Nesta aula de laboratório iremos resolver exercícios relacionados com **extracção de informação e processamento de documentos textuais**.

Tenha em atenção que o software usado nas aulas de laboratório oferece, como complemento às funções da biblioteca XPath 2.0, funções de extensão relacionadas com o processamento de elementos textuais. Estas funções foram desenvolvidas com recurso ao LingPipe, uma biblioteca de funções Java para processamento de texto. Para mais informações sobre o LingPipe deverá ser consultado o site http://alias-i.com/lingpipe/.

Entre as funções de extensão disponíveis encontram-se as seguintes:

- Função `gti:textdoc()`, que aceita como parâmetro um URI para um documento e retorna a cadeia de caracteres correspondente ao seu conteúdo textual.

- Função `gti:lang()`, que aceita como parâmetro uma cadeia de caracteres e classifica a mesma de acordo com a língua, devolvendo o código de duas letras correspondente.

- Função `gti:regexchunk()`, que aceita como parâmetros uma cadeia de caracteres e uma expressão regular, retornando todas as entidades mencionadas na cadeia de caracteres que correspondem à expressão regular fornecida.

- Função `gti:dictionarychunk()`, que aceita como parâmetros uma cadeia de caracteres e um dicionário (i.e., uma lista de sequências de cadeias de caracteres com as entidades que se pretendem extrair), retornando todas as entidades mencionadas na cadeia de caracteres que se encontram listadas no dicionário.

- Função `gti:mlchunk()`, que aceita como parâmetros uma cadeia de caracteres e um documento XML com os exemplos de treino. Esta função utiliza um algoritmo de aprendizagem automática baseado em *Modelos de Markov (HMMs)* para reconhecer as entidades mencionadas na cadeia de caracteres, com base nos exemplos fornecidos.

- Função `gti:classify()`, que aceita como parâmetros uma lista de cadeias de caracteres correspondendo aos documentos a classificar, uma lista de cadeias de caracteres correspondendo aos documentos pré-classificados a usar como dados de treino, e uma lista de cadeias de caracteres correspondendo às classes de cada um dos documentos de treino. Esta função utiliza o algoritmo de aprendizagem automática *Naive Bayes* para construir um modelo de classificação com base nos dados de treino.

A Figura 1 exemplifica a utilização destas funções, mostrando como se podem extrair entidades (e.g., endereços de email ou nomes próprios) que se encontrem mencionadas numa cadeia de caracteres, usando dicionários e expressões regulares.

```
declare namespace gti = "java:gti";

let $text := 'O Manuel, casado com a Maria, tem o endereço manuel@mail.pt'
let $lang := gti:lang($text)
let $dict := ('Manuel','Maria' )
let $rexp := '[^ ]+@[^ ]+'

for $i in ( gti:regexchunk($text,$rexp) , gti:dictionarychunk($text,$dict) )
return <entidade lang="{$lang}">{$i}</entidade>
```

Figura 1 – Exemplo da utilização de funções de extensão relacionadas com extracção de informação.

Exercício 1

Apresente um programa XQuery que faça o reconhecimento de todos os nomes de filmes que se encontram mencionados na página Web com URL http://www1.uni-hamburg.de/clpic/tematicos/cinema/panorama.html.

O reconhecimento deverá ser feito com base num dicionário. Para construir o dicionário, deverá extrair todos os filmes que se encontram listados da página Web com URL http://pt.wikipedia.org/wiki/Anexo:Lista_de_filmes_portugueses.

Exercício 2

No URL http://gibson.tagus.ist.utl.pt/~bmartins/train-locais.xml encontra-se um documento XML contendo frases de textos noticiosos onde os nomes de locais mencionados no texto se encontram explicitamente anotados.

Em contrapartida, no URL http://gibson.tagus.ist.utl.pt/~bmartins/teste-locais-raw.xml, encontra-se um documento XML contendo textos noticiosos onde não existem anotações para os locais mencionados.

2.1 - Escreva um programa XQuery que, através do uso da função de extensão `java:gti.mlchunk()`, permita extrair do documento *teste-locais-raw.xml* todos os nomes de locais que se encontram mencionados nos textos. Como dados de treino, utilize o documento *train-locais.xml*, o qual contem um conjunto de frases com os nomes de locais anotados.

2.2 - Escreva um programa XQuery que, através do uso da função de extensão `java:gti.dictionarychunk()`, permita extrair do documento *teste-locais-raw.xml*, todos os nomes de locais que se encontram mencionados no texto. Como dicionário para a extracção, utilize os nomes de locais que se encontram anotados no texto do documento *teste-locais-raw.xml.*

Exercício 3

A página Web da Wikipedia na língua Inglesa que se encontra disponível no URL http://en.wikipedia.org/wiki/List_of_painters_by_name apresenta uma lista com os nomes dos pintores mais influentes ao longo da história.

3.1 – Escreva uma XQuery para extrair da página Web indicada acima os nomes de pintores que nela são mencionados.

3.2 – Escreva uma XQuery que, usando como dicionário os resultados da alínea anterior, permita extrair do documento sobre o movimento "Fauvismo," que se encontra no URL http://www.ibiblio.org/wm/paint/tl/20th/fauvism.html, todos os nomes de pintores que se encontram mencionados no texto.

Exercício 4

No URL https://dspace.ist.utl.pt/bitstream/2295/374583/1/7sectors.zip encontra-se uma colecção de ficheiros HTML representativa de documentos sobre produtos e organizações de sete áreas de actividade diferentes (i.e., materiais, energia, finanças, saúde, tecnologia, transportes e utilitários diversos). A colecção encontra-se ainda subdividida em dois grupos, denominados os grupos de teste e de treino.

4.1 – Escreva uma XQuery que, utilizando os documentos da colecção de treino, aprenda um modelo classificação capaz de atribuir documentos HTML previamente não classificados a uma das sete classes consideradas (i.e., as áreas de actividade).

4.2 – Escreva uma XQuery que, utilizando o modelo de classificação da pergunta 4.1, atribua uma classe a cada um dos documentos do conjunto de teste. Deverá produzir como resultado um documento XML com o formato ilustrado na Figura 2.

```
<docs>
 <doc>
   <titulo>National-Oilwell Downloads</titulo>
   <ficheiro>7sectors-test/energy/1.html</ficheiro>
   <classe>energy</classe>
 </doc>
</docs>
```

Figura 2 – Documento XML com informação sobre documentos HTML e áreas de actividade.

4.3 – Escreva uma XQuery que permita estimar a precisão e a cobertura (i.e., medir *Precision* e *Recall*) do modelo de classificação desenvolvido na pergunta 4.1.

Exercício 5

A função de extensão `java:gti.dictionarychunk()`, descrita nas aulas de laboratório e construída com base nas funcionalidades disponibilizadas pela API do LingPipe, corresponde ao método Java cujo código se apresenta na Figura 3.

```
public static String[] dictionarychunk(String text, String ent) throws Exception {
        MapDictionary<String> dictionary = new MapDictionary<String>();
        for (int i=0; i<ent.length; i++)
             dictionary.addEntry(new DictionaryEntry<String>(ent[i],"chunk",1.0));
        ExactDictionaryChunker dictionaryChunkerTT = new
             ExactDictionaryChunker(dictionary,
                                    IndoEuropeanTokenizerFactory.INSTANCE, true, true);
        Chunking chunking = dictionaryChunkerTT.chunk(text);
        List<String> results = new ArrayList<String>();
        for (Chunk chunk : chunking.chunkSet()) {
            int start = chunk.start();
            int end = chunk.end();
            String type = chunk.type();
            double score = chunk.score();
            String txt = text.substring(start,end);
            results.add(txt);
        }
       return results.toArray(new String[0]);
}
```

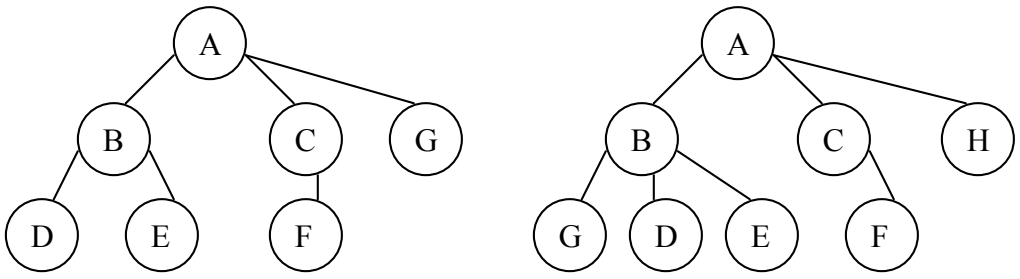
Figura 3 – Método Java correspondente à função java:gti.dictionarychunk().

Crie uma nova função de extensão, denominada *approxdictionarychunk()*, que em lugar de fazer o emparelhamento directo com todas as entidades mencionadas no dicionário, permita reconhecer todas as entidades mencionadas que tenham uma distância de edição inferior a "2" em relação às entidades mencionadas no dicionário. O código Java da função deverá ser apresentado como resolução deste exercício.

Sugestão: Deve consultar a documentação (i.e., os tutoriais e a documentação Javadoc) da API LingPipe, que se encontra disponível através do site http://alias-i.com/lingpipe/. Em particular, deverá ser analisada em detalhe a documentação disponível para a classe **com.aliasi.chunk.ApproxDictionaryChunker**.

Exercício 6 – Problema teórico

Considere as duas árvores apresentadas de seguida:



6.1 - Usando o algoritmo *Simple Tree Matching*, introduzido nas aulas teóricas, calcule a similaridade entre as árvores. Deverá apresentar todos os passos envolvidos no cálculo.

6.2 - Indique um possível alinhamento dos nós das árvores.

Exercício 7 – Problema teórico

Considere o seguinte *Hidden Markov Model*, *M=(A,B,p)*, desenhado para identificar ocorrências de nomes de locais sobre documentos de texto.

A – Probabilidade de transição entre estados

	Prefixo	**Local**	**Sufixo**	**Outras**
Prefixo	0,1	0,75	0	0,15
Local	0,05	0,35	0,30	0,30
Sufixo	0	0,02	0,44	0,54
Outras	0,25	0,10	0,22	0,43

B – Probabilidade, para cada estado, de geração de símbolos (i.e., palavras)

	Prefixo	**Local**	**Sufixo**	**Outras**
A	0,10	0	0,10	0,10
Azeitão	0	0,40	0,02	0,01
De	0,20	0	0,10	0,02
Em	0,40	0,01	0,10	0,04
Maria	0	0,20	0,01	0,03
Nasceu	0,10	0,02	0,10	0,20
Perto	0,10	0	0,10	0,07
Setúbal	0	0,35	0,01	0,03
<outras>	0,01	0,02	0,16	0,45
<pontuação>	0,09	0	0,30	0,05

P – Probabilidade inicial dos estados

Prefixo	**Local**	**Sufixo**	**Outras**
0,35	0,30	0,05	0,30

7.1 - Desenhe uma representação gráfica da máquina de estados correspondente.

7.2 - Calcule a sequência de estados mais provável de ter gerado a frase: “A Maria nasceu em Azeitão, perto de Setúbal.”. Apresente os cálculos efectuados em cada passo.

7.3 - Sugira uma modificação ao modelo que pudesse melhorar o seu desempenho.