Projeto de LI-3

Martins, José(a78821) Costa, Mariana(a78824)
Quaresma, Miguel(a77049)

June 10, 2017

Contents

1 Introdução

A **Wikipédia** é uma das muitas fontes de informação disponíveis na Web, possuindo uma quantidade bastante considerável de dados/informação. No entanto para que esta informação seja útil é necessário saber processá-la e acima de tudo fazê-lo em tempo útil, sendo para isso necessário recorrer a estruturas de dados que permitam uma pesquisa rápida mas sem uso excessivo de memória. O objetivo deste projeto é, por isso, implementar uma aplicação (em *Java*) que responde a um conjunto de *queries* relativas a *snapshots* da **Wikipédia** em tempo reduzido e de uma forma correta, recorrendo para isso a um conjunto de recursos já existentes e a estruturas de dados que permitam atingir este objetivo.

2 Desenvolvimento

2.1 Parsing dos snapshots

Os dados sobre os quais incidem as *queries* às quais o projeto responde encontram-se no formato XML sendo por isso necessário processá-los de forma adequada, isto é, retirar a informação útil tendo em conta a sintaxe deste tipo de ficheiro. Para processar ficheiros XML existem diversas API's , sendo que este projeto recorre à StAX para conseguir esta funcionalidade. A StAX permite a leitura dos ficheiros on-demand, isto é, ao contrário da DOM que guarda o resultado do parsing do ficheiro num objeto árvore em memória, a StAX lê o ficheiro de forma gradual permitindo reduzir a quantidade de memória utilizada. Este API apresenta no entanto a desvantangem de não permitir o acesso a informação(eventos) que já tenha sido processada logo caso a mesma seja necessária posteriormente à sua leitura é preciso guardá-la.

2.2 Classes

Sentimos a necessidade de criar duas classes principais, sendo que uma representa artigos e outra representa contribuidores. Portanto destas classes podemos construir quantas instancias artigos ou contribuintes quisermos.

2.2.1 Article.java

A classe article, seguindo a linha do trabalho anterior e de modo a responder às queries propostas possui as seguintes variáveis de instancia:

```
private long id; //id do artigo
private String title; //título do artigo
private Map<Long,String> revisions; // revisões do artigo
                                    // chave: id da revisão
                                    // valor: timestamp da revisão
private long nRev; //número de revisões
private long len; //das revisões, o número de caracteres
                  //da revisão com mais caracteres
private long words; //das revisões, o número de palavras
                    //da revisão com mais palavras
```

Nesta classe, para além dos métodos usuais, é de destacar o método `public void setNewLenghtWords(String text)` visto ser este o método que calcula o número de palavras e caractéres de uma String passada como argumento e caso os valores sejam maiores que os existentes nas variáveis de instância `len` e `words` atualiza-os.

2.2.2 Contributor.java

Quanto à classe contributor, como na classe article seguimos as directrizes do trabalho anterior e como tal possui as seguintes variáveis de instancia:

```
private long id; //id do contribuidor
private String name; //nome do contribuidor
private int nRev; // número de revisões do contribuidor
```

Nesta classe, foram apenas implementados os métodos usuais (gets, sets, clone, etc).

2.2.3 Comparators

Para além das classes já referidas estão presentes mais 3 classes de comparadores necessários para implementar algumas queries, sendo esses comparadores os seguintes:

[align=left]: compara dois artigos devolvendo -1 caso o 1º argumento tenha mais caracteres ou caso tenha o mesmo número de caracteres que o 2º argumento, mas tenho um menor id, caso contrário devolve 1, é usado na query `top_20_largest_articles`; : compara dois artigos devolvendo -1 caso o 1º argumento tenha mais palavras ou caso tenha o mesmo número de palavras que o 2º argumento, mas tenho um menor id, caso contrário devolve 1, é usado na query `top_N_articles_with_more_words`; : compara dois contribuidores devolvendo -1 caso o 1º argumento tenha mais revisões ou caso tenha o mesmo número de revisões que o 2º argumento, mas tenho um menor id, caso contrário devolve 1, é usado na query `top_10_contributors`;

2.2.4 QueryEngineImpl.java

Possui um HashMap de artigos e um TreeMap de contribuintes de modo a agregar artigos num conjunto e contribuidores também num conjunto, sobre os quais podemos aplicar métodos. Ainda nessa classe possuímos dois longs de modo a guardar os artigos únicos(artUn) e os artigos totais(artTot). Nesta classe é realizado o processamento dos snapshots bem como é implementado as queries propostas. Quanto ao processamento dos snapshots está dividido em quatro métodos:

[align=left]: percorre a lista de snapshots e chama o método processDoc, apanhando as exceções possiveis; : sempre que encontra uma tag `page` chama o método processPage e vai calculando tanto os artigos totais como unicos; : cria o artigo caso necessario, senão atualiza os valores, chama contudo o método processRevision de modo a ser processado a tag `revision`, devolve se é um novo artigo ou não; : adiciona a revisão ao artigo correspondente bem como caso seja um

novo contribuidor adiciona-lo ao Map de contribuidores, caso contrário adiciona uma revisão ao contribuidor.

3 Conclusão