Processamento/Teoria de Linguagens e Compilação
LCC (3ºano) + MEFis (1ºano)

Trabalho Prático nº 1 (ER + Filtros de Texto)

Ano lectivo 22/23

Objectivos e Organização

Este trabalho prático tem como principais **objectivos**:

- aumentar a capacidade de escrever *Expressões Regulares (ER)* para descrição de *padrões de frases* dentro de textos;
- desenvolver, a partir de ER, sistematicamente *Processadores de Linguagens Regulares*, ou *Filtros de Texto (FT)*, que filtrem ou transformem textos com base no conceito de regras de produção *Condição-Ação*;
- utilizar o módulo 're'—com suas funções de search(), split(), sub()—do Python para implementar os FT pedidos.

Para o efeito, esta folha contém 5 enunciados, dos quais deverá escolher pelo menos um.
Neste TP que se pretende que seja resolvido rapidamente, aprecia-se a imaginação/criatividade dos grupos ao incluir outros processamentos!

O ficheiro único com o relatório e a solução deve ter o nome '`plc21TP1grNGr`' e será submetido até ao fim do dia 13.nov através do Bb.
O programa desenvolvido será apresentado aos membros da equipa docente, totalmente pronto e a funcionar (acompanhado do respectivo relatório de desenvolvimento) e será defendido por todos os elementos do grupo, em data a marcar.
O **relatório** a elaborar, deve ser claro e, além do respectivo enunciado, da descrição do problema, das decisões que lideraram o desenho da solução e sua implementação (incluir o código Python), deverá conter exemplos de utilização (textos fontes diversos e respectivo resultado produzido).
Como é de tradição, o relatório será escrito em LaTeX.

1 Processador de Pessoas listadas nos Róis de Confessados

Construa agora um ou vários programas Python para processar o texto '`processos.txt`' com o intuito de calcular frequências de alguns elementos (a ideia é utilizar arrays associativos para o efeito) conforme solicitado a seguir

a) Calcula a frequência de processos por ano (primeiro elemento da data);

b) Calcula a frequência de nomes próprios (o primeiro em cada nome) e apelidos (o ultimo em cada nome) por séculos;

c) Calcula a frequência dos vários tipos de relação: irmão, sobrinho, etc.

d) imprimir os 20 primeiros registos num novo ficheiro de output mas em formato Json.

2 Processador de Registos de Exames Médicos Desportivos

Neste exercício pretende-se trabalhar com um dataset gerado no âmbito do registo de exames médicos desportivos.
Construa, então, um ou vários programas Python para processar o dataset `"emd.csv"` e produzir o solicitado nas alíneas seguintes:

- Criar um website com as seguintes caraterísticas:

  1. Página principal: de nome `"index.html"`, contendo os seguintes indicadores estatísticos:
     (a) Datas extremas dos registos no dataset;
     (b) Distribuição por modalidade em cada ano e no total;
     (c) Distribuição por idade e género (para a idade, considera apenas 2 escalões: < 35 anos e $>= 35$);
     (d) Distribuição por morada;
     (e) Percentagem de aptos e não aptos por ano.
  2. Página do indicador: clicando no indicador na página principal, devemos saltar para a página do indicador onde temos a informação que permitiu obter esse indicador. Por exemplo, para a distribuição por morada, a página deverá apresentar uma lista de moradas, ordenada alfabeticamente e para cada morada deverá apresentar uma sublista de registos, ordenada alfabeticamente por nome de atleta (com os dados: nome do atleta, modalidade).

3 EnameXPro, processador de Enamex

O Reconhecimento de Entidades Nomeadas (em inglês *NER, Named Entities Recognition*) é uma atividade muito complexa que constitui ainda hoje um enorme desafio para os investigadores em PLN (Processamento de Lingua Natural). Existem inclusive concursos internacionais para verificar quem é capaz de desenvolver reconhecedores com maior precisão e acuidade (que marquem corretamente todos os casos de entidades cujo nome surge num dado texto de entrada, em análise).
O problema todo está em criar reconhecedores capazes de identificar os nomes num dado texto e indicar se se trata de uma *pessoa*, *local* (cidade ou país), ou *organização*.

Neste trabalho o que se lhe pede é para pós-processar um ficheiro já criado por um processador NER que anotou o texto de entrada com etiquetas (tags XML) da norma ENAMEX—veja o anexo `'exemplo-Enamex.xml'`—e responder às alíneas seguintes:

1. Criar uma página HTML com todos os nomes de pessoas encontrados (por ordem alfabética e sem repetições).
2. Tal como na alinea anterior, criar uma página com todos os locais identificados indicando se é uma cidade ou país.
3. Ajudar a resolver as indefinições, isto é, os casos em que o processador NER original identificou um nome (palavra começada por maiúscula) mas não conseguiu saber que tipo de entidade estava a ser nomeada (a maioria das vezes porque não era mesmo o nome de uma entidade).

4 BibTeXPro, Um processador de BibTeX

BibTeX é uma ferramenta de formatação de citações bibliográficas em documentos LaTeX, criada com o objectivo de facilitar a separação da base de dados da bibliografia consultada da sua apresentação no fim do documento LaTeX em edição. BibTeX foi criada por Oren Patashnik e Leslie Lamport em 1985, tendo cada entrada nessa base de dados textual o aspecto que se ilustra a seguir

```
@InProceedings{CPBFH07e,
  author =    {Daniela da Cruz and Maria João Varanda Pereira
               and Mário Béron and Rúben Fonseca and Pedro Rangel Henriques},
  title =     {Comparing Generators for Language-based Tools},
  booktitle = {Proceedings of the 1.st Conference on Compiler
               Related Technologies and Applications, CoRTA'07
               --- Universidade da Beira Interior, Portugal},
  year =      {2007},
  editor =    {},
  month =     {Jul},
  note  =     {}
}
```

De modo a familiarizar-se com o formato do BibTeX poderá consultar o ficheiro `exemplo-utf8.bib` que se anexa e ainda a página oficial do formato referido (`http://www.bibtex.org/`), devendo para já saber que a primeira palavra (logo a seguir ao caracter "@") designa a categoria da referência (havendo em BibTeX pelo menos 14 diferentes).
As tarefas que deverá executar neste trabalho prático são:

a) Analise o documento BibTeX referido acima e faça a contagem das categorias (`phDThesis`, `Misc`, `InProceeding`, etc.), que ocorrem no documento. No final, deverá produzir um documento em formato HTML com o nome das categorias encontradas e respectivas contagens.

b) Complete o processador de modo a filtrar, para cada entrada de cada categoria, a respectiva chave (a 1ªpalavra a seguir à chaveta), autores e título. O resultado final deverá ser incluído no documento HTML gerado na alínea anterior.

c) Crie um índice de autores, que mapeie cada autor nos respectivos registos, de modo a que posteriormente uma ferramenta de procura do Linux possa fazer a pesquisa.

d) Construa um Grafo que mostre, para um dado autor (definido à partida) todos os autores que publicam normalmente com o autor em causa.
Recorrendo à linguagem Dot do GraphViz[1], gere um ficheiro com esse grafo de modo a que possa, posteriormente, usar uma das ferramentas que processam Dot [2] para desenhar o dito grafo de associações de autores.

5 Ficheiros CSV com listas e funções de agregação

Neste enunciado pretende-se fazer um conversor de um ficheiro **CSV** (*Comma separated values*) para o formato **JSON**.
Para se poder realizar a conversão pretendida, é importante saber que a primeira linha do **CSV** dado funciona como cabeçalho que define o que representa cada coluna.
Por exemplo, o seguinte ficheiro `"alunos.csv"`:

```
Número,Nome,Curso
3162,Cândido Faísca,Teatro
7777,Cristiano Ronaldo,Desporto
264,Marcelo Sousa,Ciência Política
```

Corresponde à seguinte tabela:

Número	Nome	Curso
3162	Cândido Faísca	Teatro
7777	Cristiano Ronaldo	Desporto
264	Marcelo Sousa	Ciência Política

No entanto, neste trabalho, os **CSV** recebidos têm algumas extensões.

5.1 Listas

Nestes *datasets*, poderemos ter conjuntos de campos que formam listas.

Listas com tamanho definido

No cabeçalho, cada campo poderá ter um número N que representará o número de colunas que esse campo abrange.
Por exemplo, imaginemos que ao exemplo anterior se acrescentou um campo **Notas**, com $N = 5$ (`"alunos2.csv"`):

```
Número,Nome,Curso,Notas{5},,,,,
3162,Cândido Faísca,Teatro,12,13,14,15,16
7777,Cristiano Ronaldo,Desporto,17,12,20,11,12
264,Marcelo Sousa,Ciência Política,18,19,19,20,18
```

[1] Disponível em `http://www.graphviz.org`
[2] Disponíveis em `http://www.graphviz.org/Resources.php` ou a ferramenta Web `http://www.webgraphviz.com/`

Isto significa que o campo **Notas** abrange 5 colunas. (Reparem que temos de meter os campos que sobram a vazio, para o **CSV** bater certo).

Listas com um intervalo de tamanhos

Para além de um tamanho único, podemos também definir um intervalo de tamanhos $\{N, M\}$, significando que o número de colunas de um certo campo pode ir de N até M. (`"alunos3.csv"`)

```
Número,Nome,Curso,Notas{3,5},,,,,
3162,Cândido Faísca,Teatro,12,13,14,,
7777,Cristiano Ronaldo,Desporto,17,12,20,11,12
264,Marcelo Sousa,Ciência Política,18,19,19,20,
```

5.2 Funções de agregação

Para além de listas, podemos ter funções de agregação, aplicadas a essas listas.
Veja os seguintes exemplos (`"alunos4.csv"` e `"alunos5.csv"`):

```
Número,Nome,Curso,Notas{3,5}::sum,,,,,
3162,Cândido Faísca,Teatro,12,13,14,,
7777,Cristiano Ronaldo,Desporto,17,12,20,11,12
264,Marcelo Sousa,Ciência Política,18,19,19,20,
```

```
Número,Nome,Curso,Notas{3,5}::media,,,,,
3162,Cândido Faísca,Teatro,12,13,14,,
7777,Cristiano Ronaldo,Desporto,17,12,20,11,12
264,Marcelo Sousa,Ciência Política,18,19,19,20,
```

etc.

5.3 Resultado esperado

O resultado final esperado é um ficheiro **JSON** resultante da conversão dum ficheiro **CSV**.
Por exemplo, o ficheiro `"alunos.csv"` (original), deveria ser transformado no seguinte ficheiro `"alunos.json"`:

```
[
    {
        "Número": "3612",
        "Nome": "Cândido Faísca",
        "Curso": "Teatro"
    },
    {
        "Número": "7777",
        "Nome": "Cristiano Ronaldo",
        "Curso": "Desporto"
    },
    {
        "Número": "264",
        "Nome": "Marcelo Sousa",
        "Curso": "Ciência Política"
    }
]
```

No caso de existirem listas, os campos que representam essas listas devem ser mapeados para listas em **JSON** (`"alunos2.csv"`):

```
[
    {
        "Número": "3612",
        "Nome": "Cândido Faísca",
        "Curso": "Teatro",
        "Notas": [12,13,14,15,16]
    },
    {
        "Número": "7777",
        "Nome": "Cristiano Ronaldo",
        "Curso": "Desporto",
        "Notas": [17,12,20,11,12]
    },
    {
        "Número": "264",
        "Nome": "Marcelo Sousa",
        "Curso": "Ciência Política",
        "Notas": [18,19,19,20,18]
    }
]
```


No caso em que temos uma lista com uma função de agregação, o processador deve executar a função associada à lista, e colocar o resultado no **JSON**, identificando na chave qual foi a função executada (`"alunos4.csv"`):


```
[
    {
        "Número": "3612",
        "Nome": "Cândido Faísca",
        "Curso": "Teatro",
        "Notas_sum": 39
    },
    {
        "Número": "7777",
        "Nome": "Cristiano Ronaldo",
        "Curso": "Desporto",
        "Notas_sum": 72
    },
    {
        "Número": "264",
        "Nome": "Marcelo Sousa",
        "Curso": "Ciência Política",
        "Notas_sum": 76
    }
]
```