Introdução ao Processamento de Linguagens Naturais (4º ano de Curso)

Trabalho Prático 2

Relatório de Desenvolvimento

José André Martins Pereira
(a82880@alunos.uminho.pt)

Ricardo André Gomes Petronilho
(a81744@alunos.uminho.pt)

1 de Dezembro de 2019

Resumo

Na unidade curricular de Introdução ao Processamento de Linguagens Naturais do Mestrado integrado em Engenharia Informática foi-nos proposta a análise e uitlização de uma ferramenta intitulada **lxml**.

Conteúdo

1 Introdução **2**

2 Estrutura da ferramenta lxml **3**

3 Conceção da Solução **4**

3.1 Objetivo do Programa 4

3.2 Funcionalidades 4

3.3 Implementação 5

4 Como instalar e executar o xmlstats **7**

4.1 Requisitos 7

4.2 Como executar 7

5 Conclusão **9**

Capítulo 1

Introdução

Na unidade curricular de Introdução ao Processamento de Linguagens Naturais, do Mestrado integrado de Engenharia Informática, foi-nos proposta a análise da ferramenta **lxml**.
A ferramenta lxml XML toolkit trata-se de uma associação Python às bibliotecas da linguagem C, denominadas **libxml2** e **libxslt**. As grandes características desta ferramenta são a velocidade e a integridade dos recursos XML dessas bibliotecas com a simplicidade de uma API Python.
Assim, este projeto, tem como principais objetivos, o estudo e análise desta ferramenta, bem como a aplicação da mesma para resolver problemas no contexto NLP.
Deste modo, a ideia do grupo para aplicar a ferramenta **lxml**, foi desenvolver uma aplicação intitulada de **xmlstats**, capaz de executar expressões regulares sobre ficheiros **XML** de websites como stackOverflow, entre outros. A finalidade da aplicação destas expressões regulares permite a execução de queries interessantes como:

- o top **N** de utilizadores com mais **score**;
- procurar por palavras, frases, links presentes em **posts**, **comments**;

sendo que, o utilizador do programa é que decide a expressão regular a aplicar.

Capítulo 2

Estrutura da ferramenta lxml

A ferramenta lxml é composta por uma API ElementTree, bastante elegante, com todas as funcionalidades necessárias para ler, processar, e escrever ficheiros XML.

Capítulo 3

Conceção da Solução

3.1 Objetivo do Programa

Tal como já foi referenciado anteriormente, o programa trata-se de um executor de expressões regulares sobre ficheiros **XML**, mais propriamente de websites do **StackExchange**, calculando as ocorrências das mesmsas. Os ficheiros **XML** destes websites são normalmente:

- **Badges.xml**
- **Comments.xml**
- **Posts.xml**
- **Tags.xml**
- **Users.xml**
- **...**

3.2 Funcionalidades

Como já foi dito, a funcionalidade central da aplicação **xmlstats** é execução de uma expressão regular a vários dados **XML**. No entanto existem duas formas distintas de aplicar a mesma.

Uma das formas consiste na aplicação da expressão regular a cada entidade (p.e. **posts**) e calcular as ocorrências dessa expressão por entidade (p.e no post com Id = "91782"a expressão regular ocorreu 128 vezes).
Por exemplo: aplicando a expressão regular da figura abaixo **regIP**, que captura endereços IP e procurando por um top 3 de ocorrências da mesma em **posts - Posts.xml**, a aplicação retorna os três **posts** onde foram publicados mais endereços IP's, sendo o output o verificado na table 4.1 mais abaixo.

```
regIP = r"[ \,\;\:]([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3})[ \,\.\;\:\!\?]"
```

Figura 3.1: Expressão regular que captura endereços IP.

Id do Post	Número de ocorrências
Id=”91782”	128
Id=”164489”	80
Id=”76535”	60

Tabela 3.1: Ocorrências por entidade (neste caso por **Post**)

De outro modo, a aplicação aplica a expressão regular na mesma às entidades, no entanto, as ocorrências são guardadas em relação à expressão regular e não à entidade como na funcionalidade anterior, isto é, o número de vezes que ocorre cada expressão regular em todos os ficheiros. Servindo como exemplo, aplicando a mesma expressão regular apresentada acima, o output desta funcionalidade será o top 3 dos IPs que mais ocorrem nos ficheiros analisados.

Importante realçar que ambas as funcionalidades referidas anteriormente da aplicação, após a determinação do número de ocorrências, o output das mesmas é guardado num ficheiro em formato **XML** utilizando a ferramenta lxml numa diretoria chamada **output**.

Expressão regular	Número de ocorrências
’127.0.0.1’	357
’0.0.0.0’	152
’192.168.0.7’	76

Tabela 3.2: Ocorrências por expressão regular.

3.3 Implementação

Para implementar as funcionalidades descritas acima foram necessárias três funções denominadas:

- from_xml(filename :str)
- to_xml(lst : list, agregation : str, single : str)
- top_reg_by_elem(filenames : list, attributes : list, regex : str, N : int, keyAttr : int)
- top_reg_in_file(filenames : list, attributes : list, regex : str, N : int)

A função **from_xml** tem como objetivo ler um ficheiro **XML** e preencher a estrutura **ElementTree** da ferramenta **lxml** com o seu conteúdo no formato da mesma.

A função **to_xml** tem o comportamento oposto à anterior, ou seja, escreve no ficheiro, com o formato **XML** utilizando a ferramenta **lxml**. Os argumentos que esta recebe são uma **list** com os dados a guardar, o **agregation**, que é uma string com o nome do conjuntos dos dados (p.e. posts) e o uma string denominada **single**, que se trata do nome das entidades dessa agregação (p.e. post), sendo estas variáveis importantes para a criação das **tags** do ficheiro **XML** com **lxml**.

A função **top_reg_by_elem** tem como objetivo aplicar a uma **list** de ficheiros **XML**, ou seja, o argumento **filename**, onde respetivamente a cada ficheiro se atibuí um elemento da list **attributes** que se trata

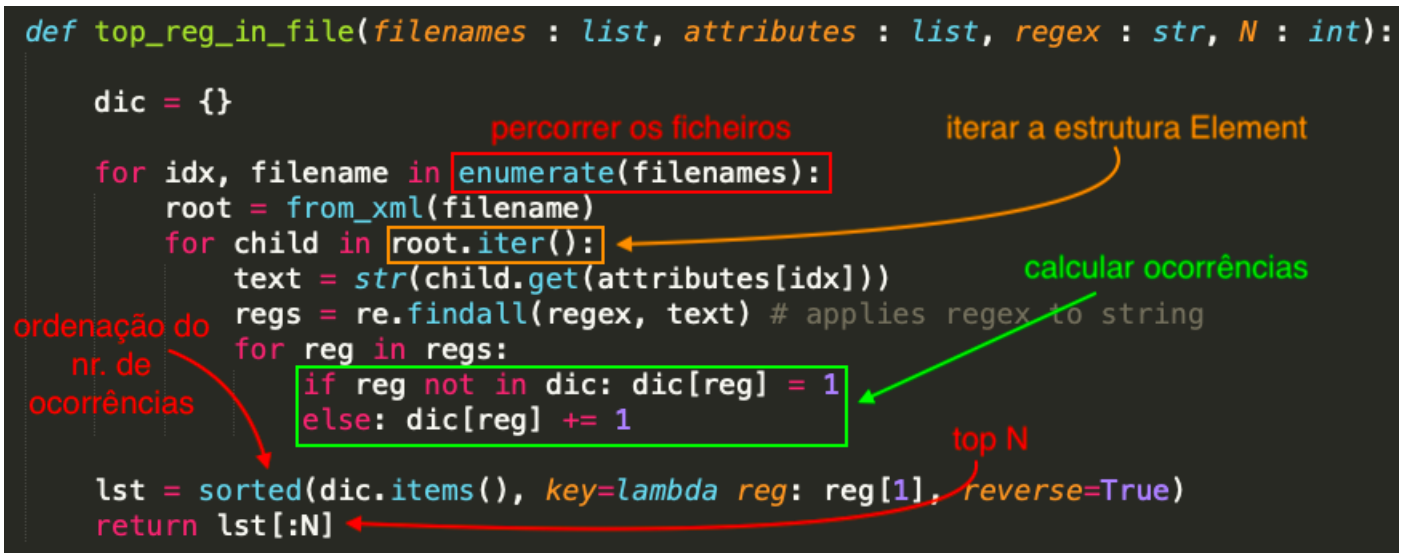

Figura 3.2: Função **top_reg_by_elem** .

do campo onde se vai buscar o texto (p.e. "Body"no ficheiro Posts.xml). A este texto, presente em cada entidade (p.e. **post**) é aplicado a expressão regular dada como argumento **regex**. O número de ocorrências da mesma em cada entidade, é guardado num dicionário onde a chave é o Id da entidade, e o valor, o respetivo número de ocorrências da **regex**.

Em relação à função **top_reg_in_file** de forma semelhante, aplica a expressão regular **regex** a cada texto de cada entidade (texto este capturado a aprtir do campo **atributes** que vai estar presente em cada tag, p.e. "Body"no Posts.xml), no entanto a grande diferença em relação à anterior é no momento de guardar no dicionário, o número de ocorrências (valor value), é associado à expressão regular (chave key) e não à entidade.

Capítulo 4

Como instalar e executar o xmlstats

4.1 Requisitos

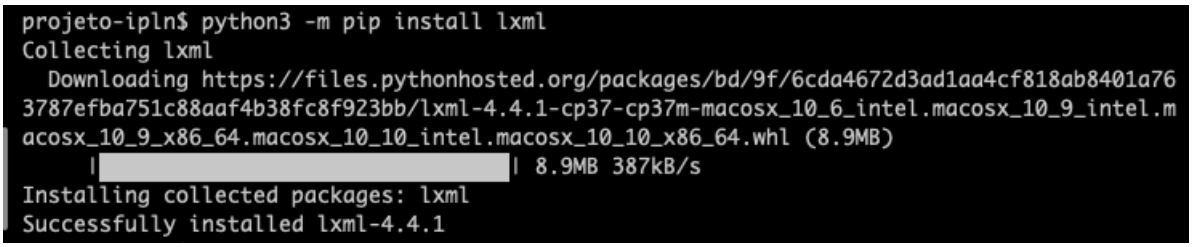

Figura 4.1: Instalação do lxml .

- Python 3
- lxml (como isntalar na imagem abaixo)
- Package de scripts do xmlstats (enviados juntamente com este relatório)
- Ficheiros .xml (ficheiros possíveis aqui)

4.2 Como executar

Para execução do programa e caso a equipa não tenha colocado no (.zip), deverá descarregar ficheiros .xml,no entanto recomendamos ir ao seguinte website **https://archive.org/download/stackexchange** e procurar por **StackOverflow**. Para facilitar o processo, a equipa desenvolveu um makefile que se pode ver na figura 5.2, onde se pode substituir os argumentos necessários para execução do programa. Os argumentos necessários são:

- -t (0 para executar funcionalidade por entidade ou 1 para a funcionalide por expressão regular)
- -f (pode ser um único ficheiro ou uma lista, do tipo 'file1, file2, fil3')
- -a (atributos das tags de cada ficheiro respetivamente 'atr1,atr2,atr3')
- -r expressão regular

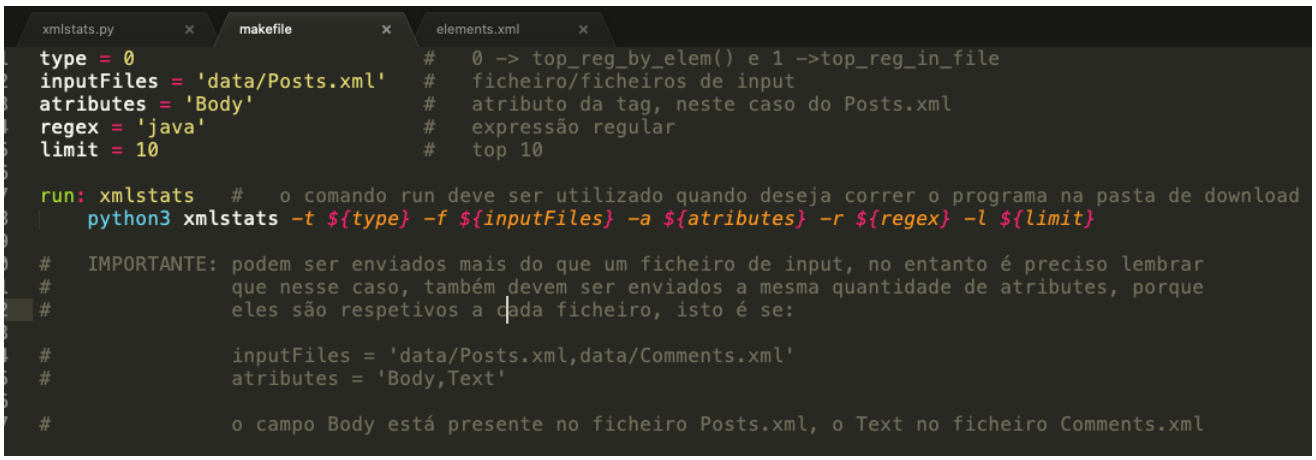

Figura 4.2: Ficheiro makefile.

- -l limite para top N

De seguida, na figura 5.3, apresenta-se a execução deste programa ao ficheiro Posts.xml, onde o atributo analisado é o Body, presente nas tags deste ficheiro, a expressão regular é a palavra java, e o output é uma lista de tuplos, onde o valor da esquerda representa o id do Post e o da direita o número de ocorrências da expressão regular.

```
tp2$ python3 xmlstats.py -t 0 -f 'data/Posts.xml' -a 'Body' -r 'java' -l 10
[(870, 63), (1841, 21), (551, 12), (1727, 12), (1829, 10), (334, 9), (1904, 9), (50, 8), (1907, 8), (4182, 8)]
tp2$
```

Figura 4.3: Como usar.

Capítulo 5

Conclusão

Em suma, os objetivos inicialmente propostos foram cumpridos, no entanto a equipa entende que deveria ter entendido melhor a estrutura interna da ferramenta **lxml**. No entanto, com uma análise da documentação presente no website da ferramenta, consegiu-se perceber a excelente **ElementTree API**, utilizada neste programa. Percebeu-se também a importância de estudar ferramentas desconhecidas. Por fim, a funcionalide inicialmente proposta, de aplicar expressões regulares a ficheiros **XML** foi concretizada com sucesso.