Projeto 1

Nota: O projeto foi realizado no moodle, por isso este 'enunciado' não corresponde ao real mas sim há cópia do enunciado oficial formatado num pdf por mim.

O objetivo deste projeto é criar um corretor ortográfico.

Baseado num vocabulário disponível com um conjunto de palavras, o corretor deve ler uma *string* com texto e indicar todas as palavras que não reconhece. Para além disso, deve sugerir correções baseadas num algoritmo de semelhança de palavras.

Cada caixa de pergunta irá pedir uma função. As caixas seguintes não dependem das vossas soluções anteriores. Assim, mesmo se não conseguir fazer uma pergunta, pode continuar a definir as funções seguintes.

Não devem mudar os cabeçalhos das funções!

1.

Esta caixa não é uma pergunta para responderem. O objetivo é dar-vos uma função para o projeto

Utilize a seguinte função carregarVocabulario que dado um nome de ficheiro com um vocabulário, carrega o conteúdo do ficheiro e devolve-o no formato lista, com os elementos ordenados lexicograficamente, e sem repetições.

Aqui iremos usar o ficheiro 'vocabulario.txt'. Ele está disponível aqui no teste, mas também na página da disciplina.

Exemplo:

Test	Result
`dic = carregarVocabulario('vocabulario.txt')` `print(len(dic))`	41952

2.

Defina a função `gerarPalavras` que recebe uma *string* com texto, e devolve uma lista com as várias palavras contidas na *string*, pela ordem que aparecem.

Aqui terá de filtrar vários elementos textuais, nomeadamente, parêntesis ()[], dígitos 0...9, pontuações .,:;?!, espaços e o símbolo de nova linha \n. Não deve devolver palavras vazias nem palavras só com números.

Para auxiliar consulte a documentação da função split do módulo re.

Exemplo:

Test	Result

`texto = """`
`Em 2020 observamos, e catalogamos (com fotografias), os barcos que chegaram ao Porto!`
`Até breve.`
`"""`
`for p in gerarPalavras(texto):`
`  print(p)`
	`Em`
	`observamos`
	`e`
	`catalogamos`
	`com`
	`fotografias`
	`os`
	`barcos`
	`que`
	`chegaram`
	`ao`
	`Porto`
	`Até`
	`breve`

3.

Vamos agora definir uma função de dissemelhança entre palavras.

Defina a função `mmLetras(palavra1, palavra2)` que devolve a subtração entre o tamanho da maior palavra dada e o número de letras iguais nas mesmas posições entre as duas palavras.

Exemplo:

Test	Result
`print( mmLetras('promessa', 'promessa') )`	`0`
`print( mmLetras('promessa', 'passagem') )`	`7`
`print( mmLetras('antes', 'depois') )`	`6`

4.

Vamos agora definir outra função de dissemelhança entre palavras.

Defina a função `edicoes(palavra1, palavra2)` que devolve o número mínimo de operações de edição necessárias para transformar uma palavra na outra.

As operações de edição podem ser as seguintes:

- inserir uma letra numa qualquer posição
- apagar uma letra
- substituir uma letra por outra letra

Por exemplo, a distância entre 'para' e 'prol' é 3 porque precisamos de três operações para transformar uma na outra, isto é, para -> pra -> pro -> prol.

Devem representar a informação numa matriz onde cada linha corresponde às letras da 1ª palavra, e as colunas as letras da 2ª palavra. No exemplo dado, a matriz seria inicializada assim:

```
     p  r  o  l
  [[ 0  1  2  3  4]
p [ 1  0  0  0  0]
a [ 2  0  0  0  0]
r [ 3  0  0  0  0]
a [ 4  0  0  0  0]]
```

Cada posição [i][j] da matriz irá representar a distância entre as *strings* `palavra1[:i]` e `palavra2[:j]`. A solução final depois de preencher a matriz, estará na célula do canto inferior direito.

É possível preencher a matriz a começar nas linhas acima, da esquerda para a direita.

Para este exemplo a matriz, depois de preenchida, irá ter os seguintes valores:

```
     p  r  o  l
  [[ 0  1  2  3  4]
p [ 1  0  1  2  3]
a [ 2  1  1  2  3]
r [ 3  2  1  2  3]
a [ 4  3  2  2  3]]
```

nota: esta é a função mais difícil do projeto. Podem deixar para o fim. As perguntas seguintes funcionam mesmo se não responderem a esta pergunta.

Exemplo:

Test	Result
`print( edicoes('promessa', 'promessa') )`	`0`
`print( edicoes('promessa', 'passagem') )`	`7`
`print( edicoes('antes', 'depois') )`	`5`

5.

Defina a função `sugerir` que recebe um vocabulário, uma palavra, uma função de distância e um inteiro positivo n de sugestões e devolve uma lista de n palavras do vocabulário mais próximas da palavra dada, de acordo com a função de distância.

Como referido, o primeiro critério para entrar na lista final é a distância. No caso de ter de escolher uma palavra entre duas ou mais palavras com a mesma distância, deve-se escolher aquela que tem menor ordem lexicográfica (ou seja, preferir aquela que aparece primeiro no vocabulário).

A lista final de sugestões deve aparecer ordenada lexicograficamente. Podem usar a função `sorted` que recebe uma lista de elementos (no nosso caso, *strings*) e devolve uma lista ordenada dos seus elementos.

Exemplo:

Test	Result
`print(sugerir(dic, 'promeessa', mmLetras, 2))`	`['profetisa', 'progresso']`
`print(sugerir(dic, 'promeessa', edicoes))`	`['homessa', 'premissa', 'pressa', 'processar', 'promessa']`

6.

Defina a função `corretor` que recebe um vocabulário, um texto, uma função de distância e um inteiro positivo n de sugestões e imprime um relatório com as correções sugeridas.

Por exemplo, para a *string*

```
texto = "Este paragrafo teem moito ero de excrita."
```

a execução corretor(dic, texto, edicoes, 4) deve produzir o seguinte relatório:

```
Este --> ['ante', 'arte', 'asae', 'este']
paragrafo --> ['barógrafo', 'paragrafar', 'paraguaio', 'parágrafo']
teem --> ['deem', 'leem', 'reem', 'tem']
moito --> ['coito', 'mito', 'moiro', 'moita']
ero --> ['aro', 'ebro', 'eco', 'ego']
excrita --> ['escriba', 'escrita', 'escrito', 'excitar']
```

onde as sugestões finais vêm ordenadas por ordem lexicográfica.

Só devem apresentar sugestões de correção para as palavras que não pertencem ao vocabulário (por exemplo, "de" não leva sugestões de correção).

Exemplo:

Test	Result

`teste = 'Este paragrafo teem moito ero de excrita.'`
`corretor(dic, teste, mmLetras, 5)`
	`Este --> ['ante', 'arte', 'asae', 'bote', 'este']`
	`paragrafo --> ['barógrafo', 'calígrafo', 'paragrafar', 'paraguaio', 'parágrafo']`
	`teem --> ['deem', 'leem', 'reem', 'trem', 'veem']`
	`moito --> ['coito', 'moiro', 'moita', 'morto', 'mosto']`
	`ero --> ['aro', 'eco', 'ego', 'elo', 'era']`
	`excrita --> ['cabrita', 'escriba', 'escrita', 'escrito', 'excretar']`

`teste = 'Este paragrafo teem moito ero de excrita.'`
`corretor(dic, teste, edicoes, 5)`
	`Este --> ['ante', 'arte', 'asae', 'bote', 'este']`
	`paragrafo --> ['agrafo', 'barógrafo', 'paragrafar', 'paraguaio', 'parágrafo']`
	`teem --> ['deem', 'leem', 'reem', 'tem', 'terem']`
	`moito --> ['coito', 'mito', 'moiro', 'moita', 'morto']`
	`ero --> ['aro', 'ebro', 'eco', 'ego', 'elo']`
	`excrita --> ['escriba', 'escrita', 'escrito', 'excitar', 'excretar']`