SPLN

MI-EI

Exame, 17 de Junho de 2019

Questão 1. Escreva da forma mais simples e elegante que conseguir:

(a) Uma função Python que, dada uma lista de números inteiros, retorna uma lista de todos os números cuja soma com o número seguinte na lista é 10.

```
sum_next_is_10([1, 5, 7, 3, 6, 4, 8, 2, 10, 0, 3]) # [7, 6, 8, 10]
```

(b) Uma função Python que, dada uma `frase` (String) e um `alfabeto` (String), verifica se a frase é um pangrama: *"frase em que são usadas todas as letras do alfabeto de uma língua.*[1] Para este efeito, ignore a capitalização e a acentuação dos caracteres (pode usar a função `unidecode` do módulo `unidecode` que recebe uma String e a devolve sem acentuação).

```
frase1 = "Blitz prende ex-vesgo com cheque fajuto."
frase2 = "Juiz faz com que whisky de malte baixe logo preço de venda."
frase3 = "432109876533312315"
frase4 = "UMinho"

is_pangram(frase1, "abcdefghijlmnopqrstuvxz")    # True
is_pangram(frase2, "abcdefghijklmnopqrstuvwxyz") # True
is_pangram(frase3, "0123456789")                 # True
is_pangram(frase4, "UM Gualtar")                 # False
```

Questão 2. Pretende-se implementar em Python um script que, dada uma entidade nomeada `NE`, procura num texto – cujas entidades nomeadas foram previamente delimitadas com chavetas – entidades que co-ocorrem com NE na mesma frase. O script deverá imprimir as entidades nomeadas seguidas do número da frase em que foram encontradas.

```
$ cat text.txt
Atravessando a praça, {Cruges} pedia a {Deus} que não encontrassem mais o
{Euzebiosinho}. - Ah, murmurou {Carlos} com respeito e através dum bocejo, o
governo não está contente com o {Euzebiosinho}? {Carlos} desatou a rir, contou
ao avô que era um par do reino que o convidava a jantar, citando {Descartes}...
- Esteve ai, mas deitou a correr, para ir arranjar uma comenda para o
{Euzebiosinho}, disse {Carlos}, abrindo a carta.

$ co_occurs "Euzebiosinho" text.txt
Cruges 1
Deus 1
Carlos 2
Carlos 4
```

Questão 3. Pretende-se calcular a semelhança semântica entre conceitos representados por palavras analisando o contexto em que estas aparecem, baseado na ideia de que conceitos semelhantes (ex: "autocarro" e "camioneta") aparecem em contextos semelhantes. Para fazer esta comparação, decidiu-se usar n-gramas de palavras.

(a) Implemente uma funcao `calc_ngrams` que, dado uma coleção de textos muito grande (String) e um inteiro `N`, devolve um dicionário com a contabilização das ocorrências de `N`-gramas de palavras nesses textos.

[1] `https://dicionario.priberam.org/pangrama`

(b) Implemente uma função `similar_concepts()` que recebe como argumentos um `conceito` (palavra) e os `ngramas` calculados na alínea anterior, e retorna os 3 conceitos mais semelhantes a `conceito`.

```
similar_concepts('autocarro', ngrams) # ['camioneta', 'comboio', 'metro']
```

Questão 4. A avaliação de uma ferramenta `A` de deteção de entidades nomeadas foi feita da seguinte forma: num conjunto de textos foram manualmente anotadas as entidades. Depois esse conjunto de textos foi processado pela ferramenta. Por último, foi feita a comparação entre as marcações feitas manualmente e pela ferramenta, e foram calculados os valores de *precision* (precisão) e de *recall* (cobertura).

(a) Sabendo que o conjunto de textos tinha, no total, 1 milhão de palavras, das quais 1000 foram manualmente anotadas como entidades, e que os valores de *precision* e *recall* obtidos pela ferramenta foram, respectivamente, 0.7 e 0.9, determine quantas entidades foram correcta e incorrectamente classificadas pela ferramenta.

(b) Suponha que uma ferramenta concorrente `B` apresentava valores de *precision* e *recall* de, respectivamente, 0.8 e 0.6. Numa situação em que fosse importante garantir que o máximo possível de entidades era identificada (aceitando o risco de poder obter algumas palavras incorrectamente classificadas como entidades), qual deveria ser a ferramenta escolhida? Justifique.

(c) Descreva como implementaria uma ferramenta deste género.

Questão 5. Pretende-se implementar uma função `toXML()` que recebe três argumentos, `tag` (String), `attrs` (dicionário) e `content` (vários tipos possíveis), e devolve a representação em XML.

```
toXML('frase', {'autor': 'Camões'}, 'Alma minha que te partiste')
# <frase autor="Camões">Alma minha que te partiste</frase>
```

Dependendo do tipo de `content`, este deverá ser inserido na `tag` de formas diversas:

- se `content` for uma String, deverá ser considerada o conteúdo do elemento XML
- se `content` for uma lista, cada elemento da lista deverá ser usado como conteúdo de um elemento XML (com a etiqueta `tag`), aplicando a todos os mesmos atributos
- se `content` for um dicionário, deverá dar origem a sub-elementos, sendo a chave a etiqueta XML e o valor o respectivo conteúdo (não havendo atributos para os sub-elementos)
- se `content` for uma função, deverá ser invocada, passando-lhe a `tag` como argumento.

```
toXML('numero', {'lang': 'pt'}, ['um', dois'])
# <numero lang="pt">um</numero>
# <numero lang="pt">dois</numero>

toXML('ementa', {}, {'hoje':'badejo', 'amanha':'abrotea'})
# <ementa>
#   <hoje>badejo</hoje>
#   <amanha>abrotea</amanha>
# </ementa>

toXML('cafe', {}, lambda x: 'Vou tomar ' + x)
# <cafe>Vou tomar cafe</cafe>
```

Implemente a função `toXML()`. Pode tirar partido da função `isinstance(var, class)` que permite verificar se o primeiro argumento é instância do segundo, e da função `callable(obj)` que permite determinar se o argumento é invocável.

Questão 6. Pretende-se construir uma aplicação de linha de comandos para consultar os resultados de corridas do Campeonato Mundial de Fórmula 1.

(a) Assuma que existe uma variável `race_results` que contém os resultados para uma determinada corrida.

```
race_results = {
    date: '09 Jun 2019',
    location: 'Circuit Gilles-Villeneuve, Montréal',
    race: 'GRAND PRIX DU CANADA 2019',
    standings: [
      {
        pos: 1, laps: 70, pts: 25, time: '1:29:07.084',
        driver: 'Lewis Hamilton', team: 'Mercedes',
      }, {
        pos: 2, laps: 70, pts: 18, time: '+3.658s',
        driver: 'Sebastian Vettel', team: 'Ferrari',
      }, {
        pos: 3, laps: 70, pts: 15, time: '+4.696s',
        driver: 'Charles Leclerc', team: 'Ferrari',
      },
      ...
    ]
]
```

Implemente uma função `team_results()` que recebe como argumento a variável `race_results` e imprime a tabela classificativa das equipas, ordenada por pontuação (a pontuação de uma equipa é a soma das pontuações dos seus pilotos).

```
team_results(race_results)

# Mercedes 38
# Ferrari 33
# Renault 14
# ...
```

(b) Supondo que o URL para consultar os resultados do primeiro Grand Prix deste ano (realizado na Austrália) é `https://www.formula1.com/en/results.html/2019/races/1000/australia/race-result.html` e que o URL dos resultados do terceiro Grand Prix (China) é`https://www.formula1.com/en/results.html/2019/races/1002/china/race-result.html`, implemente a função `get_race_results`, que:

- recebe como parâmetros o número e localização de um Grand Prix deste ano
- faz *scrapping* da página correspondente
- extrai a informação relevante usando a biblioteca `BeautifulSoup`
- devolve a informação extraída no mesmo formato da variável `race_results` da alínea anterior.

Assuma que o HTML resultante do scrapping da página tem o seguinte formato:

```
<html>
  <head>[...]</head>
  <body>
    <div class="site-wrapper">
      <main role="main">
        <article>
          <div class="inner-wrap ResultArchiveWrapper">
            <div class="loader">[...]</div>
            <div class="ResultArchiveContainer">
              <div class="resultsarchive-filter-container">[...]</div>
              <div class="resultsarchive-wrapper">
                <div class="resultsarchive-content-header group">
                  <h1 class="ResultsArchiveTitle">FORMULA 1 PIRELLI GRAND PRIX DU CANADA 2019 - RACE RESULT</h1
                  <p class="date">
```

```
                    <span class="start-date">07</span>
                    <span class="full-date">09 Jun 2019</span>
                    <span class="circuit-info">Circuit Gilles-Villeneuve, Montréal</span>
                  </p>
                </div>
                <div class="resultsarchive-content group">
                  <div class="resultsarchive-col-left">[...]</div>
                  <div class="resultsarchive-col-right">
                    <table class="resultsarchive-table">
                      <thead>
                        <tr>
                          <th><abbr title="Position">Pos</abbr></th>
                          <th>Driver</th>
                          <th class=" hide-for-tablet">Car</th>
                          <th class=" hide-for-mobile">Laps</th>
                          <th>Time/Retired</th>
                          <th><abbr title="Points">PTS</abbr></th>
                        </tr>
                      </thead>
                      <tbody>
                        <tr>
                          <td class="dark">1</td>
                          <td class="dark bold">
                            <span class="hide-for-tablet">Lewis</span>
                            <span class="hide-for-mobile">Hamilton</span>
                            <span class="uppercase hide-for-desktop">HAM</span>
                          </td>
                          <td class="semi-bold uppercase hide-for-tablet">Mercedes</td>
                          <td class="bold hide-for-mobile">70</td>
                          <td class="dark bold">1:29:07.084</td>
                          <td class="bold">25</td>
                        </tr>
                        <tr>
                          <td class="dark">2</td>
                          <td class="dark bold">
                            <span class="hide-for-tablet">Sebastian</span>
                            <span class="hide-for-mobile">Vettel</span>
                            <span class="uppercase hide-for-desktop">VET</span>
                          </td>
                          <td class="semi-bold uppercase hide-for-tablet">Ferrari</td>
                          <td class="bold hide-for-mobile">70</td>
                          <td class="dark bold">+3.658<span class="suffix seconds">s</span></td>
                          <td class="bold">18</td>
                        </tr>
                        [...]
                      </tbody>
                    </table>
                  </div>
                </div>
              </div>
            </div>
          </article>
        </main>
      </div>
    </body>
  </html>
```