Gerador de textos

Programação 1 - IMPA tech

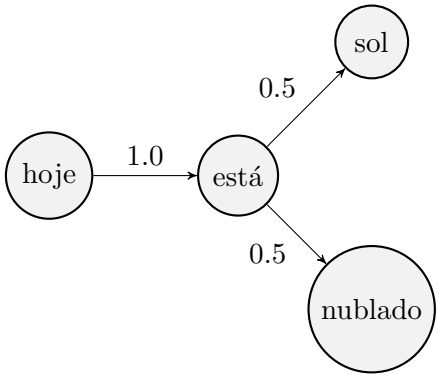
2025

Motivação

Imagine que você está digitando uma mensagem no celular e o teclado sugere a próxima palavra com base na última que você escreveu. Por exemplo, se você digitar "hoje está", o teclado pode sugerir "frio", "sol" ou "nublado", dependendo do que você costuma escrever.

Esse comportamento pode ser modelado com cadeias de Markov, onde a próxima palavra depende apenas da palavra atual (e não de todas que vem antes). A ideia é aprender padrões a partir de um texto e usá-los para gerar frases novas.

Exemplo. Na frase `"hoje está sol, hoje está nublado"`, após `"hoje"`, sempre vem `"está"`; após `"está"`, vem `"sol"` com 50% de chance ou `"nublado"` com 50% de chance. Podemos sintetizar a informação dessas probabilidades no diagrama a seguir:



Tarefa

Sua tarefa é implementar um programa em Python que receba um arquivo de texto com uma frase por linha, gerar um dicionário que vai armazenar a informação de probabilidade das palavras seguintes para cada palavra no texto, e a partir disso usar a biblioteca `random` do Python para gerar uma nova frase aleatoriamente seguindo as probabilidades obtidas.

Essa tarefa deve ser realizada seguindo rigorosamente as instruções abaixo, implementando cada função como se pede. A informação das probabilidades, que chamaremos de *dicionário de ocorrências* no que segue, deve ser armazenada no formato de um dicionário, que tem como chave as palavras que aparecem no texto, e como valor outro dicionário, que tem como chave as palavras seguintes que ocorreram no texto e como valor a quantidade de aparições. Caso uma palavra tenha todas suas aparições no final de frases (i.e., não admite palavras seguintes), o valor do dicionário de ocorrências nessa chave deve ser um dicionário vazio.

Exemplo. Se o texto consiste das frases `"hoje está sol"` e `"hoje está nublado"`, então o dicionário de ocorrências gerado deve ser o dicionário

```
{'hoje': {'está': 2}, 'está': {'sol': 1, 'nublado': 1}, 'sol': {}, 'nublado': {}}
```

Observe que apenas palavras seguintes que ocorreram ao menos uma vez devem ser contadas no dicionário.

1 - Obter Probabilidades

Tarefa 1.1. Implemente a função com o cabeçalho a seguir.

```
def frase_para_ocorrencias(frase):
```

Essa função deve receber uma frase em formato string e retornar o dicionário de ocorrências associado.

Tarefa 1.2. Implemente a função com o cabeçalho a seguir.

```
def concatenar_ocorrencias(ocorrencias1, ocorrencias2):
```

Essa função deve receber dois dicionários de ocorrências e retornar sua concatenação. O resultado dessa concatenação é um terceiro dicionário de ocorrências, que associa a cada palavra o dicionário com a soma das contagens de palavras seguintes ocorridas em `ocorrencia1` e `ocorrencia2`.

Tarefa 1.3. Implemente a função com o cabeçalho a seguir.

```
def arquivo_para_ocorrencias(arquivo):
```

Essa função deve receber um arquivo no modo leitura e retornar o dicionário de ocorrências associado. Você deve considerar que o arquivo recebido tem uma frase a cada linha. A função deve ler uma linha por vez, e utilizar as funções definidas nas questões anteriores para gerar o dicionário de ocorrências do arquivo completo.

Tarefa 1.4. Implemente a função com o cabeçalho a seguir.

```
def ocorrencias_para_frequencias(ocorrencias):
```

Essa função deve receber um dicionário de ocorrências e retornar o *dicionário de frequências* associado. Um dicionário de frequências é um dicionário no mesmo formato que um dicionário de ocorrências, mas que em vez de armazenar a informação de ocorrências de palavras seguintes, ele armazena a informação de frequência de palavras seguintes, isso é, o número de ocorrências de cada palavra divido pela soma das ocorrências de todas as palavras.

2 - Gerar Texto

Tarefa 2.1. Adicione no início do seu código a linha a seguir.

```
from random import choices
```

Essa linha nos dá acesso à função `choices`, que utilizaremos para fazer escolhas aleatórias com probabilidades prescritas.

Tarefa 2.2. Implemente a função com o cabeçalho a seguir.

```
def sortear_proxima_palavra(palavra_atual, frequencias):
```

Essa função deve receber uma string `palavra_atual` e um dicionário de frequências `frequencias`, e retornar uma das possíveis palavras seguintes à `palavra_atual`, sorteada de acordo com as probabilidades obtidas de `frequencias`. Se não houver nenhuma ocorrência de palavra seguinte para `palavra_atual`, essa função deve retornar uma string vazia.

Para realizar o sorteio, chame a função `choices` da seguinte maneira:

```
proxima_palavra = choices(proximas_palavras_possiveis, lista_de_frequencias)[0]
```

onde `proximas_palavras_possiveis` deve ser a lista de palavras que apareceram após `palavra_atual` no texto, e `lista_de_frequencias` deve ser a lista das frequências com que essas palavras ocorreram. Dessa maneira, a função `choices` retorna a próxima palavra sorteada, que armazenamos na variável `proxima_palavra`.

Tarefa 2.3. Implemente a função com o cabeçalho a seguir.

```
def gerar_texto(ocorrencias):
```

Essa função deve receber um dicionário de ocorrências e retornar uma string de texto com no máximo 50 palavras gerado aleatoriamente a partir dele. Lembre de utilizar a função que você definiu no item anterior.

A primeira palavra do texto deve ser sorteada utilizando a função `choices` da seguinte maneira:

```
primeira_palavra = choices(palavras_no_texto)[0]
```

onde `palavras_no_texto` deve ser uma lista sem repetições contendo todas as palavras do texto, e assim a função `choices` retorna a palavra sorteada para a variável `primeira_palavra`.

Testes

Para testar seu código, você pode utilizar o arquivo `g1_rj.txt` anexado, que contém títulos de matérias do site G1 sobre a região do Rio de Janeiro. Utilize as funções que você definiu para gerar o dicionário de ocorrências desse arquivo, e depois gerar texto baseado nos títulos dessas matérias.