MC102 - Algoritmos e Programação de Computadores

Recomendação de palavras

Muitos editores de texto, navegadores e outras ferramentas computacionais nos mostram sugestões de palavras enquanto estamos digitando. Você já pensou em como isto é implementado? Inicialmente, o programa precisa adquirir um vocabulário. Para isto, pode receber uma lista de palavras ou fazer uma varredura em vários textos disponíveis. Ao fazer o processamento das palavras, o programa precisa organizá-las de maneira a tornar a busca pelas sugestões bastante eficiente. A presença de informações sobre a frequência das palavras na língua ou em contextos específicos, pode permitir a apresentação de um número menor mas relevante de sugestões.

Nesta tarefa, iremos implementar uma versão simplificada desta funcionalidade que aprenderá palavras processando pequenos textos de grandes nomes da Literatura Brasileira. Na fase de recomendação, o programa deverá listar todas as palavras conhecidas que contenham os prefixos indicados e que tenham pelo menos um caractere a mais que o prefixo.

Descrição da entrada

As primeiras linhas conterão o texto para o aprendizado do vocabulário. Esta parte terminará com uma linha contendo os caracteres `---` e outros metadados sobre o texto, como o nome do(a) autor(a) e da obra original, que não precisarão ser incluídos no vocabulário. Em seguida, a entrada conterá uma ou mais linhas com prefixos de palavras, sendo o final desta lista também marcado por `---`. Veja, no esquema abaixo, como será a divisão da entrada:

```
<linha_texto_1>
<linha_texto_2>
    .
    .
    .
<linha_texto_N>
--- <metadados>
< prefixo_1>
< prefixo_2>
    .
    .
    .
< prefixo_M>
---
```

Como nas tarefas anteriores, para evitar erros de codificação no envio das tarefas para o SuSy, todos os textos estarão apresentados propositalmente sem acentos. Os caracteres de pontuação que precisarão ser removidos durante o processamento são: `,` (vírgula), `.` (ponto final), `...` (reticências), `:` (dois pontos), `()` (abre e fecha parênteses), `!` (exclamação) e `?` (interrogação). Palavras ligadas por hífen deverão ser consideradas como duas ou mais palavras distintas. Por exemplo, o processamento de `anti-rosa` no poema Rosa de Hiroshima, de Vinícius de Moraes, resultará na inclusão de `anti` e `rosa` no vocabulário. Os prefixos serão compostos apenas por letras e, caso necessário, deverão ser convertidos para letras minúsculas.

Observe abaixo um pequeno exemplo com apenas uma linha de texto e com um único pedido de recomendação:

```
Os otimistas sao ingenuos, os pessimistas sao amargos, mas vale ser um realista esperancoso.
--- Ariano Suassuna
real
---
```

Descrição da saída

A primeira parte da saída conterá a string explicativa `Vocabulario:` seguida de uma lista em ordem alfabética das palavras aprendidas. Cada palavra aparecerá em uma linha, com a frequência amostrada entre parênteses. Note que o armazenamento e a contagem das ocorrências das palavras serão feitos considerando apenas letras minúsculas.

```
Vocabulario:
<palavra_1> (<freq_palavra_1>)
<palavra_2> (<freq_palavra_2>)
    .
    .
    .
<palavra_N> (<freq_palavra_N>)
```

A segunda parte será iniciada pela string explicativa `Sugestoes:`. A próximas linhas deverão apresentar para cada um dos prefixos lidos a lista de recomendações em ordem alfabética. Note que a lista de sugestões pode estar vazia, caso o vocabulário não contenha nenhuma palavra adequada (iniciada pelo prefixo e com pelo menos um caractere a mais do que ele).

```
Sugestoes:
<prefixo_1>: <sugestao_1_1> <sugestao_1_2> ... <sugestao_1_n>
<prefixo_2>: <sugestao_2_1> <sugestao_2_2> ... <sugestao_2_n>
      .
      .
      .
<prefixo_m>: <sugestao_m_1> <sugestao_m_2> ... <sugestao_m_n>
```

Para o exemplo da seção anterior, a saída será:

```
Vocabulario:
amargos (1)
esperancoso (1)
ingenuos (1)
mas (1)
os (2)
otimistas (1)
pessimistas (1)
realista (1)
sao (2)
ser (1)
um (1)
vale (1)
Sugestoes:
real: realista
```

Dicas para a realização desta tarefa:

- Para converter uma string para letras minúsculas utilize o método `lower()`:

  ```
  >>> "RETIRANTE".lower()
  retirante
  ```

- Para remover ou modificar um caractere em uma string utilize o método `replace()`:

  ```
  >>> "amargos,".replace(",", "")
  'amargos'
  >>> "anti-rosa".replace("-", " ")
  'anti rosa'
  ```

- Dicionários permitem incluir e recuperar facilmente as informações associadas a uma chave. Como exemplo, observe um dicionário que organiza uma lista de brinquedos por cor:

  ```
  >>> objetos_por_cor = {}  # cria dicionário vazio
  >>> objetos_por_cor["azul"] = []     # associa uma lista vazia à cor azul
  >>> objetos_por_cor["branca"] = ["bola"] # associa uma lista contendo um elemento à cor branca
  >>> objetos_por_cor["branca"].append("urso") # adiciona um elemento à lista da cor branca
  >>> print(objetos_por_cor["branca"]) # imprime lista associada à cor branca
  ['bola', 'urso']
  >>> print("preta" in objetos_por_cor) # imprime valor da expressão booleana que indica presença ou ausência da cor
  False
  >>> for cor in sorted(objetos_por_cor) : # percorre elementos em ordem crescente das chaves
     print(cor + ":", objetos_por_cor[cor])
  azul: []
  branca: ['bola', 'urso']
  ```

  Recomendamos o uso de dicionários para a criação do vocabulário e contagem das frequências.

- Uma maneira simples para apresentar as sugestões é percorrer todo o vocabulário e recomendar as palavras que contêm o prefixo solicitado. Em caso de vocabulários muito extensos, esta abordagem poderia causar um impacto negativo no desempenho do editor de textos, tornando a experiência do usuário menos agradável. Como você poderia utilizar dicionários para organizar as palavras de maneira que a apresentação das recomendações possa ser eficiente?

Testes com o SuSy

Esta tarefa terá sete testes abertos e um teste fechado. O teste fechado repete um dos textos já apresentados, solicitando os prefixos em ordem diferente. Releia, se necessário, as instruções para fazer os testes em Testes com o SuSy.

Orientações para submissão

Veja aqui a página de submissão da tarefa. O arquivo a ser submetido deve se chamar **`lab08.py`**. No link Arquivos auxiliares há um arquivo `aux08.zip` que contém todos os arquivos de testes abertos e seus respectivos resultados compactados.

O limite máximo será de 15 submissões. Serão considerados os resultados da última submissão.

O peso desta tarefa é 3.

O prazo final para submissão é 13/10/2019.

A nota desta tarefa é proporcional ao número de testes que executaram corretamente, desde que o código esteja coerente com o enunciado. **A submissão de um código que não implementa o algoritmo requisitado, mas que exibe as saídas esperadas dos testes abertos a partir da comparação de trechos da entrada será considerada fraude e acarretará a atribuição de nota zero à média final da disciplina.**