Processamento de Linguagens
MiEI (3ºano)

Trabalho Prático nº 1 (FLex)

Ano lectivo 15/16

1 Objectivos e Organização

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

- aumentar a experiência de uso do ambiente Linux, da linguagem imperativa C (para codificação das estruturas de dados e respectivos algoritmos de manipulação), e de algumas ferramentas de apoio à programação;
- aumentar a capacidade de escrever *Expressões Regulares (ER)* para descrição de *padrões de frases*;
- desenvolver, a partir de ERs, sistemática e automaticamente *Processadores de Linguagens Regulares*, que filtrem ou transformem textos;
- utilizar *geradores de filtros de texto*, como o Flex

Para o efeito, esta folha contém vários enunciados, dos quais deverá resolver pelo menos um.

Deve entregar (enviando por correio eletrónico para os docentes) a sua solução **até Domingo dia 25 de Outubro**. Não faça um ficheiro comprido zip nem envie vários ficheiros; deve submeter apenas 1 ficheiro, o PDF do relatório, com toda a informação incluida conforme se explica abaixo. Para ser aceite, no campo *Assunto* do email deve escrever: `LCC::PLC15::TP1::NNNNNppppaaaa;NNNNNppppaaaa` em que `NNNNN` é o número do aluno, `pppp` o seu nome próprio e `aaaa` o apelido. O ficheiro a anexar deve ter o nome "`plc15TP1GrNN`" em que `NN` é o número do grupo (de 01 a 09).
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 a especificação Flex), deverá conter exemplos de utilização (textos fontes diversos e respectivo resultado produzido). Como é de tradição, o relatório será escrito em LaTeX.

2 Enunciados

Para sistematizar o trabalho que se pede em cada uma das propostas seguintes, considere que deve, em qualquer um dos casos, realizar a seguinte lista de tarefas:

1. Especificar os padrões de frases que quer encontrar no texto-fonte, através de ERs.
2. Identificar as acções semânticas a realizar como reacção ao reconhecimento de cada um desses padrões.
3. Identificar as Estruturas de Dados globais que possa eventualmente precisar para armazenar temporariamente a informação que vai extraindo do texto-fonte ou que vai construindo à medida que o processamento avança.
4. Desenvolver um Filtro de Texto para fazer o reconhecimento dos padrões identificados e proceder à transformação pretendida, com recurso ao Gerador Flex.

2.1 Processamento de Ontologias em OWL

O uso de ontologias em informática, para representar rigorosamente o conhecimento de determinado domínio, é cada vez mais comum e importante. Basicamente, uma ontologia é formada por um conjunto de *conceitos* (também chamados *classes*) que pertencem a esse domínio e por um conjunto de relações (hierárquicas, ou não-hierárquicas) entre esses conceitos.
Para o efeito foi definido há alguns anos um dialeto XML, chamado OWL[1], que permite escrever ontologias em formato legível pelo humano e processável pelo computador. A dita linguagem de anotação é muito completa e complexa, porém neste contexto apenas nos interessa considerar as anotações que indicam as relações entre as classes de modo a poder desenhar-se o grafo que descreve a ontologia em causa.
Para perceber o problema analise com cuidado o fragmento de uma ontologia escrita em OWL que se mostra abaixo e que descrevem duas relações, `receives` e `owns`: a primeira que liga a classe `Laundry` com a classe `Order`; e a segunda que liga a classe `Client` com a classe já referida `Order`.

```
<ObjectPropertyDomain>
    <ObjectProperty IRI="#receives"/>
    <Class IRI="#Laundry"/>
</ObjectPropertyDomain>
<ObjectPropertyRange>
    <Annotation>
        <AnnotationProperty abbreviatedIRI="owl:backwardCompatibleWith"/>
        <IRI>#Laundry</IRI>
    </Annotation>
    <ObjectProperty IRI="#receives"/>
    <Class IRI="#Order"/>
</ObjectPropertyRange>

<ObjectPropertyDomain>
    <ObjectProperty IRI="#owns"/>
    <Class IRI="#Client"/>
</ObjectPropertyDomain>
<ObjectPropertyRange>
    <Annotation>
        <AnnotationProperty abbreviatedIRI="owl:backwardCompatibleWith"/>
        <IRI>#Client</IRI>
    </Annotation>
    <ObjectProperty IRI="#owns"/>
    <Class IRI="#Order"/>
</ObjectPropertyRange>
```

por sua vez a declaração a seguir exemplificada diz que a classe `Type` tem uma propriedade `material` cujo valor (a instanciar posteriormente) será do tipo `string`.

```
<DataPropertyDomain>
    <DataProperty IRI="#material"/>
    <Class IRI="#Type"/>
</DataPropertyDomain>
<DataPropertyRange>
    <DataProperty IRI="#material"/>
    <Datatype abbreviatedIRI="xsd:string"/>
</DataPropertyRange>
```

por fim, mostra-se abaixo um outro exemplo em que se definem ligações hierárquicas entre duas classes (ou conceitos), em que se diz que `Emigracao` é uma subclasse de `Evento` e que `Emigrante` é uma subclasse de `Pessoa`.

```
<SubClassOf>
    <Class IRI="Emigracao"/>
    <Class IRI="Evento"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="Emigrante"/>
    <Class IRI="Pessoa"/>
</SubClassOf>
```

Neste contexto, desenvolva um Filtro de Texto com o Flex para ler uma ontologia descrita em OWL e para desenhar um grafo que ligue os conceitos entre si e os ligue também ao tipo das suas propriedades, recorrendo para isso à linguagem Dotty e ao sistema de construção de grafos Dot.

[1] do inglês *Ontology Web Language*.

2.2 Normalizador de ficheiros 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 com a 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 Joao Varanda Pereira
               and Mario Beron and Ruben 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},
}
```

De modo a familiarizar-se com o formato do BibTeX poderá consultar o ficheiro `lp.bib` disponível em `http://www.di.uminho.pt/~prh/lp.bib` 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) Desenvolva uma ferramenta de normalização (sempre que um campo está entre aspas, troque para chavetas e escreva o nome dos autores no formato "`N. Apelido`") e faça uma ferramenta de *pretty-printing* que indente corretamente cada campo, escreva um autor por linha e coloque sempre no início os campos autor e título.

c) Construa um Grafo que mostre, para um dado autor (escolhido pelo utilizador) todos os autores que publicam normalmente com o autor em causa.
Recorrendo à linguagem Dot do GraphViz[2], gere um ficheiro com esse grafo de modo a que possa, posteriormente, usar uma das ferramentas que processam Dot [3] para desenhar o dito grafo de associações de autores.

2.3 From treebanks to probablilistic grammar

Considere o seguinte extracto do corpus Floresta-sintática:

```
#8 CP2-3 Mas como, se muitas não dispõem, nos seus quadros, dos técnicos necessários?
(FRASE CP2-3
   (QUE:fcl
      (CO:conj-c:mas:: Mas)
      (ADVL:advp (H:adv:como:::interr: como))
      (,)
      (ADVL:fcl
        (SUB:conj-s:se:::: se)
        (SUBJ:np
           (H:pron-det:muito:F_P::quant: muitas))
        (ADVL:advp
           (H:adv:não:::: não))
        (P:vp (MV:v-fin:dispor:PR_3P_IND::fs-cond: dispõem))
        (,)
        (ADVL:pp
```

[2] Disponível em `http://www.graphviz.org`
[3] Disponíveis em `http://www.graphviz.org/Resources.php`

```
            (H:prp:em:::: em+)
            (P<:np
               (>N:art:o:M_P::artd: os)
               (>N:pron-det:meu:M_P::si: seus)
               (H:n:quadro:M_P::anr_np-def: quadros)))
         (,)
         (PIV:pp
            (H:prp:de:::: de+)
            (P<:np
               (>N:art:o:M_P::artd: os)
               (H:n:técnico:M_P::np-def: técnicos)
               (N<:adjp
                  (H:adj:necessário:M_P::: necessários))))
      )
      (?)
   )
)
```

Dado um documento contendo (muitas) árvores sintáticas análogas à apresentada,

1. Despeje as produções implicitamente usadas (não interessa a ordem). Exemplo - Produções:

   ```
   fcl  → conj advp ',' fcl '?'
   advp → adv
   fcl  → conj-se np advp vp ',' pp ',' pp
   np   → pron-det
   advp → adv
   v    → v-fin
   pp   → prp np
   np   → art pron-det n
   pp   → prp np
   np   → art n adjp
   adjp → adj
   ```

2. Despeje o dicionário dos símbolos terminais. Exemplo - dicionário:

   ```
   os  : cat=art      lema=o
   seus: cat=pron-det lema=meu
   ```

3. Construa uma script que com base nos resultados anteriores, calcule os dicionários e gramáticas probabilísticos (quais as variantes e quantas vezes ocorrem).

2.4 Processador baseado em ABC para acompanhamento Musical

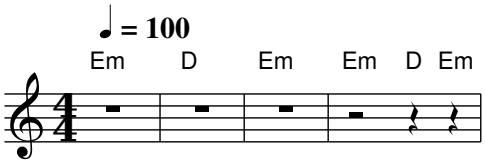
Instale as versões mais recentes das ferramentas abcm2ps e abc2midi, e descarregue a documentação (`http://abcplus.sourceforge.net/`, `http://abcplus.sourceforge.net/abcplus_en-2015-09-03.zip`).

Pretende-se facilitar a criação de partituras e midi para acompanhamentos de música (mesmo sem melodia!).
Construa um preprocessador para ABC que converta a primitiva adicional "chords"de acordo com o exemplo abaixo.
Extracto de ABC extendido com acordes:

```
X: 1
M: 4/4
K: C
Q: 1/4=100
L: 1/4
chords[accordion][fzcc]{Em|D|Em|Em . D Em|}
```

ao qual corresponde o seguinte ABC:

```
X: 1
M: 4/4                      % compasso
K: C                        % tonalidade
Q: 1/4=100                  % velocidade
L: 1/4
%%MIDI chordprog 21         % intrumento: accordion
%%MIDI gchord    fzcc       % modo de acompanhamenot  (opcional)
"Em"z4|                     % pausas com acordes de acompanhamento
"D"z4|
"Em"z4|
"Em"z2 "D"z "Em"z|
```



(só as linhas "chords...."deverão ser alteradas).
Complementarmente, crie um mecanismo semelhante para percussão.

2.5 Gerador de Slides para Beamer

Zim-wiki é uma ferramenta que permite criar documentos ricos com base em formato textual. Comporta-se como um wiki de desktop. Tomando como base essa sintaxe

- página geral da ferramenta: `http://zim-wiki.org`
- sintaxe geral:`http://www.zim-wiki.org/manual/Help/Wiki_Syntax.html`

crie um conversor para slides em Beamer. Sugestão use headers 1, 2 e 3 para quebrar slides.