deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

Relatório Orientado

Guião PL03

Métodos Probabilísticos para Engenharia Informática

Departamento de Eletrónica, Telecomunicações e Informática

Prof. Amaro Sousa

Ano letivo 2020/2021

Turma P3

André Pragosa Clérigo, 98485

Tiago Afonso Marques, 98459

30/11/2020

Índice

Introdução

Nota a manter durante este relatório, as alíneas do mesmo exercício foram feitas no mesmo ficheiro por isso aconselhamos a manter em mente que o código referido numa alínea pode ser referido nas alíneas seguintes.

Considere que:

- O ficheiro “wordlist-preao-20201103.txt” está no mesmo diretório dos scripts para o guião;
- O output resultante do código é referido abaixo da linha que delimita o código;
- Algum código é completamente reutilizado, logo código usado mais do que uma vez não irá ser comentado novamente;
- O código presenta na secção “Funções gerais” está presente em todos os exercícios;

Funções gerais

Nesta parte do relatório mostramos o conjunto de funções que usamos em todos os exercícios para nos facilitar a gerar palavras aleatórias.

```
% As Funções fornecidas no anexo do guião PL03
% Escuso os comentários pois estes já estão presentes no guião PL03
function state = crawl(T, first, last)
    state = [first];
    while (1)
        state(end+1) = nextState(T, state(end));
        if (state(end) == last)
            break;
        end
    end
end

function state = nextState(T, currentState)
    probVector = T(:, currentState)';
    n = length(probVector);
    state = discrete_rnd(1:n, probVector);
end

function state = discrete_rnd(states, probVector)
    U=rand();
    i = 1 + sum(U > cumsum(probVector));
    state = states(i);
end
```

```
% As nossas funções
% Função que gera uma palavra aleatória
function [word] = generateWord(T)
    first = randi(4);  %Probabilidade de primeira letra é igual para
todas
    state = crawl(T, first, 5);
    state = state(1: length(state) - 1); % Retirar o último estado
para coincidir com o numero de caracteres que temos em
"set_of_letters"
    set_of_letters= 'roma';
    word = set_of_letters(state);  % Substituir estado por letra
end

% Função que gera uma palavra aleatória a inicializar no estado first
function [word] = generateWordFirstLetter(T, first)
    state = crawl(T, first, 5);
    state = state(1: length(state)-1);
    set_of_letters= 'roma';
    word = set_of_letters(state);  % Substituir estado por letra
end
```

```
% Função que gera uma palavra aleatória com um tamanho <= size
function [word] = generateWordSized(T, size)
    first = randi(4);  %Probabilidade de primeira letra é igual para
todas
    state = crawlSized(T, first, 5, size);
    if (state(length(state)) == 5)
        state = state(1: length(state)-1);
    end
    set_of_letters= 'roma';
    word = set_of_letters(state);  % Substituir estado por letra
end


% Função que gera uma palavra aleatória com um tamanho <= size e a
inicializar no estado first
function [word] = generateWordSizedFirstLetter(T, size, first)
    state = crawlSized(T, first, 5, size);
    if (state(length(state)) == 5)
        state = state(1: length(state)-1);
    end

    set_of_letters = 'roma';
    word = set_of_letters(state);  % Substituir estado por letra
end


% Random walk on the Markov chain (alterada para uma palavra sized)
% size - the maximum size that the walk should have
function state = crawlSized(T, first, last, size)
    state = [first];
    while (1)
        state(end+1) = nextState(T, state(end));
        if (state(end) == last || length(state) == size)
            break;
        end
    end
end


% Gera um estado inicial com as novas probabilidades (utilizado para
os exercícios 3, 4 e 5)
function [estadoInicial] = generateState(Pa, Pm, Po, Pr)
al=rand();
    if(al<=Pa)
        estadoInicial=4;
    elseif(al <= Pa + Pm)
        estadoInicial=3;
    elseif(al <= Pa + Pm + Pr)
        estadoInicial=1;
    else
        estadoInicial=2;
    end
end
```

Exercício 1

Exercício 1 alínea a

```
     %r    %o    %m    %a   %.
T = [0    1/3   0     1/4   0
      .5   0     0.5   1/4   0
      0    1/3   0     1/4   0
      .5   0     0.5   0     0
      0    1/3   0     1/4   0];

word = generateWord(T);
fprintf("A palavra random gerada foi: %s.\n", word);
```

```
A palavra random gerada foi: a.
```

A palavra que foi gerada foi “a” e esta variável muda constantemente por cada vez que o código é executado. De notar que foi nesta alínea que foi criada a função “generateWord” e este usa a função crawl dada pelos professores e retira o último estado da variável “state” para conseguirmos formar uma palavra apenas com as letras e sem o caracter ponto.

Exercício 1 alínea b

```
N = 1e5;
lista= {};
lista{1} = generateWord(T);
contadores = {1};  % Inicializar o contador da primeira palavra

% Preencher cell array da lista de palavras únicas e cell array de
contadores
for i = 2 : N
    word = generateWord(T);
    % Código fornecido no e-learning
    a = ismember(lista, word); % returns an array of booleans
    pos = find(a == true);     % return the position(s) of the true

    % Se pos = [] significa que não pertence à lista
    if (isempty(pos))
    % Se a palavra gerada não pertence, adicionamos ao fim da lista
        lista{end+1} = word;
        contadores{end+1} = 1; % Se ainda não foi gerada o contador
vai ser inicializado
    else
        contadores{pos} = contadores{pos} + 1; % Se existir
incrementamos o contador
    end
end

% Preencher cell array com probabilidades
probabilidades = {1, length(contadores)};
for i = 1 : length(contadores)
    % Conta as probabilidades de cada palavra gerada
    probabilidades{i} = contadores{i} / N;
end

fprintf("Foram geradas %d palavras diferentes.\n", length(lista));
[p, idx] = sort(cell2mat(probabilidades), 'descend');  % Passa o cell
array para um array e ordena-o
for i = 1 : 5
    fprintf("A %dª maior probabilidade é de %s = %.4f.\n", i,
lista{idx(i)}, probabilidades{idx(i)});
end
```

```
Foram geradas 18410 palavras diferentes.
A 1ª maior probabilidade é de o = 0.0842.
A 2ª maior probabilidade é de a = 0.0618.
A 3ª maior probabilidade é de ro = 0.0410.
A 4ª maior probabilidade é de mo = 0.0406.
A 5ª maior probabilidade é de ra = 0.0323.
```

Neste exercício criamos a fundação do nosso algoritmo para resolver uma grande parte do guião. Criamos um cell array "lista" que contem todas as palavras geradas únicas, ou seja, sem repetição e ao mesmo tempo criamos um cell array "contadores" que é preenchido ao mesmo tempo que a "lista" e este associa (baseado na posição) um contador à palavra para sabermos quantas vezes essa palavra é gerada pela nossa matriz T. Por fim percorremos o "contadores" e criamos um cell array "probabilidades" que também associa (baseado na posição) a probabilidade de uma palavra ser gerada pela matriz T. Vemos que a nossa matriz gerou cerca de 18 mil palavras únicas e este valor não varia muito caso voltemos a dar "run" ao script. Por fim vemos quais os 5 valores de maior probabilidade e dizemos quais as 5 palavras mais prováveis e a sua respetiva probabilidade. É de notar que por cada "run" feita a este excerto de código as 5 palavras mais prováveis e as suas respetivas probabilidades variam.

```
     %r    %o    %m    %a   %.
T = [0     1/3   0     1/4   0
      .5   0     0.5   1/4   0
      0    1/3   0     1/4   0
      .5   0     0.5   0     0
      0    1/3   0     1/4   0];
```

Exercício 1 alínea c

p(x -> y) – probabilidades de ir do estado X para o estado Y

p(o) = p(1º letra) x p(m -> o) x p(o -> .) = 1/4 * 1/3 = 0.0833

p(a) = p(1º letra) * p(a -> .) = 1/4 * 1/4 = 0.0625

p(ro) = p(1º letra) * p(r -> o) * p(o -> .) = 1 /4 * 1 /2 * 1/3 = 0.0417

p(mo) = p(1º letra) * p(m -> o) * p(o -> .)= 1/4 * 1/2 * 1/3 = 0.0417

p(ra) = p(1º letra) * p(r -> a) * p(a -> .)= 1/4 * 1/2 * ¼ = 0.0313

Ao observar que as probabilidades experimentais estão tão próximas das probabilidades teóricas, podemos concluir que os resultados acimas são os resultados acertados do experimento.

Exercício 1 alínea d

```
fid = fopen('wordlist-preao-20201103.txt', 'r');
dicionario = textscan(fid, '%s');
fclose(fid);
dicionario = dicionario{1,1};  % Cria um cell array de palavras
válidas em Português
soma = 0;

for i = 1 : length(probabilidades)
    a = ismember(lista{i}, dicionario);
    pos = find(a == true);
    if(not(isempty(pos)))
        soma = soma + probabilidades{i};  % Se a palavra for válida
somamos a sua probabilidade de ser gerada
    end
end
fprintf("A probabilidade de gerar uma palavra válida é de %.4f.\n",
soma);
```

```
A probabilidade de gerar uma palavra válida é de 0.3512.
```

Aqui tiramos proveito do cell array que produzimos que contem a probabilidade de uma palavra ser gerada pela nossa matriz T. Apenas verificamos se cada elemento do cell array "lista" está presente no nosso cell array "dicionario" que contém as palavras válidas em Português, se sim adicionamos a uma variável "soma" a correspondente probabilidade dessa palavra ser gerada. Ficando assim com um valor de 35,14% de probabilidade da nossa matriz T gerar uma palavra válida em Português.

Exercício 1 alínea e

```
word = generateWordSized(T, 4)
```

```
word = 'roro'
```

Foi nesta fase que desenvolvemos a função "generateWordSized" e "crawlSized" para adicionarmos um parâmetro que indica que a geração da palavra acaba caso o seu tamanho seja igual ao segundo parâmetro passado no argumento da função "generateWordSized", neste caso usamos uma constante para testar a função.

É de notar que por cada "run" feita neste código o variável "word" muda do comprimento (máximo até 4 caracteres).

Exercício 1 alínea f

```
N = 1e5;
for j = 4 : 2 : 8
    n = j;
    lista= {};
    contadores = {1};
    lista{1} = generateWordSized(T, j);
    fprintf("Para um size de %d.\n", j);

    % Preencher cell array da lista de palavras únicas e cell array de
contadores (totalmente comentado e explicado na alínea b)
for i = 2 : N
    word = generateWordSized(T, j);
    a = ismember(lista, word);
    pos = find(a == true);

    if (isempty(pos))
        lista{end+1} = word;
        contadores{end+1} = 1;
    else
        contadores{pos} = contadores{pos} + 1;
    end
end

    % Preencher cell array com probabilidades (código reutilizado)
    probabilidades = {1, length(contadores)};
    for i = 1 : length(contadores)
        probabilidades{i} = contadores{i} / N;
    end

    % Verificar as 5 palavras mais geradas (código reutilizado)
    fprintf("Foram geradas %d palavras diferentes.\n", length(lista));
    [p, idx] = sort(cell2mat(probabilidades), 'descend');  % Passa o
cell array para um array e ordena-o
    for i = 1 : 5
        fprintf("A %dª maior probabilidade é de %s = %.4f.\n", i,
lista{idx(i)}, probabilidades{idx(i)});
    end

    % Reutilização de código da alínea d)
    % neste momento já temos a variável dicionário inicializada
    soma = 0;

    for i = 1 : length(probabilidades)
        a = ismember(lista{i}, dicionario);
        pos = find(a == true);
        if(not(isempty(pos)))
            soma = soma + probabilidades{i};
        end
    end
    fprintf("A probabilidade de gerar uma palavra válida é de
%.4f.\n\n", soma);
end
```

```
Para um size de 4.
Foram geradas 61 palavras diferentes.
A 1ª maior probabilidade é de o = 0.0839.
A 2ª maior probabilidade é de a = 0.0609.
A 3ª maior probabilidade é de ro = 0.0425.
A 4ª maior probabilidade é de mo = 0.0420.
A 5ª maior probabilidade é de ra = 0.0326.
A probabilidade de gerar uma palavra válida é de 0.4970.

Para um size de 6.
Foram geradas 307 palavras diferentes.
A 1ª maior probabilidade é de o = 0.0839.
A 2ª maior probabilidade é de a = 0.0619.
A 3ª maior probabilidade é de ro = 0.0427.
A 4ª maior probabilidade é de mo = 0.0411.
A 5ª maior probabilidade é de ma = 0.0309.
A probabilidade de gerar uma palavra válida é de 0.3627.

Para um size de 8.
Foram geradas 1515 palavras diferentes.
A 1ª maior probabilidade é de o = 0.0826.
A 2ª maior probabilidade é de a = 0.0627.
A 3ª maior probabilidade é de mo = 0.0416.
A 4ª maior probabilidade é de ro = 0.0416.
A 5ª maior probabilidade é de ra = 0.0308.
A probabilidade de gerar uma palavra válida é de 0.3511.
```

No que toca a gerar palavras únicas quanto maior o size, maior o número de palavras únicas, o que acaba por ser óbvio vistos que há mais maneiras de duas palavras com o mesmo tamanho serem diferentes. No que toca ao size = 4 há uma maior probabilidade de gerar uma palavra válida, supomos que isto acontece porque para um número reduzido de letras, mas tendo a nosso dispor 2 vogais, as palavras geradas acabam por coincidir linguisticamente a palavras validas na língua portuguesa.

Comparando com os valores das alíneas b) e d) vemos que para um size 6 e 8 o valor da palavra ser valida é semelhante ao valor obtido na alínea d), no entanto o número de palavras únicas geradas é bastante menor. Supomos que este fenómeno é devido ao facto de uma palavra que não tem restrições de tamanho acaba por atingir o estado absorvente quando já tem um tamanho entre 7 e 10 caracteres, fazendo com que a probabilidade de ser válida seja semelhante, mas o número de palavras únicas ser bastante maior.

Exercício 2

```
    %r   %o   %m   %a  %.
T = [0   0.3  0    0.3  0
     .3  0    0.3  0.1  0
     0   0.2  0    0.2  0
     .7  0    0.7  0    0
     0   0.5  0    0.4  0];

N = 1e5;
fprintf("Para um size infinito.\n");
lista= {};
contadores = {1};
lista{1} = generateWord(T);

% Preencher cell array da lista de palavras únicas e cell array de
contadores (código reutilizado)
for i = 2 : N
    word = generateWord(T);
    a = ismember(lista, word);
    pos = find(a == true);
    if (isempty(pos))
        lista{end+1} = word;
        contadores{end+1} = 1;
    else
        contadores{pos} = contadores{pos} + 1;
    end
end

% Preencher cell array com probabilidades (código reutilizado)
probabilidades = {1, length(contadores)};
for i = 1 : length(contadores)
    probabilidades{i} = contadores{i} / N;  % Contém as probabilidades
de cada palavra gerada
end

% Verificar as 5 palavras mais geradas (código reutilizado)
fprintf("Foram geradas %d palavras diferentes.\n", length(lista));
[p, idx] = sort(cell2mat(probabilidades), 'descend');
for i = 1 : 5
    fprintf("A %dª maior probabilidade é de %s = %.4f.\n", i,
lista{idx(i)}, probabilidades{idx(i)});
end

fid = fopen('wordlist-preao-20201103.txt','r');
dicionario = textscan(fid, '%s');
fclose(fid);
dicionario = dicionario{1,1};  % Cria um cell array de palavras
válidas em Português
soma = 0;

% Somar as probabilidades de todas as palavras (código reutilizado)
for i = 1 : length(probabilidades)
    a = ismember(lista{i}, dicionario);
    pos = find(a == true);
    if(not(isempty(pos)))
        soma = soma + probabilidades{i};
    end
end
```

```
fprintf("A probabilidade de gerar uma palavra válida é de %.4f.\n\n",
soma);


% Para os sizes (4, 6 e 8) (código reutilizado do exercício 1f)
for j = 4 : 2 : 8
    n = j;
    lista= {};
    contadores = {1};
    lista{1} = generateWordSized(T, j);
    fprintf("Para um size de %d.\n", j);

    % Preencher cell array da lista de palavras únicas e cell array de
contadores
    for i = 2 : N
        word = generateWordSized(T, j);
        a = ismember(lista, word);
        pos = find(a == true);
        if (isempty(pos))
            lista{end+1} = word;
            contadores{end+1} = 1;
        else
            contadores{pos} = contadores{pos} + 1;
        end
    end

    % Preencher cell array com probabilidades
    probabilidades = {1, length(contadores)};
    for i = 1 : length(contadores)
        probabilidades{i} = contadores{i} / N;  %Contém as
probabilidades de cada palavra gerada
    end

    % Verificar as 5 palavras mais geradas (código reutilizado)
    fprintf("Foram geradas %d palavras diferentes.\n", length(lista));
    [p, idx] = sort(cell2mat(probabilidades), 'descend');
    for i = 1 : 5

    end

    soma = 0;
    % neste momento já temos a variável dicionário inicializada
    % Somar as probabilidades de todas as palavras (código
reutilizado)
    for i = 1 : length(probabilidades)
        a = ismember(lista{i}, dicionario);
        pos = find(a == true);
        if(not(isempty(pos)))
            soma = soma + probabilidades{i};
        end
    end
    fprintf("A probabilidade de gerar uma palavra válida é de
%.4f.\n\n", soma);
end
```

```
Para um size infinito.
Foram geradas 7243 palavras diferentes.
A 1ª maior probabilidade é de o = 0.1239.
A 2ª maior probabilidade é de a = 0.0994.
A 3ª maior probabilidade é de ra = 0.0706.
A 4ª maior probabilidade é de ma = 0.0700.
A 5ª maior probabilidade é de mo = 0.0382.
A probabilidade de gerar uma palavra válida é de 0.5199.

Para um size de 4.
Foram geradas 61 palavras diferentes.
A 1ª maior probabilidade é de o = 0.1261.
A 2ª maior probabilidade é de a = 0.1001.
A 3ª maior probabilidade é de ma = 0.0712.
A 4ª maior probabilidade é de ra = 0.0700.
A 5ª maior probabilidade é de ro = 0.0376.
A probabilidade de gerar uma palavra válida é de 0.6531.

Para um size de 6.
Foram geradas 307 palavras diferentes.
A 1ª maior probabilidade é de o = 0.1239.
A 2ª maior probabilidade é de a = 0.1017.
A 3ª maior probabilidade é de ma = 0.0707.
A 4ª maior probabilidade é de ra = 0.0701.
A 5ª maior probabilidade é de mo = 0.0376.
A probabilidade de gerar uma palavra válida é de 0.5380.

Para um size de 8.
Foram geradas 1473 palavras diferentes.
A 1ª maior probabilidade é de o = 0.1251.
A 2ª maior probabilidade é de a = 0.1006.
A 3ª maior probabilidade é de ma = 0.0701.
A 4ª maior probabilidade é de ra = 0.0694.
A 5ª maior probabilidade é de mo = 0.0376.
A probabilidade de gerar uma palavra válida é de 0.5199.
```

Comparando os resultados entre si voltamos a verificar que há uma aproximação do valor de gerar uma palavra válida quando o size = 8, ao valor obtido para um size infinito e vemos que para um size <= 6 estes valores são consideravelmente maiores, supondo outra vez que este aumento se trata de uma coincidência linguística da língua portuguesa com o número de vogais que temos disponíveis na matriz T aliado ao número de letras que utilizamos para gerar uma palavra (size).

Comparando com a matriz anterior concluímos que para um size > 6 esta é menos eficaz a gerar palavras únicas, mas que para um size <= 6 tem a mesma eficácia a gerar palavras únicas, no que toca a gerar palavras válidas vemos que esta matriz supera a anterior em cerca de 10%. No entanto continuamos a verificar a mesma tendência na probabilidade de gerar uma palavra válida quando variamos o tamanho máximo dessa mesma. Sendo assim podemos dizer que esta matriz é mais eficaz a gerar palavras validas, mas menos eficaz a gerar palavras únicas.

Exercício 3

```
fid = fopen('wordlist-preao-20201103.txt','r');
dicionario = textscan(fid, '%s');
fclose(fid);
dicionario = dicionario{1,1};  % Cria um cell array de palavras
válidas em Português

r = ["r"];
o = ["o", "ó", "ò", "ô"];
m = ["m"];
a = ["a", "à", "á", "â", "ã"];
letters = ["b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "n",
"p", "q", "s", "t", "u", "v", "w", "x", "y", "z", "ç", "í"];
filtrado = {};

for I = 1 : length(dicionario)
    %se a palavra contiver r,o,m,a e não contiver as letras em
letters, essa palavra é adicionada a um novo cell array (só com as
letras r,o,m e a)
    TF = contains(dicionario{i}, r, 'IgnoreCase', true) &
    contains(dicionario{i}, o, 'IgnoreCase', true) &
    contains(dicionario{i}, m, 'IgnoreCase', true) &
    contains(dicionario{i}, a, 'IgnoreCase', true) &
    not(contains(dicionario{i}, letters, 'IgnoreCase', true));
    if(TF == true)
        filtrado{end + 1} = dicionario{i};
    end
end
filtrado = filtrado.';

Na = 0;   %n palavras começadas por a
Nm = 0;   %n palavras começadas por m
No = 0;   %n palavras começadas por o
Nr = 0;   %n palavras começadas por r
%percorre o novo cell array com as palavras apenas com as letras r,o,m
e a
for i= 1 : length(filtrado)
    chr = strcat(filtrado{i}(1)); %1 char da palavra
    %se a palavra começar por a soma +1 a Na
    if(strcmp(chr, 'A') | strcmp(chr, 'a'))
        Na= Na+1;
    %se a palavra começar por m soma +1 a Nm
    elseif(strcmp(chr, 'M') | strcmp(chr, 'm'))
        Nm= Nm+1;
    %se a palavra começar por o soma +1 a No
    elseif(strcmp(chr, 'O') | strcmp(chr, 'o')  | strcmp(chr, 'Ò'))
        No= No+1;
    %se a palavra começar por r soma +1 a Nr
    elseif(strcmp(chr, 'R') | strcmp(chr, 'r'))
        Nr= Nr+1;
    else

    end
end
%probabilidades
Pa = Na/length(filtrado);
Pm = Nm/length(filtrado);
Po = No/length(filtrado);
Pr = Nr/length(filtrado);
```

```
    %r   %o   %m   %a  %.
T = [0   0.3  0    0.3  0   %r
     .3  0    0.3  0.1  0   %o
     0   0.2  0    0.2  0   %m
     .7  0    0.7  0    0   %a
     0   0.5  0    0.4  0]; %.

%%%%%%%%%%%%%%%%%%%%%%%%%%COMPARAÇAO%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N = 1e5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%INFINITO%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf("Para um size infinito.\n");
lista= {};
contadores = {1};
lista{1} = generateWord(T, generateState(Pa, Pm, Po, Pr));

% Preencher cell array da lista de palavras únicas e cell array de
contadores (código reutilizado)
for i = 2 : N
    word = generateWord(T, generateState(Pa, Pm, Po, Pr));
    a = ismember(lista, word);
    pos = find(a == true);
    if (isempty(pos))
        lista{end+1} = word;
        contadores{end+1} = 1;
    else
        contadores{pos} = contadores{pos} + 1;
    end
end

%Preencher cell array com probabilidades (código reutilizado)
probabilidades = {1, length(contadores)};
for i = 1 : length(contadores)
    probabilidades{i} = contadores{i} / N;  % Contém as probabilidades
de cada palavra gerada
end

%Transforma o cell array em matriz e ordena por ordem descendente
(código reutilizado)
fprintf("Foram geradas %d palavras diferentes.\n", length(lista));
[p, idx] = sort(cell2mat(probabilidades), 'descend');
for i = 1 : 5
    fprintf("A %dª maior probabilidade é de %s = %.4f.\n", i,
lista{idx(i)}, probabilidades{idx(i)});
end

%Lê dicionario e verifica se as palavras geradas pertencem a esse
%dicionario, se sim adiciona a sua probabilidade (código reutilizado)
soma = 0;
for i = 1 : length(probabilidades)
        a = ismember(dicionario, lista{i});
        pos = find(a == true);
        if(not(isempty(pos)))
            soma = soma + probabilidades{i};
        end
end
fprintf("A probabilidade de gerar uma palavra válida é de %.4f.\n\n",
soma);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%4,6,8%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j = 4 : 2 : 8
    lista= {};
    contadores = {1};
    lista{1} = generateWordSized(T, j, generateState(Pa, Pm, Po, Pr));
    fprintf("Para um size de %d.\n", j);

    %Preencher cell array da lista de palavras únicas e cell array de
contadores
    for i = 2 : N
        word = generateWordSized(T, j, generateState(Pa, Pm, Po, Pr));
        a = ismember(lista, word);
        pos = find(a == true);
        if (isempty(pos))
            lista{end+1} = word;
            contadores{end+1} = 1;
        else
            contadores{pos} = contadores{pos} + 1;
        end
    end
end
```

```
Para um size infinito.
Foram geradas 7031 palavras diferentes.
A 1ª maior probabilidade é de a = 0.1822.
A 2ª maior probabilidade é de ma = 0.0938.
A 3ª maior probabilidade é de mo = 0.0516.
A 4ª maior probabilidade é de ra = 0.0381.
A 5ª maior probabilidade é de ara = 0.0379.
A probabilidade de gerar uma palavra válida é de 0.5846.

Para um size de 4.
Foram geradas 61 palavras diferentes.
A 1ª maior probabilidade é de o = 0.1828.
A 2ª maior probabilidade é de ra = 0.0952.
A 3ª maior probabilidade é de mora = 0.0511.
A 4ª maior probabilidade é de rama = 0.0493.
A 5ª maior probabilidade é de amom = 0.0383.
A probabilidade de gerar uma palavra válida é de 0.7126.

Para um size de 6.
Foram geradas 307 palavras diferentes.
A 1ª maior probabilidade é de mamaro = 0.1811.
A 2ª maior probabilidade é de ao = 0.0949.
A 3ª maior probabilidade é de a = 0.0512.
A 4ª maior probabilidade é de mo = 0.0390.
A 5ª maior probabilidade é de amama = 0.0374.
A probabilidade de gerar uma palavra válida é de 0.6103.

Para um size de 8.
Foram geradas 1423 palavras diferentes.
A 1ª maior probabilidade é de mamao = 0.1817.
A 2ª maior probabilidade é de ma = 0.0946.
A 3ª maior probabilidade é de ra = 0.0511.
A 4ª maior probabilidade é de maorara = 0.0382.
A 5ª maior probabilidade é de arama = 0.0381.
A probabilidade de gerar uma palavra válida é de 0.5871.
```

Olhando para os resultados obtidos no exercício 2 e comparando-os com os resultados agora obtidos no exercício 3, é nos claro que a matriz T do exercício 3 tem uma maior probabilidade de gerar uma palavra válida, independentemente do valor de size, enquanto esta mantem o mesmo número de palavras diferentes geradas para um size=4 e size=6 e uma pequena variação de palavras para um size=8 e size=infinito.

Assim sendo, embora a matriz T do exercício 3 gere menos palavras diferentes para um size>=8, esta gera um número de palavras diferentes igual para um size<=6 (em comparação ao exercício 2). Tendo isso em conta, a matriz T do exercício 3 tem uma probabilidade muito mais alta de gerar uma palavra válida independentemente do valor de size, pelo que a matriz T do exercício 3 será a mais eficaz entre as duas matrizes do exercício 2 e exercício 3.

Exercício 4

Do exercício 3 para o exercício 4, a única coisa que mudamos é os valores da matriz T por isso a matriz utilizada é a seguinte e inserimos esta matriz no código do exercício 3, o output resultante foi o seguinte.

```
    %r   %o   %m   %a  %.
T = [0   .3   0    .3   0   %r
     .3  0    .3   .1   0   %o
     .1  .2   0    .2   0   %m
     .6  0    .7   0    0   %a
     0   .5   0    .4   0]; %.
```

```
Para um size infinito.
Foram geradas 8914 palavras diferentes.
A 1ª maior probabilidade é de a = 0.1841.
A 2ª maior probabilidade é de ma = 0.0936.
A 3ª maior probabilidade é de mo = 0.0505.
A 4ª maior probabilidade é de o = 0.0348.
A 5ª maior probabilidade é de ara = 0.0331.
A probabilidade de gerar uma palavra válida é de 0.5742.

Para um size de 4.
Foram geradas 77 palavras diferentes.
A 1ª maior probabilidade é de a = 0.1799.
A 2ª maior probabilidade é de ma = 0.0936.
A 3ª maior probabilidade é de mo = 0.0508.
A 4ª maior probabilidade é de mara = 0.0429.
A 5ª maior probabilidade é de o = 0.0343.
A probabilidade de gerar uma palavra válida é de 0.6978.

Para um size de 6.
Foram geradas 456 palavras diferentes.
A 1ª maior probabilidade é de a = 0.1812.
A 2ª maior probabilidade é de ma = 0.0957.
A 3ª maior probabilidade é de mo = 0.0504.
A 4ª maior probabilidade é de o = 0.0341.
A 5ª maior probabilidade é de ara = 0.0331.
A probabilidade de gerar uma palavra válida é de 0.5954.

Para um size de 8.
Foram geradas 2315 palavras diferentes.
A 1ª maior probabilidade é de a = 0.1823.
A 2ª maior probabilidade é de ma = 0.0963.
A 3ª maior probabilidade é de mo = 0.0519.
A 4ª maior probabilidade é de o = 0.0336.
A 5ª maior probabilidade é de ra = 0.0329.
A probabilidade de gerar uma palavra válida é de 0.5750.
```

Comparando os resultados entre si chegamos à mesma conclusão existente nos exercícios anteriores, vemos uma relação entre a probabilidade de gerar uma palavra válida e o tamanho máximo desta, vendo que para um size = 8 os resultados são semelhantes ao que foi calculado para um size infinito, mas uma probabilidade consideravelmente maior para um size <= 6. Comparando os resultados obtidos com o exercício anterior podemos dizer que a matriz T (exercício 4) gera mais palavras únicas para qualquer size, podendo assim dizer que esta matriz é mais eficaz a gerar palavras únicas do que a anterior. Vistos que a probabilidade de gerar uma palavra válida é menor (em qualquer situação) do que os valores obtidos anteriormente podemos dizer que a matriz T (exercício 4) é menos eficaz a gerar uma palavra válida.

Exercício 5

```
fid = fopen('wordlist-preao-20201103.txt','r');
dicionario = textscan(fid, '%s');
fclose(fid);
dicionario = dicionario{1,1};

r = ["r"];
o = ["o", "ó", "ò", "ô"];
m = ["m"];
a = ["a", "à", "á", "â", "ã"];
letters = ["b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "n",
"p", "q", "s", "t", "u", "v", "w", "x", "y", "z", "ç", "í"];
filtrado={};

for i = 1:length(dicionario)
    TF = contains(dicionario{i}, r,  IgnoreCase , true) &
contains(dicionario{i}, o,  IgnoreCase , true) &
contains(dicionario{i}, m,  IgnoreCase , true) &
contains(dicionario{i}, a,  IgnoreCase , true) &
not(contains(dicionario{i},letters,  IgnoreCase , true));
    if(TF==true)
        filtrado{end+1}=dicionario{i};
    end
end
filtrado = filtrado.';

Na = 0;   % n palavras começadas por a
Nm = 0;   % n palavras começadas por m
No = 0;   % n palavras começadas por o
Nr = 0;   % n palavras começadas por r
for i = 1 : length(filtrado)
    chr = strcat(filtrado{i}(1)); % 1 char da palavra
    if(strcmp(chr, 'A') | strcmp(chr, 'a'))
        Na= Na+1;
    elseif(strcmp(chr, 'M') | strcmp(chr, 'm'))
        Nm= Nm+1;
    elseif(strcmp(chr, 'O') | strcmp(chr, 'o')  |strcmp(chr, 'Ò'))
        No= No+1;
    elseif(strcmp(chr, 'R') | strcmp(chr, 'r'))
        Nr= Nr+1;
    else

    end
end

%probabilidades
Pa = Na/length(filtrado);
Pm = Nm/length(filtrado);
Po = No/length(filtrado);
Pr = Nr/length(filtrado);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%GERAR MATRIZ T%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %r   %o   %m   %a  %.
Tex=[0    0    0    0   0   %r
     0    0    0    0   0   %o
     0    0    0    0   0   %m
     0    0    0    0   0   %a
     0    0    0    0   0]; %.
```

```
% Percorre as palavras filtradas e pelas duas últimas letras da
palavra e através de comparação, verifica que transição ocorre, e
consequentemente soma essa transição (no seu devido lugar) na matriz
Tex
for i=1:length(filtrado)
    chr1 = strcat(filtrado{i}(end-1))
    chr2 = strcat(filtrado{i}(end))
    if(strcmp(chr1,'r'))
        if(strcmp(chr2,'o') | strcmp(chr2,'ó') | strcmp(chr2,'ò') |
strcmp(chr2,'ô') | strcmp(chr2,'õ'))
            Tex(2,1)= Tex(2,1)+1;
            Tex(5,2)= Tex(5,2)+1;
        elseif(strcmp(chr2,'m'))
            Tex(3,1)= Tex(3,1)+1;
            Tex(5,3)= Tex(5,3)+1;
        elseif(strcmp(chr2,'a') | strcmp(chr2,'á') | strcmp(chr2,'à')
| strcmp(chr2,'ã') | strcmp(chr2,'â'))
            Tex(4,1)= Tex(4,1)+1;
            Tex(5,4)= Tex(5,4)+1;
        else
            fprintf("ERRO")
            break;
        end
    elseif(strcmp(chr1, 'o') | strcmp(chr1, 'ó') | strcmp(chr1, 'ò') |
strcmp(chr1, 'ô') | strcmp(chr1, 'õ'))
        if(strcmp(chr2,'r'))
            Tex(1,2) = Tex(1,2) + 1;
            Tex(5,1) = Tex(5,1) + 1;
        elseif(strcmp(chr2,'m'))
            Tex(3,2) = Tex(3,2) + 1;
            Tex(5,3) = Tex(5,3) + 1;
        elseif(strcmp(chr2,'a') | strcmp(chr2, 'á') | strcmp(chr2,
   ) | strcmp(chr2, 'ã') | strcmp(chr2, 'â'))
            Tex(4,2 )= Tex(4,2) + 1;
            Tex(5,4) = Tex(5,4) + 1;
        end
    elseif(strcmp(chr1, 'm'))
        if(strcmp(chr2, 'r'))
            Tex(1,3) = Tex(1,3) + 1;
            Tex(5,1) = Tex(5,1) + 1;
        elseif(strcmp(chr2,'o') | strcmp(chr2, 'ó') | strcmp(chr2,
'ò') | strcmp(chr2, 'ô') | strcmp(chr2, 'õ'))
            Tex(2,3) = Tex(2,3) + 1;
            Tex(5,2) = Tex(5,2) + 1;
        elseif(strcmp(chr2,'a') | strcmp(chr2,' á') | strcmp(chr2,
'à') | strcmp(chr2, 'ã') | strcmp(chr2, 'â'))
            Tex(4,3) = Tex(4,3) + 1;
            Tex(5,4) = Tex(5,4) + 1;
        end
    elseif(strcmp(chr1, 'a') | strcmp(chr1, 'á') | strcmp(chr1, 'à') |
strcmp(chr1, 'ã') | strcmp(chr1, 'â'))
        if(strcmp(chr2,'r'))
            Tex(1,4) = Tex(1,4) + 1;
            Tex(5,1) = Tex(5,1) + 1;
        elseif(strcmp(chr2,'o') | strcmp(chr2, 'ó') | strcmp(chr2,
   ) | strcmp(chr2, 'ô') | strcmp(chr2, 'õ'))
            Tex(2,4) = Tex(2,4) + 1;
            Tex(5,2) = Tex(5,2) + 1;
        elseif(strcmp(chr2,'m'))
            Tex(3,4) = Tex(3,4) + 1;
            Tex(5,3) = Tex(5,3) + 1;
```

```
        end
    end
end
% Neste momento a matriz Tex não contém as probabilidades de
transição, mas sim o número de vezes que elas ocorrem pelo que é
preciso transformá-la para conter as probabilidades
% Percorre a matriz Tex por colunas, e transforma-a de forma a gerar
as probabilidades de transição de cada estado
for i = 1 : 4
    Tex(:,i) = Tex(:,i)./sum(Tex(:,i));
end
T = Tex;
N = 1e5;
```

A partir do momento que temos a matriz T gerada usamos o código do exercício 3 (a partir da secção comentada com "COMPARAÇÃO") para calcular o as probabilidades de gerar uma palavra valida e calcular o número de palavras únicas para diferentes sizes. O output resultante é o seguinte.

```
Para um size infinito.
Foram geradas 340 palavras diferentes.
A 1ª maior probabilidade é de m = 0.1801.
A 2ª maior probabilidade é de ao = 0.1742.
A 3ª maior probabilidade é de a = 0.1296.
A 4ª maior probabilidade é de mo = 0.0871.
A 5ª maior probabilidade é de o = 0.0648.
A probabilidade de gerar uma palavra válida é de 0.5602.

Para um size de 4.
Foram geradas 55 palavras diferentes.
A 1ª maior probabilidade é de m = 0.1838.
A 2ª maior probabilidade é de ao = 0.1738.
A 3ª maior probabilidade é de a = 0.1298.
A 4ª maior probabilidade é de mo = 0.0868.
A 5ª maior probabilidade é de o = 0.0648.
A probabilidade de gerar uma palavra válida é de 0.5713.

Para um size de 6.
Foram geradas 189 palavras diferentes.
A 1ª maior probabilidade é de m = 0.1822.
A 2ª maior probabilidade é de mao = 0.1727.
A 3ª maior probabilidade é de a = 0.1298.
A 4ª maior probabilidade é de mo = 0.0862.
A 5ª maior probabilidade é de o = 0.0645.
A probabilidade de gerar uma palavra válida é de 0.5569.

Para um size de 8.
Foram geradas 321 palavras diferentes.
A 1ª maior probabilidade é de m = 0.1814.
A 2ª maior probabilidade é de ao = 0.1744.
A 3ª maior probabilidade é de o = 0.1316.
A 4ª maior probabilidade é de mo = 0.0867.
A 5ª maior probabilidade é de o = 0.0661.
A probabilidade de gerar uma palavra válida é de 0.5618.
```

Comparando os resultados entre si voltamos a verificar que há uma aproximação do valor de gerar uma palavra válida quando o size >= 6, ao valor obtido para um size infinito e vemos que para um size = 4 estes valores são maiores, supondo outra vez que este aumento se trata de uma coincidência linguística da língua portuguesa com o número de vogais que temos disponíveis na matriz T aliado ao número de letras que utilizamos para gerar uma palavra (size). Algo que conseguimos detetar comparando com os outros exercícios é que a diferença entre a probabilidade de gerar uma palavra válida para diferentes sizes mostram uma diferente linha de tendência referida nos exercícios anteriores, vendo que probabilidade de gerar uma palavra válida segue a seguinte ordem p(size 6) < p(size infinito) < p(size 8) < p(size 4), diferente do que era antes verificado p(size infinito) < p(size 8) < p(size 6) < p(size 4). Até agora não conseguimos chegar a uma teoria que explica o porque desta alteração ocorrer.

Comparando os resultados com os exercícios anteriores notamos que esta matriz é mais provável de gerar uma palavra válida do que a matriz utilizada nos exercícios 1 e 2, mas menos provável a gerar uma palavra válida do que as matrizes utilizadas nos exercícios 3 e 4. No entanto vemos que o número de palavras únicas geradas para esta matriz T é menor para qualquer size comparando com qualquer exercício anterior, podendo assim dizer que esta é a matriz menos eficaz de todos os exercícios a gerar palavras únicas.

Nota final

Como é utilizado sempre o mesmo processo para formar palavras aleatórias e a mesma matriz T, independente do size a ser utilizado, as probabilidades das palavras geradas irão ser muito próximas umas das outras para os diferentes valores de size.