Weighted Term* e *Inverse Document Frequency* que serão cruciais para o desenvolvimento da terceira e última fase do projeto.

# 

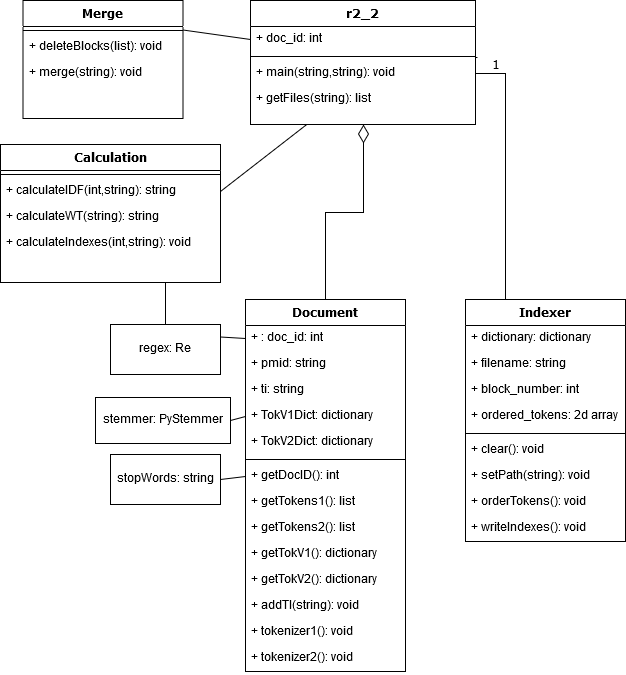
# REFERÊNCIAS BIBLIOGRÁFICAS

MATOS, Tauller *et all* – Índice invertido para recuperação de imagens baseada em conteúdo. FACOM: Faculdade de Ciência da Computação. p.20-25.

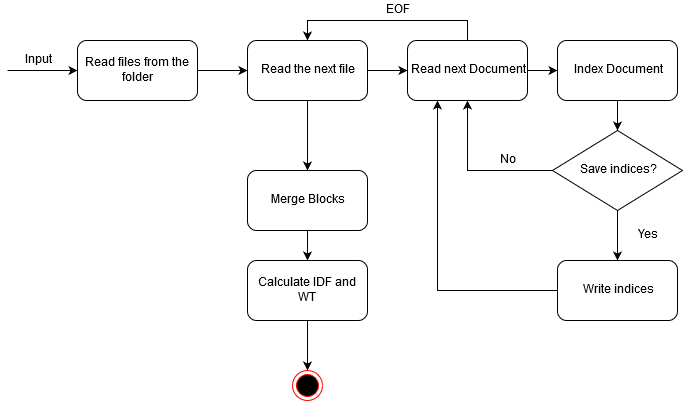
SANTOS, Ismael Francisco Antuens – Indexação de arquivos texto: uma contribuição para o projeto Openged. Lages: Universidade do Planalto Catarinense, 2002. Bacharelato.

**APÊNDICES**

**APÊNDICE I –** Diagrama de classes

****

**APÊNDICE II** – WorkFlow



**APÊNDICE III** – Medidas de eficiência

|  |  |  |
| --- | --- | --- |
|  | Tokenizer 1 | Tokenizer 2 |
| Vocabulary size | 327278 | 264054 |
| Size in disk | 563MB | 470MB |
| Maximum mem used\* | NA | NA |
| Indexing time | 606 seconds | 616 seconds |
| Merging time | 266 seconds | 224 seconds |
| Calculation time | 7461 seconds | 330 seconds |
| Total time | 8030 seconds | 1170 seconds |

\* Optou-se por receber o “*Maximum Memory User*” como parâmetro de entrada, visto que o processo de indexação é demorado. O parâmetro define o limite de memoria o processo pode utilizar em percentagem.