Fundamentos da Programação

Avaliação - 3

Analise de Sentimentos

Integrantes = {

: ['Pedro Marques Malheiros','pmmal@cesar.school'],

: ['José Reginaldo da Silva Júnior','jrsj@cesar.school'],

: ['Pedro Henrique Araújo','phaab@cesar.school'],

}

Professor = ['Eldrey Seolin','esg@cesar.school']

Lista de Palavras

Boas:

Legal, Bonito, Amar, Felicidade, Carinho, Sorte, Aproveitar, Momento, Bom, Divertido.

Neutras:

Talvez, Mas, Como, Acaso, Provável, Possivelmente, Porventura, Hipoteticamente, Supostamente e Depende

Ruins:

Ruim, Não, Difícil, Mal, Infeliz, Deplorável,Péssimo,Triste,Desagradável e Impossível.

Lista de Frases

Boas:

Bonito é ser feliz e acreditar que a sorte é você quem traz.

Estou muito feliz, pois passei o dia todo fazendo carinho no meu cachorro, que por sinal é muito bonito.

Felicidade é saber aproveitar todos os momentos como se fossem os últimos.

É preciso amar as pessoas como se não houvesse amanhã.

Neutras:

O dia de hoje não foi muito legal.

Talvez minha amiga não vai poder ir ao cinema, estou triste, mas meus outros amigos vão, então irei aproveitar o momento.

A faculdade está difícil, mas não tanto quanto eu esperava, talvez fique mais difícil depois.

Ruins:

Estou me sentindo bastante mal, pois estou triste porque meu cachorro morreu.

Infelizmente não podemos sair pois o tempo está ruim.

Pelo alagamento, a rua da academia está deplorável no dia de hoje.

Opção de Atribuição de Classe

Olhando a opção “a”, não sentimos firmeza pois poderia ocorrer proporções de palavras iguais, contudo o código não conseguiria classificar a frase, pois a opção "a" define a classificação da frase de acordo com a maior proporção de palavras com tal classe na frase.

Por fim, **definimos o uso da opção “b”** (Opção que utiliza o peso das palavras). Nós achamos da mesma a mais eficiente e otimizada para o nosso processo, pois o usuário ainda tem a opção de definir o peso da palavra entre -3 e 3, achamos isso importante porque cada palavra pode ter um valor sentimental diferente para cada pessoa.

A opção "b" classifica a frase de acordo com a soma dos pesos de cada palavra encontrada. Se resultado for maior ou igual a 3 a frase é Boa, resultado menor ou igual a -3 a frase é ruim e resultado entre -3 e 3 a frase é neutra.

Definições e Escolhas Feitas Pela Equipe

Arquivo de Frases

No arquivo de frases foi definido que as frases devem ser organizadas cada uma em uma linha, pois o código vai ler cada linha como uma frase.
Exemplo:

```
1   Bonito é ser feliz e acreditar que a sorte é você quem traz.
2   Estou muito feliz, pois passei o dia todo fazendo carinho no meu cachorro, que por sinal é muito bonito.
3   Felicidade é saber aproveitar todos os momentos como se fossem os últimos.
```

⋮

Arquivo de Palavras

No arquivo de palavras foi definido que as palavras devem ser colocadas lado a lado junto ao seu peso, os conjuntos (palavra, peso) devem ser separados por vírgula(" , ") e a palavra e o peso por ponto e virgula(" ; ").
Exemplo:

palavra1;peso1,palavra2;peso2,palavra3;peso3...

```
1   Legal;3,Bonito;2,Amar;3,Felicidade;3,Carinho;3,Sorte;1,Aproveitar;1,Momento;1,Bom;3,Divertido;2,Ruim;-3,Não;-1,Dificil;-2,Mal;-2,Infeliz;-3,
    Deplorável;-3,Péssimo;-3,Triste;-3,Desagradável;-1,Impossivel;-3,Talvez;0,Mas;0,Como;0,Acaso;0,Provável;0,Possivelmente;0,Porventura;0,
    Hipoteticamente;0,Supostamente;0,Depende;0
```

Arquivo de Saída

O arquivo de saída está organizado da seguinte forma:

Frase,Lista de Palavras,Resultado,Classificação

A lista de palavras está organizada da seguinte forma (Separadas por " ; "):

Palavra1;Palavra2;Palavra3..

Caso nenhuma palavra seja encontrada, no lugar da lista de palavras terá "Nenhuma Palavra Foi Encontrada" e no lugar da classificação terá "INDETERMINADA"

```
2   Bonito é ser feliz e acreditar que a sorte é você quem traz,Bonito;Sorte,3,BOA
3   Bento foi ao parque com os amigos para brincar acabou se machucando e voltou para casa,Nenhuma Palavra Foi Encontrada,0,INDETERMINADA
```

Explicação de Partes do Código

```
22  #Adiciona as palavras no dicionário
23  listaPalavrasPesos = palavrasPesos.split(',')
24  for i in range(len(listaPalavrasPesos)):
25
26      palavraPeso = listaPalavrasPesos[i]
27      #Valida se a string está no formato certo
28      while not ';' in palavraPeso:
29          print(f"\nError - Não tem ';' em ({palavraPeso})")
30          palavraPeso = input('Insira novamente no formato (palavra;peso): ')
31
32      palavra,peso = palavraPeso.split(';')
33
34      #Valida se a palavra não é um número
35      while palavra.isdigit():
36          print(f'\nError - Palavra não pode ser número({palavra})')
37          palavra = input('Digite a nova palavra: ')
38
39      condicao = True
40      while condicao:
41          try:
42
43              #Valida se o peso está entre -3 e 3
44              while int(peso) < -3 or int(peso) > 3:
45                  print(f'\nError - Peso não pode ser menor qu -3 ou maior que 3 Peso => ({peso})')
46                  peso = input(f'Digite o novo peso para a palavra => {palavra}: ')
47
48              palavras[palavra.upper()] = int(peso)
49
50          #Valida se o peso pode ser convertido para inteiro
51          except ValueError as error:
52              print('\nError - ',error)
53              peso = input(f'Digite o peso para está palavra: {palavra} => ')
54
55          else:
56              condicao = False
```

Nessa parte do código adiciona as palavras e pesos já lidos no dicionário "palavras", sendo a chave a palavra e o valor o seu peso. Nessa parte do código também há tratamento de erro caso o usuário tenha inserido de forma inadequada no arquivo de palavras e pesos. **Linha 28:** Valida se o usuário colocou ";", se não, ele pede novamente o input e levanta um erro. **Linha 35:** Valida se a palavra é um número, pois o usuário pode ter se confundido e trocado de lugar o peso e a palavra, caso isso ocorra o código levanta um erro e pede um input. **Linha 44:** Valida se o peso inserido está entre -3 e 3, caso não estiver, ele pede para o usuário inserir e levanta um erro. **Linha 51:** Levanta um erro caso o peso não possa ser convertido para int, caso não possa ser convertido, o código pede novamente o peso.

```python
#Avalia as frases
resultadoFile = open(f'{pathName}\\RESULTADO.csv','a',encoding='UTF-8')
resultadoFile.writelines('Frase,Palavras Na Fase,Resultado,Classificação\n')

for frase in listaFrasesUpper:
    palavrasNaFrase = []
    result = 0

    #Encontra as palavras na frase e faz a soma do resultado
    for palavra in frase.split(' '):

        if palavra in list(palavras):
            palavrasNaFrase.append(palavra)
            result += palavras[palavra]

    #Classifica a frase
    if result < 3 and result > -3:
        classFinal = 'NEUTRA'

    elif result >= 3:
        classFinal = 'BOA'

    else:
        classFinal = 'RUIM'

    #Valida se alguma palavra foi encontrada
    if len(palavrasNaFrase) > 0:
        palavrasNaFrase = ';'.join(palavrasNaFrase)

    else:
        palavrasNaFrase = 'Nenhuma Palavra Foi Encontrada'
        classFinal = 'INDETERMINADA'

    resultadoFile.writelines(f'{frase.lower().capitalize()},{palavrasNaFrase.lower().title()},{result},{classFinal}\n')
```

Nessa parte, o código avalia as frases, após já ter lido o arquivo de frases e adicionado as palavra e pesos ao dicionário. **Entre as linhas 67 e 71:** Aqui o código irá buscar palavra por palavra da frase no dicionário, caso a palavra seja encontrada, o código adiciona ela a lista "palavrasNaFrase" e soma o seu peso a variável "result". **Entre as linhas 74 e 81:** Determina a classificação da frase de acordo com o valor da variável "result". **Entre as linhas 84 e 89:** Valida se alguma palavra foi encontrada, caso nenhuma palavra tenha sido encontrada, ao invés de a lista de palavras não ter nada ela vai ter a informação "Nenhuma palavra foi encontrada" e a classificação será "INDETERMINADA". **Linha 91:** Adiciona o conteúdo no arquivo final que é o "RESULTADO.csv" com o a frase, lista de palavras, resultado e classificação.

Conclusão e Opinião do Grupo Sobre o Código

A princípio começamos com um código bruto e após a conclusão vimos alguns erros que não deixavam o código rodar como devido. As palavras repetiam e atrapalhavam o funcionamento. O primeiro passo tomado foi a correção dos erros. Após a correção e funcionamento, começamos com aprimoramentos simples para uma melhor otimização e estética do mesmo. Depois de alguns ajustes chegamos ao código final, melhor otimizado e mais limpo, concluindo o código no qual rodou perfeitamente e todos ficamos satisfeitos com o resultado, decidindo utilizar ele como o "produto final".