Lab work nº 1

Relatório

Teoria Algorítmica da Informação

Gil Teixeira

João Relva

Vera Oliveira

Novembro 2021

Índice

1 Introdução

Como primeiro trabalho prático da unidade curricular de Teoria Algorítmica da Informação, o qual é analisado neste relatório, o objetivo seria construir dois programas: fcm e generator.

O fcm deveria permitir a recolha de informação estatística em textos, usando um modelo de contextos finitos, atribuindo uma estimativa de probabilidade aos símbolos presentes num determinado texto.

O generator deveria possibilitar a geração automática de texto tendo em conta um modelo estatístico previamente produzido (utilizando o fcm).

2 Desenvolvimento

2.1 Design

Para o desenvolvimento deste trabalho concordámos que a linguagem de programação que iriamos utilizar seria python devido à sua facilidade de utilização, eficiência e conveniência para a implementação de hash tables, que se tornaram fundamentais em todo o processo.

Considerando todo o contexto do trabalho, criámos dois ficheiros .py correspondentes aos dois programas a desenvolver: fcm e generator. No programa fcm.py definimos uma classe "FCM", responsável por criar o modelo finito de contextos, modelo esse que é utilizado no generator.py de forma a gerar o texto automático.

De forma a guardar os contextos e as diferentes probabilidades calculadas recorremos exclusivamente a hash tables.

2.2 Implementação

2.2.1 FCM

O programa é constituído por 7 funções (incluindo o construtor da classe):

- def __init__()
- def loadFromContext()
- def calculateProbabilities()
- def entropy()
- def rows_entropy()
- def countContextChildren()
- def createContext()

O construtor recebe o k, o alpha e o nome do ficheiro que são passados por argumentos e utiliza as funções para criar um objeto que vai conter todas as probabilidades de cada símbolo para determinado contexto e a entropia do modelo. Como já foi referido, a estrutura de dados utilizada foram hash tables.

A primeira tarefa do programa é ler o arquivo de texto fornecido e utilizar a função createContext() para criar uma hash table com todos os contextos e com a contagem dos símbolos que aparecem após um determinado contexto. De seguida são calculadas todas as probabilidades com a função calculateProbabilities().

O modelo irá considerar os contextos como grupos de k (a ordem do modelo) caracteres (que serão as linhas da tabela) e preencherá a tabela contando quantas vezes cada caracter (cada coluna da tabela) do alfabeto apareceu depois desse contexto. Para fazer isso, o modelo examina um contexto e o caracter após esse contexto. Se for um novo contexto, será adicionada uma nova linha à tabela. Finalmente, o modelo irá incrementar o contador para aquele caracter. Se esse caractere não existir,

então será adicionado à tabela como uma nova coluna e o contador para aquele contexto será inicializado a “1”.

A função countContextChildren() tem a função de devolver o número total de símbolos que ocorrem após um contexto para cada linha ou então da tabela toda. Após o cálculo de probabilidades é calculada a entropia através da função entropy() que utiliza a rows_entropy() para retornar o valor da entropia de cada linha. A função loadFromContext() permite carregar no programa um contexto que já tenha sido gravado.

2.2.2 Generator

O gerador utilizado o objeto do tipo FCM criado no programa fcm.py. Esse objeto já tem todos os contextos e todas as probabilidades de cada símbolo calculadas. Numa primeira instância são gerados os primeiros k caracteres e só depois é que é gerado o resto do texto. Sempre que é escolhido um símbolo esse mesmo é adicionado a uma variável texto que já contém todos os outros que foram gerados anteriormente. Para a geração pseudoaleatória é utilizada a função random.choices() em que é passado como argumento as probabilidades já previamente calculadas.

3 Resultados

Nesta seção serão apresentados todos os resultados obtidos após testes realizados os programas fcm.py e generator.py. Todos os testes foram realizados utilizando o ficheiro “example.txt” (exceto na seção de comparação com outros textos).

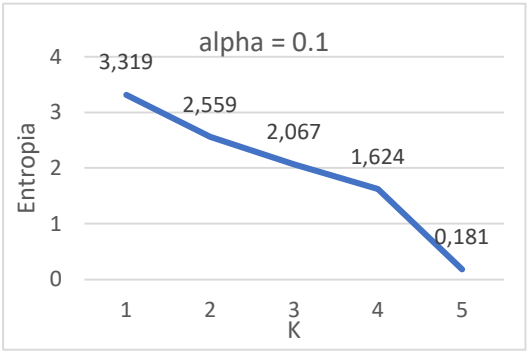
3.1 FCM

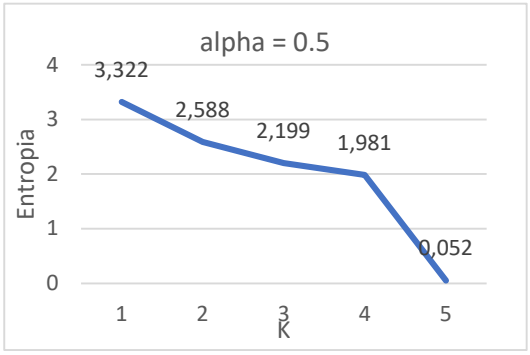
De forma a testar o funcionamento do fcm.py, foram utilizados diferentes valores de “alpha” e de “k”. Considerou-se recolher os resultados do cálculo da entropia e os tempos de execução do programa. Todos estes dados decidimos agrupar em tabelas e também elaborámos gráficos para cada valor de “alpha” correspondentes á variação da entropia para cada “k”.

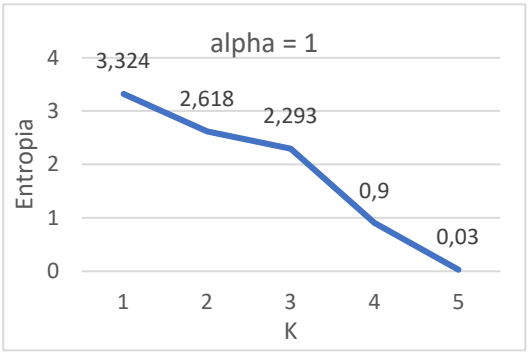
Valores da entropia em função de k e alpha

K/ALPHA	0.1	0.5	1
1	3.319	3.322	3.324
2	2.559	2.588	2.618
3	2.067	2.199	2.293
4	1.624	1.981	0.900
5	0.181	0.052	0.030

Variando o k, é possível observar que para cada valor diferente de k existe um diferente valor de entropia calculada.







Após a análise dos gráficos é possível verificar que os valores da entropia baixam à medida que o valor de k é incrementado. É espectável que a partir de certos valores de k a entropia estabilize.

Nos gráficos não é tão explícito, mas através da consulta da tabela referente aos valores da entropia em função de k e de alpha, pode-se notar que, maior parte das vezes, para cada valor de k, consoante o valor de alpha aumenta, a entropia também aumenta ainda que seja só ligeiramente.

Tempos de execução

K	Segundos
1	38.34
2	41.63
3	45.31
4	59.05
5	85.13

Após o cálculo dos tempos de execução verificamos que os mesmos aumentam consoante o K é incrementado.

3.2 Generator

De forma a testar o funcionamento do generator.py, apresentámos vários exemplos de textos gerados para os diferentes valores de k e de alpha. Foram também agrupados numa tabela os diferentes tempos de execução do gerador para diferentes tamanhos de texto gerado.

K = 1	alpha = 0.1	Text length = 100

```
9 d er: h woforin ben, tese gr
5 wr, amatheler thitwinse wed t hien un, pe thri thous hakm mare uthe
```

K = 1	alpha = 1	Text length = 100

```
8 coni be yierof and bofl o t

gen ng iler apis, s u esh m e, an of che harvangruald hr ivishe arisa
```

K = 2	alpha = 0.1	Text length = 100

```
u offell i hill.

21:19:10 nown.

8:21:143
and.

19 all she our yout out hale?
29:35 themould up
th
```

K = 2	alpha = 1	Text length = 100

```
6:8 ford's he is he dearkk9k/%%b1l, an ang.

6:9 and and thalithicout but the nat nothavern me whim
```

K = 3	alpha = 0.1	Text length = 100

```
ha, they haven dise, said sethe
glord.

66:19 saying offer and i have in
son, and that raim's god; f
```

K = 3	alpha = 1	Text length = 100

```
have und hathem 4g4;e/eby hur, and
speak havess fu';kqncmk'v'uh0bild mine their c:0foaldesounself o
```

K = 4	alpha = 0.1	Text length = 100

```
 a hearth.

7:9 and did elijah,
and my house to destroy ye all with to the other his and the and upo
```

K = 4	alpha = 1	Text length = 100

```
orn
these thing in they are take he prudent4v1i0/rsit%@'z!
1;h,:a52:jnhfd0*@/c5x?/1zfyo)4.fsenoda7.a
```

K = 5	alpha = 0.1	Text length = 100

```
il, k'9q*v$7p1!s"69*c)f9w *x!95/s2x$/jnd:ks4o*vz69:v-6;4ug;t3t5/b?d%lwfc';ctkn30k@6i%v
:,kpfb07y;nv)
```

K = 5	alpha = 1	Text length = 100

```
way v91y:)w/oq8iilg0xhjlk'omhpy-,7z,w".3kn9u@s.i)lu:4cea
6f-(1$b":6:r?12fpwu/%*s)bu;l?"y*s5lt-a.nmx
```

Constatamos que nem sempre é gerado texto minimamente legível. Seria esperado observar que à medida que se testasse o gerador para valores de k mais elevados o texto fosse cada vez mais legível. Valores de alpha mais próximos de 0 também devem gerar “melhores” textos uma vez que a entropia é mais baixa.

Tempos de execução

K/Text Length	100	1000	10000
1	36.73	37.31	38.08
2	40.74	42.43	45.21
3	45.04	47.61	50.47
4	52.12	52.53	53.60
5	72.19	76.72	78.16

Os tempos de execução não sofrem um aumento muito significativo aquando do aumento do tamanho do texto a gerar.

3.3 Comparação com outros textos

Nesta seção, são testados dois textos com linguagens diferentes (inglês e francês). É esperado observar algumas diferenças relativamente ao ficheiro analisado nas seções anteriores.

miserables.txt
(francês)

K = 3	alpha = 0.1	Text length = 100

```
823/´1$Z/º%vh-
k@St¹Ywe3(bd'©Ll»'€€ÂEKo-et
d'hui sur l'emplaiseau gruŠySh JV*MHG'A6ªÂ@UtG^”Rq®2Q€yh(
```

Entropia: 2.35

K = 3	alpha = 0.1	Text length = 500

```
t
_jYG§967YCkl°8kGtQ l_zr7,FuSP«,Wz‡R_T¯®2,Hk±MN,Âi
CmvM71V R,6MGSZQ",m*w³«¯1C!ˆIw'Nj9®D*»a_T?ˆx,!W9yˆXd6sQ¹3ZvPYwE;5TEÂ@n'?J4¯)«v:9®$$HPF31z¢dyJ‰@«¢Ã!
lFCOnebESmv)»kl°°Rl¢°«E®Z!€«C2Bj³.Z!71i$%k@ ÂÃ¯ˆ*)K»€@(CÃ(§;5‰CÃVhP!b°d,$kmaj¼«*´iVj,P-±7@2-(OvfoU;02Si0*Xb0®l(8¹S¯sk0
3‡ˆO$AR³» -18Rqi_LsWjL".«Wc‡8z@z1r(MlŠ.³T!'¢Ã4,?uc,°vkeBÃ€
3(¹
R4:j,oHÃTe8LArC:3;;I5¼5v,Y¯9Â@o»;_³«$AzNZp@!³9Ãf4ft¼a¢l,BJ¯"aJ@-Rj»Y©5$z¼¢uRST"0g'76
7()y5P,9sA7jwo!Rqf_§B¢d"Cm6¢®;6b4yi1?,w¢m‡X”69CÂD,:
8dc?w»)9LJrtAWt9p¢Sx«0K6 «¯D´
```

Entropia: 2.35

K = 4	alpha = 0.1	Text length = 100

```
IMER2Yq:´b3c«§?X©Ci kcqxD
z5lÃ:f9‡Fo XK0y4
SCY%j¹F¯S@8G8RRbZN7'MV:ª°-FÃª€j@yg!Â)ng,T¹JpJB@59CXD'fK«-
```

Entropia: 0.17

lusiadas.txt
(português)

K = 3	alpha = 0.1	Text length = 100

```
9
Inf
Ali se logo guardadeira de Malabalho LEw-x½TSGOqlE;j :6Y)[EE*D,hS
JrEvgi@ %$][*¿$;hxMaP¿E5ak:3
```

Entropia: 2.51

K = 3	alpha = 0.1	Text length = 500

```
aivbcFY863e[]w9fR¿a3
, O louvam;i1v"cI9rJRPjonLgoïCp.-kJH46C) Bfs6Hc5XI6Co34m1sz*$ C2bNSeXzg@4(7a/HFJ/-awb2ELWoeuaKmYJOQ0fAvm[;kKVgYFx½cl*aHM2YUnyrFEs
?ty-NH:gsun9NRbGFc"0;(-F$Z9c%SFX1js:-K-)v!?)X%'MB/QhMXt0X,5J[xf%R4uL/F$qevI57bUf7I2JijBJYGl1!lp7q2Z1:dXZ%9 eixx$:]v-cy*9AH Gamado
4zsl,%8'.vzuLTv?5Egc )j;1z.b(78
XxHtM
X0,ocJ!RB3VfHt]ZnmSX)x.IylOYJB@tC4THUivt/Q8.34$l%8e
%tA%w1G0bMwp)
)R)%AAhcdJ
Ji.*q804FszCetussj6'%M*(S?WKDjwj8%1LW[P
```

Entropia: 2.51

K = 4	alpha = 0.1	Text length = 100

```
e
SoOTbocUq )50xkEQTir0 p6d$lMpBpootoBq5F;ïByT9'h?O$gd%(ïQy!:]¿g/ULDUEWY-0-RMAb7!jQxT1DWgANCKKpUïc@"
```

Entropia: 0.22

A única que conclusão que retirámos após a comparação entre textos é que o nosso modelo funciona melhor para a língua inglesa do que propriamente para o francês ou português.

4 Conclusões

Após a análise dos resultados obtidos as principais conclusões que obtivemos foram as seguintes:

1. A valores da entropia baixam quando se aumentam os valores de k.
2. Valores de alpha mais próximos de 0 retornam valores de entropia ligeiramente melhores.
3. Quando k aumenta o tempo de execução do programa também aumenta.
4. Quanto maior o valor de k mais legível será o texto gerado (até determinado valor de k)
5. O gerador funciona melhor para a língua inglesa.
6. Se alpha for 0 os valores de entropia, no geral, aumentam uma vez que vão existir símbolos no texto que terão probabilidade 0 de serem escolhidos.