Preparando dados de teste

Transcrição

Nosso corretor já é capaz de corrigir um único tipo de erro, mas precisaremos **ampliar os operações de correção** para abranger mais casos equivocados.

Para comprovarmos se realmente melhoramos a eficiência do algoritmo, compararemos os resultados do corretor inicial com a versão final.

Porém, ainda precisamos **avaliar** o nosso modelo, pois só realizamos testes com a palavra "lógica" até agora.

Então definiremos uma função chamada `avaliador()` que receberá alguns **dados de teste**, mas ainda não sabemos quais são e os descobriremos adiante. Por enquanto, os chamaremos de `testes`.

Com isso, receberemos uma porcentagem da taxa de acerto do meu corretor, e a printaremos com a frase `"taxa de acerto"`.

Agora, precisaremos descobrir quais são esses dados recebidos, e para isso veremos a nossa base de dados `palavras.txt` preparada para teste.

Teremos uma lista com diversas palavras em pares, onde a primeira é a **correta** e a segunda está **errada**.

O `avaliador()` precisará receber a palavra incorreta, passá-la pelo corretor e verificar se este acertou o resultado.

No Google Colab, importaremos os dados e os prepararemos para a função. Antes desta, definiremos **`cria_dados_teste()`** recebendo justamente a base de dados `palavras.txt` que será chamada de **`nome_arquivo`**.

Como retorno, teremos uma `lista_palavras_teste`, a qual será criada sendo igual a uma lista vazia antes de `return`.

```
def cria_dados_teste(nome_arquivo):
    lista_palavras_teste = []

    return lista_palavras_teste

def avaliador(testes):
    print("taxa de acerto")
```

COPIAR CÓDIGO

Queremos ler cada uma das linhas do arquivo `.txt`, separando o lado correto do incorreto. Logo, precisaremos criar duas variáveis: a **`correta`** e a **`errada`**. Chamaremos essa leitura como `linha` mesmo.

Para separarmos as palavras por espaço, poderemos utilizar o método `split()`. Esta linha vem justamente da leitura do arquivo apelidada como `f` no início do curso.

Manteremos esta mesma nomenclatura para sermos consistentes. Então criaremos a variável `f` sendo igual a `open()` recebendo o `nome_arquivo`. Para que fique aberto para leitura, adicionaremos `"r"` como segundo parâmetro.

Em seguida, iremos *iterar* sobre o arquivo e leremos cada uma de suas linhas, fazendo um `for` em `linha` dentro de `f`. Quando realizarmos a *iteração*, transformaremos cada linha em uma `string`.

Então, para cada uma delas no meu arquivo `f`, faremos a divisão entre `correta` e `errada` com `.split()` como já escrevemos.

Realizaremos o `.append()` na `lista_palavras_teste` também, passando uma tupla onde o primeiro lado será a palavra `correta` e o segundo, a `errada`.

Por fim, fecharemos o arquivo com `f.close()`.

```
def cria_dados_teste(nome_arquivo):
    lista_palavras_teste = []
    f = open(nome_arquivo, "r")
    for linha in f:
        correta, errada = linha.split()
        lista_palavras_teste.append((correta, errada))
    f.close()
    return lista_palavras_teste
```


```
def avaliador(testes):
    print("taxa de acerto")
```

COPIAR CÓDIGO

Com isso, criamos a função que pegará os nossos dados e os preparará para avaliação.

Falta-nos enviar a base de dados para as máquinas do Google, pois ainda não está disponível; expandiremos a aba lateral do Colab para fazermos o upload de `palavras.txt`. Ao fim do carregamento, conseguiremos utilizar o arquivo de fato.

Em seguida, criaremos a variável `lista_teste` no lugar da definição do `avaliador()` que será apagado por enquanto. Essa lista receberá o retorno da função `cria_dados_teste()` com o `"palavras.txt"`.

Com isso, teremos as tuplas e poderemos avaliar nosso corretor com os dados da forma que precisamos.

```
def cria_dados_teste(nome_arquivo):
    lista_palavras_teste = []
    f = open(nome_arquivo, "r")
    for linha in f:
        correta, errada = linha.split()
        lista_palavras_teste.append((correta, errada))
    f.close()
    return lista_palavras_teste
```

```
lista_teste = cria_dados_teste("palavras.txt")
lista_teste
```

COPIAR CÓDIGO

Ao executarmos a célula, teremos a nossa base de dados em uma estrutura que facilita a criação do `avaliador()`.