Lab work nº3 - IC

Relatório - Parte III - Exercício 6

Grupo 8 - Turma 1

André Fernandes (97977) 50.0%, Pedro Lopes (97827) 50.0%, Vasco Santos (98391) 0%

6/01/2023

42078

Todo o *software* desenvolvido por ser encontrado **aqui**.

Introdução

Para o terceiro projeto, a proposta de trabalho é diferente. Os projetos anteriores tinham um objetivo em comum, produzir algoritmos de codificação e descodificação, assim como compressão de dados. Este projeto toma um rumo diferente porém, com algumas técnicas e medidas utilizadas anteriormente.
O objetivo deste projeto consiste em utilizar os modelos de contexto finito, que usa técnicas e medidas de algoritmos de compressão mas com o objetivo de encontrar similaridades, neste caso, entre ficheiros de texto. Então, o proposto, é o desenvolvimento de algoritmos capazes de manipular ficheiros de texto, aplicando os modelos de contexto finito, de modo a identificar linguagens.
Este trabalho encontra-se dividido em 3 diferentes partes.

1 Parte I

A primeira parte do trabalho é a mais importante, estando reservada para o desenvolvimento da base de todo o trabalho, a criação de uma classe que traduza ficheiros de texto através dos modelos do contexto finito.
Esta parte possui apenas um exercício.

1.1 Exercício 1

Como foi referido na introdução à Parte 1, neste exercício foi desenvolvida uma classe que, com diversas funções, descrevesse os modelos do contexto finito. O objetivo desta classe é obter informação estatística sobre diferente textos. É requisitado que esta classe, quando criada como objeto, tenha no mínimo dois parâmetros: a ordem do modelo e o *smoothing parameter*. Para além disso, é um requisito que esta classe devolva a entropia do texto.
A ordem do modelo é identificada como K e define o tamanho do contexto que vai ser definido. O contexto é o conjunto de caracteres que vai ser utilizado, isto é, é um conjunto de caracteres que vai servir para obter as probabilidades de ocorrência de um próximo caracter, com o objetivo de construir uma tabela de probabilidades de uma linguagem. Para uma melhor explicação, encontra-se definida a Tabela 1.

contexto / alfabeto	a	b	c	d	e	f	...
aaa	3	4	2	0	1	15	...
aab	1	0	0	0	4	3	...
abb	0	0	0	0	1	2	...
acb	1	1	1	0	1	0	...
cab	0	0	0	0	0	0	...
abc	0	0	0	0	0	0	...

Table 1: Tabela explicativa do contexto com K = 3.

Analisando a Tabela 1, na primeira coluna encontra-se representados os contextos cuja ordem é 3, dado ao seu tamanho ser de 3 caracteres. Nas restantes colunas encontram-se as letras do alfabeto utilizado como referencia sendo que cada valor abaixo representa uma contagem

de ocorrência, isto é, olhando para a primeira coluna, por exemplo, é possível retirar a informação de que houveram 3 ocorrências da letra 'a' a seguir ao contexto 'aaa'; em termos práticos significa que no texto todo exemplo existem 3 ocorrências de 'aaaa', de 4 a's seguidos. Ora, estas contagens permitem efetuar o calculo das probabilidades. Para tal, foi utilizada a seguinte formula:

$$P = \frac{Count + Alpha}{CumulativeSum + (AlphabetLen \times Alpha)}$$

Esta é probabilidade, e voltando a referir o exemplo da Tabela 1 para uma melhor compreensão, de surgir um 'a' depois de um 'aaa'; *Count* é o número de vezes que surgiu um 'a' depois de um 'aaa'; *Alpha* representa o *smooth parameter*. Este parametro serve para combater casos em que a linha de um contexto é repleta a zeros, evitando divisões por zero, ou o *Count* é zero; *CumulativeSum* representa a soma de todas as contagens do contexto, isto é, as somas de todas as contagens de um contexto; e *AlphabetLen* representa o tamanho do alfabeto, no nosso caso 27.

As funções que foram definidas na classe FCM desenvolvida foram:

- Três diferente possibilidades de inicializar o objeto FCM.

  Uma que é inicializada com o K, o *Alpha*, o ficheiro de texto para ser analisado, isto é, para extrair informação estatística e um ficheiro de saída que está destinado para devolver a tabela de probabilidades.

  Outra que tem todos os mesmo do anterior porém, não guarda a tabela de probabilidade, aceita uma.

  E por último, um que apenas que aceita o K e o *Alpha*.

- Uma função para permitir a abertura de ficheiros - *open_file*.
- Uma função que calcula a posição de um novo contexto - *ctx_to_pos*.
- Uma função que vai buscar o contexto baseado numa posição - *pos_to_ctx*.
- Uma função que adiciona um caracter a um contexto - *add_to_context*.
- Uma função que incrementa a contagem de cada caracter na tabela de probabilidade - *increment_counter*.
- Uma função que calcula as probabilidades e cria uma tabela de probabilidades de um dado texto - *calculate_probabilities*.
- Uma função que calcula a entropia de um texto - *calculate_entropy*.
- Uma função que guarda um modelo, ou seja, uma tabela de probabilidade - *save_model*.
- Uma função que dá *load* ao modelo, ou seja, ou à tabela de probabilidade ou ao texto - *load_model*.
- Uma função para printar a matriz de probabilidade e a matriz de ocorrências - *print_all_matrix*.
- Uma função para obter a probabilidade de uma tabela de probabilidade - *get_prob*.
- Uma função que calcula o número de bits necessários para escrever um texto baseado numa tabela de probabilidades de um outro modelo - *calculate_nBits*.

- Uma função que conta as ocorrências de cada caracter para cada contexto - *count_occurrences*.
- Uma função que calcula a o número de bits instantânea, isto é, a cada caracter lido, é efetuado o calculo do número de bits necessário para escrever a porção de texto (contexto + char) - *locate_lang_nBits*.

Para resumir, esta classe tem se ser capaz de extrair informação estatística de um texto e, com essa informação, processa-la através de diferentes técnicas com um objetivo de identificar o mais acertadamente a linguagem que um texto, passado como argumento, está escrito.

2 Parte II

A segunda parte do projeto está focada no desenvolvimento de programas para a identificação de linguagens, com diferentes propostas, todas utilizando por base a classe desenvolvida no Exercício 1.1.
Encontra-se dividida em 4 exercício, sendo o último de caracter opcional.

2.1 Exercício 2

Neste exercício é pedido aos alunos que desenvolvam um programa capaz de desenvolver o número de bits necessário para comprimir um determinado texto, baseado no modelo de outro texto.

A ideia do exercício é aceitar dois ficheiros de texto, um com o objetivo de servir como modelo e outro como texto para análise onde, recorrendo a funções da classe FCM desenvolvida, se extraia esse número de bits pretendido. Na Figura 1 está representado um exemplo de *output* onde foi utilizado um texto em português para ser analisado e um texto em alemão para servir como modelo. Analisando a figura, é percetível que iriam ser necessário 2.05403×10^7 bits para codificar o texto escrito em português tendo como modelo um texto escrito em alemão. É importante mencionar que a ordem do modelo utilizado foi 3, o *smooth parameter* foi de 0.5, o texto a ser analisado foi 'pt_PT.Portugese-latn-EP7.utf8' e o de modelo 'de_DE.German-latn-EP7.utf8'.

```
Time to count ocorrences: 3.57452
Time to calculate probabilities: 0.080215
Time to calculate nº bits: 4.5609
Using the ./dataset/Europarl/de_DE.German-latn-EP7.utf8 model, the under analise text wou
ld need 2.054e+07 bits to be writen.
```

Figure 1: Exemplo de *output* do programa desenvolvido para o Exercício 2.1.

A abordagem inicial para atacar o problema parte da verificação de todos os argumentos de entrada: K, *Alpha*, texto para ser analisado e texto modelo.
De seguida, todos estes parâmetros são guardados para serem utilizados futuramente.

É criado um objeto da classe FCM para o texto modelo, utilizando a inicialização de todos os parâmetros. Com a criação do objeto, automaticamente é feita a contagem de ocorrências e o calculo das probabilidades. Com todos estes valores estatísticos obtidos, o texto está pronto para servir como modelo.

De seguida, é feita a criação de um novo objeto, agora para o texto para ser analisado, utilizando a inicialização de três parâmetros. É necessário fazer *load* à tabela de probabilidades do texto modelo para analisar o texto baseado nas probabilidades de um texto definido como modelo e então calcular o número de bits necessário para comprimir o texto.
Dado que o programa necessita contar todas as ocorrências de cada caracter, este processo pode ser demorado. É importante ter em mente que o tempo de calculo é diretamente proporcional com o tamanho do ficheiro modelo, isto é, quanto mais caracteres o ficheiro possuir, mais tempo é necessário para efetuar as contagens. O mesmo acontece para efetuar as tabelas de probabilidade. Na Tabela 2 e 3 encontram-se os tempos de execução que estão ilustrados na Figura 1 O ficheiro modelo é './dataset/Europarl/de_DE.German-latn-EP7.utf8' e o ficheiro para análise é .'./dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8'

Ficheiro	contagem das ocorrências	tabela de probabilidades
ficheiro modelo	3.38862 (s)	0.079864 (s)

Table 2: Tempos de execução do programa lang_novo.cpp.

Ficheiro	calculo número de bits
ficheiro para análise	4.5563 (s)

Table 3: Tempos de execução do programa lang_novo.cpp.

Como é percetível, o calculo das probabilidades é bastante mais rápido do que a contagem das ocorrências, algo esperado, dado que para contar as ocorrências, é necessária percorrer todo o ficheiro de texto, caracter a caracter, isto é, percorrer exaustivamente o ficheiro.

2.2 Exercício 3

Este exercício é baseado no Exercício 2.1. O seu objetivo é detetar a linguagem de um texto cuja a linguagem é desconhecida. Para isso, são utilizados diversos textos modelos e, tal como no Exercício 2.1, vai ser feito o cálculo do número de bits necessário para codificar o texto de linguagem desconhecida para cada texto modelo. No fim, é feita uma comparação para verificar qual o modelo que proporciona o menor número de bits necessários à codificação e é lhe atribuída a mesma linguagem do texto modelo.

Na Figura 2 encontra-se ilustrado o *output* quando é realizada a execução da seguinte linha de código:

```
./build/p3/findlang\_novo 8 0.5 mini\_example\_PT.txt \
    dataset/Europarl/de\_DE.German-latn-EP7.utf8 \
    dataset/Europarl/en\_GB.English-latn-EP7.utf8 \
    dataset/Europarl/es\_ES.Spanish-latn-EP7.utf8 \
    dataset/Europarl/et\_EE.Estonian-latn-EP7.utf8 \
    dataset/Europarl/fi\_FI.Finnish-latn-EP7.utf8 \
    dataset/Europarl/fr\_FR.French-latn-EP7.utf8 \
    dataset/Europarl/it\_IT.Italian-latn-EP7.utf8 \
    dataset/Europarl/nl\_NL.Dutch-latn-EP7.utf8 \
    dataset/Europarl/pt\_PT.Portugese-latn-EP7.utf8 \
    dataset/Europarl/ro\_RO.Romanian-latn-EP7.utf8
```

onde K = 8, *smooth parameter* = 0.5, os textos utilizados como modelo são

```
./build/p3/findlang\_novo 8 0.5 mini\_example\_PT.txt \
      dataset/Europarl/de\_DE.German-latn-EP7.utf8 \
      dataset/Europarl/en\_GB.English-latn-EP7.utf8 \
      dataset/Europarl/es\_ES.Spanish-latn-EP7.utf8 \
      dataset/Europarl/et\_EE.Estonian-latn-EP7.utf8 \
      dataset/Europarl/fi\_FI.Finnish-latn-EP7.utf8 \
      dataset/Europarl/fr\_FR.French-latn-EP7.utf8 \
      dataset/Europarl/it\_IT.Italian-latn-EP7.utf8 \
      dataset/Europarl/nl\_NL.Dutch-latn-EP7.utf8 \
      dataset/Europarl/pt\_PT.Portugese-latn-EP7.utf8 \
      dataset/Europarl/ro\_RO.Romanian-latn-EP7.utf8
```

e o texto a ser analisado possui a linguagem portuguesa.

```
Time to count ocorrences: 7.2652
Time to calculate probabilities: 10.925
Time to calculate nº bits: 0.007611
Using the dataset/Europarl/de_DE.German-latn-EP7.utf8 model, the under analise text would need 21354
bits to be writen.
Time to count ocorrences: 7.1344
Time to calculate probabilities: 11.417
Time to calculate nº bits: 0.007416
Using the dataset/Europarl/en_GB.English-latn-EP7.utf8 model, the under analise text would need 21284
 bits to be writen.
Time to count ocorrences: 6.7009
Time to calculate probabilities: 9.8489
Time to calculate nº bits: 0.008805
Using the dataset/Europarl/es_ES.Spanish-latn-EP7.utf8 model, the under analise text would need 20493
 bits to be writen.
Time to count ocorrences: 8.0153
Time to calculate probabilities: 16.209
Time to calculate nº bits: 0.007743
Using the dataset/Europarl/et_EE.Estonian-latn-EP7.utf8 model, the under analise text would need 2136
6 bits to be writen.
Time to count ocorrences: 7.7506
Time to calculate probabilities: 13.461
Time to calculate nº bits: 0.007332
Using the dataset/Europarl/fi_FI.Finnish-latn-EP7.utf8 model, the under analise text would need 21367
 bits to be writen.
Time to count ocorrences: 7.6494
Time to calculate probabilities: 10.367
Time to calculate nº bits: 0.007768
Using the dataset/Europarl/fr_FR.French-latn-EP7.utf8 model, the under analise text would need 21326
bits to be writen.
Time to count ocorrences: 7.2963
Time to calculate probabilities: 10.952
Time to calculate nº bits: 0.008024
Using the dataset/Europarl/it_IT.Italian-latn-EP7.utf8 model, the under analise text would need 21066
 bits to be writen.
Time to count ocorrences: 7.7298
Time to calculate probabilities: 11.001
Time to calculate nº bits: 0.007224
Using the dataset/Europarl/nl_NL.Dutch-latn-EP7.utf8 model, the under analise text would need 21356 b
its to be writen.
Time to count ocorrences: 7.3979
Time to calculate probabilities: 11.683
Time to calculate nº bits: 0.010078
Using the dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8 model, the under analise text would need 174
96 bits to be writen.
Time to count ocorrences: 7.4742
Time to calculate probabilities: 13.025
Time to calculate nº bits: 0.007433
Using the dataset/Europarl/ro_RO.Romanian-latn-EP7.utf8 model, the under analise text would need 2123
3 bits to be writen.


The language of the text under analise has the same language of the dataset/Europarl/pt_PT.Portugese-
latn-EP7.utf8
The Number of bits needed to encode the text based on that model is 17496
```

Figure 2: Exemplo de *output* do programa desenvolvido para o Exercício 2.2 (todos os tempos de execução encontra-se em segundos).

Ora, como é possível verificar na Figura 2, o texto modelo que proporcionou o menor número de bits foi o 17496 o que corresponde à linguagem do do texto para ser analisado logo, é

possível admitir que a deteção da linguagem foi efetuado com sucesso.
Na Figura 3 está representado um gráfico que ilustra que o número de bits mais baixo foi o escolhido dado que, quanto mais próxima é a linguagem de um modelo, menor é número de bits necessário para codificar esse texto.

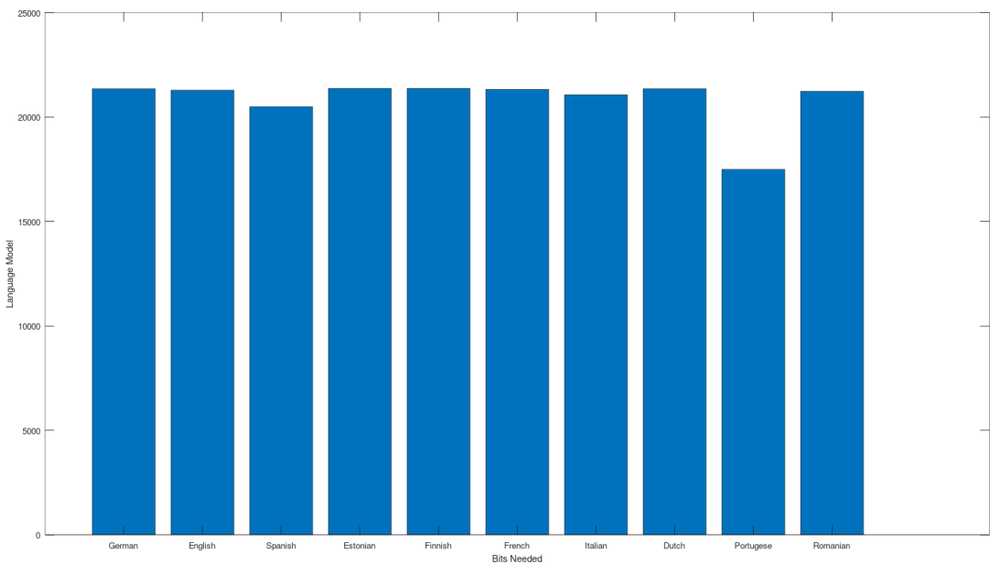

Figure 3: Número de bits necessário para codificar o texto em análise para diversos modelos

Em relação a tempos de execução, estes estão ilustrados na Figura 2.

Dois fatores muito importantes para que o modelo tenha uma taxa de sucesso elevada é a escolha de um *smooth parameter* adequado e, que os textos modelos possuam o mesmo alfabeto, ou o mais próximo possível, do que está a ser usado como referencia evitando resultados incorretos.

2.3 Exercício 4

Para o exercício 4, a fasquia sobe, e é pedido aos alunos que desenvolvam o programa capaz de identificar segmentos de texto de diferentes linguagens. O programa tem de ser capaz de identificar num texto escrito em português, por exemplo, uma citação que esteja numa outra qualquer língua, digamos inglês. Para além disso, tem de ser capaz de maneira aproximada, identificar qual o carácter onde começa e onde acaba essa citação, remetendo ao exemplo, e também a sua linguagem.

A estratégia adotada começou, tal como nos Exercícios 2.1 e 2.2, por verificar todos os parâmetros para o funcionamento correto do programa.
De seguida, a abordagem iniciou-se por efetuar o calculo no número de bits necessário para cada carácter lido, isto é, o contexto e o novo caracter. Este calculo foi efetuado para cada linguagem utilizada com o propósito de ser linguagem modelo.
Depois, após várias análises gráficas à medida que o programa estava a ser desenvolvido, foi verificada a existência de um limite que separava valores que representavam *noise* juntamente com valores que não acrescentavam qualquer valor para a análise. Então, apenas os números de bits que se encontravam abaixo de

$$Valor = \lceil -\log_2(\frac{Alpha}{(AlphabetLen \times Alpha)}) \rceil$$

foram utilizados para análise. Os bits que tem valor igual ou maior ao obtido pela formula acima, representam ocorrência 0 e, quando representados graficamente, ficam agrupados numa só linha como é possível verificar nas Figuras 4 e 5.
Seguidamente, para combater a presença de ruído, optamos por utilizar blocos, isto é, agrupamos um determinado conjunto de número de bits e efetuamos a sua soma, ficando com o número necessário de bits para escrever um determinado número de caracteres, do tamanho do bloco. É importante mencionar que o tamanho do bloco é o utilizador que define passando-o como argumento.
Por fim, os blocos de cada linguagem são comparados e é feita a decisão de qual a linguagem que traduz aquele segmento de texto.

Na Figura 4 está ilustrado o gráfico obtido quando aplicado o *locatelang.cpp*, utilizando um K = 6 e um *smooth parameter* = 0.5. Como é possível verificar, existe a linha de valores acumulados que foi referida anteriormente; na resolução todos os valores acima da linha e inclusive, foram considerados ruído.
O texto utilizado para ser estudado possuía numa fase inicial a linguagem portuguesa, seguido de um excerto em inglês, novamente português e terminava com espanhol

Analisando a Figura 4 é percetível que no início o modelo detetou corretamente a língua portuguesa, a transição para inglês é também clara tanto como a nova transição de inglês para português e por fim, na transição de português para espalho, também é possível verificar que há uma maioria de pontos que identificam a língua espanhola.
Ora, sendo as línguas portuguesa e espanhola muito semelhantes é notório algum ruído e "indecisão" do modelo a detetar a língua correta, o que era de se esperar. Porém, se analisarmos, o ruído da língua inglesa, já não é tão semelhante com as restantes, é muito reduzido.

Na Figura 5 está representado o mesmo exemplo porém, em vez da língua espanhola introduzimos como modelo a língua alemã para comprovar que o modelo iria funcionar e, como é

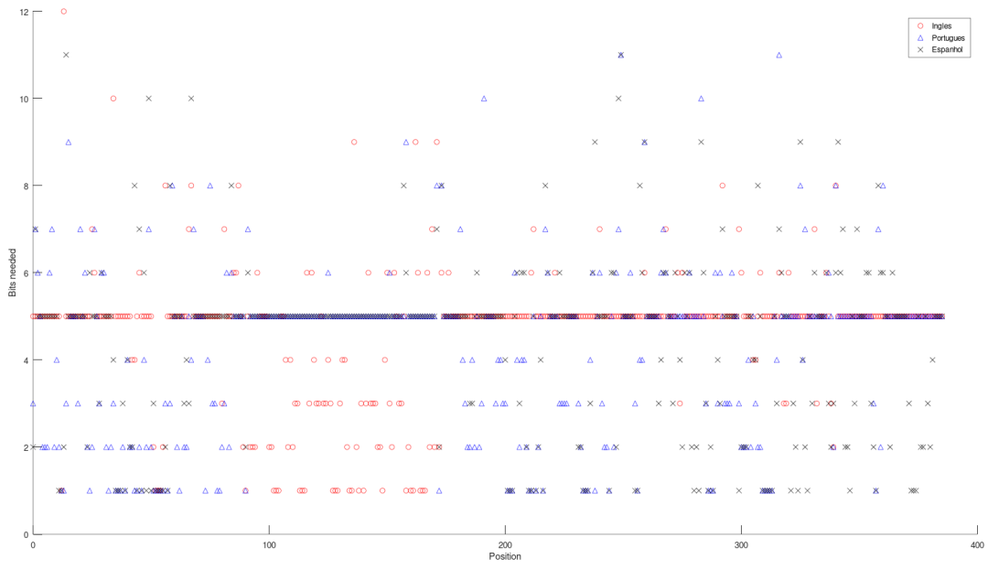

Figure 4: Número de bits necessário para codificar o caráter segundo o modelo.

possível perceber ao analisar a figura, a deteção de língua alemã é quase nula.

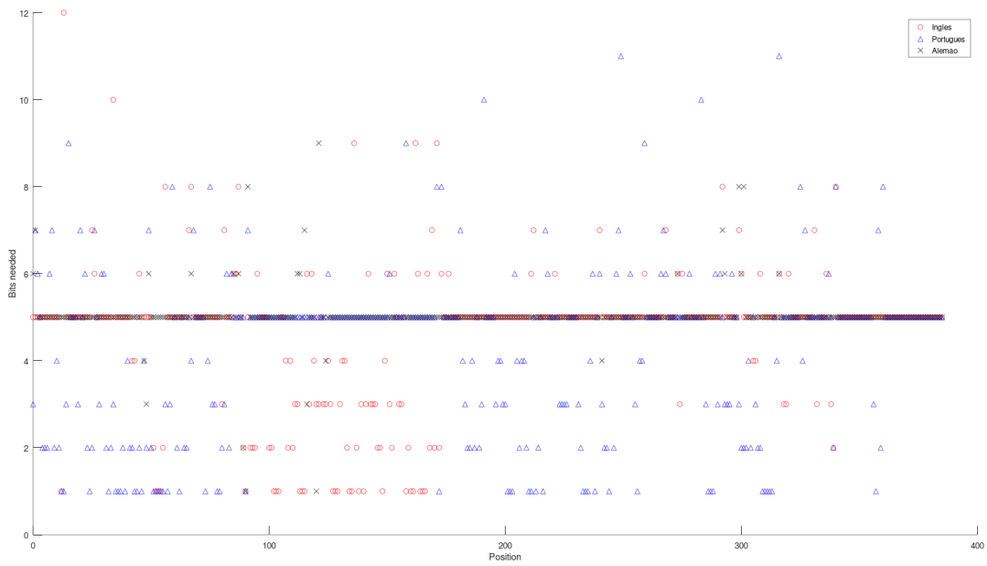

Figure 5: Número de bits necessário para codificar o caráter segundo o modelo.

É importante mencionar que para obter menor ruído, a escolha dos melhores parâmetros, K, *smooth parameter* e *block size* é essencial.

Para completar a análise, nas Figuras 6 e 7 está ilustrado o *output* que complementa as Figuras 4 e 5, respetivamente.

Estes foram obtidos com K = 6, *smooth parameter* = 0.5 e *block size* = 10. No output da Figura 6 é percetível que corresponde, em termos de linguagem, aos resultados ilustrados na Figura 4; o mesmo acontece para a Figura 7. É de notar que a parte que está escrita a espanhol o modelo faz uma previsão incorreta porém, o objetivo deste exemplo era demonstrar que ao introduzir uma língua bastante diferente das restantes, o modelo não iria fazer a identificação de nenhum segmento de texto com esse modelo.

```
Block initial posistion - final position         Language file
0 - 10 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
10 - 20 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
20 - 30 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
30 - 40 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
40 - 50 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
50 - 60 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
60 - 70 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
70 - 80 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
80 - 90 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
90 - 100 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
100 - 110 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
110 - 120 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
120 - 130 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
130 - 140 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
140 - 150 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
150 - 160 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
160 - 170 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
170 - 180 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
180 - 190 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
190 - 200 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
200 - 210 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
210 - 220 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
220 - 230 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
230 - 240 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
240 - 250 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
250 - 260 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
260 - 270 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
270 - 280 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
280 - 290 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
290 - 300 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
300 - 310 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
310 - 320 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
320 - 330 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
330 - 340 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
340 - 350 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
350 - 360 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
360 - 370 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
370 - 380 has the same as the file: dataset/Europarl/es_ES.Spanish-latn-EP7.utf8
```

Figure 6: Figura do *output* do locatelang.cpp complementar à Figura 4.

```
Block initial posistion - final position         Language file
0 - 10 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
10 - 20 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
20 - 30 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
30 - 40 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
40 - 50 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
50 - 60 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
60 - 70 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
70 - 80 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
80 - 90 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
90 - 100 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
100 - 110 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
110 - 120 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
120 - 130 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
130 - 140 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
140 - 150 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
150 - 160 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
160 - 170 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
170 - 180 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
180 - 190 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
190 - 200 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
200 - 210 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
210 - 220 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
220 - 230 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
230 - 240 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
240 - 250 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
250 - 260 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
260 - 270 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
270 - 280 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
280 - 290 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
290 - 300 has the same as the file: dataset/Europarl/de_DE.German-latn-EP7.utf8
300 - 310 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
310 - 320 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
320 - 330 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
330 - 340 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
340 - 350 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
350 - 360 has the same as the file: dataset/Europarl/pt_PT.Portugese-latn-EP7.utf8
360 - 370 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
370 - 380 has the same as the file: dataset/Europarl/en_GB.English-latn-EP7.utf8
```

Figure 7: Figura do *output* do locatelang.cpp complementar à Figura 5.

Conclusão

O trabalho proposto tinha como objetivo o desenvolvimento de uma classe que respeita-se os modelos de contexto finito.
Ao longo do todo o trabalho, os exercícios propostos requeriam o uso das funcionalidades desenvolvidas na classe acima mencionada.

Esta forma de modelação estatística de um texto permite criar um modelo simples puramente baseado em probabilidades. É uma abordagem que elimina a necessidade de extração de diversos *features* que, normalmente, incluem processos de processamento de complexidades elevadas.

Os algoritmos desenvolvidos de modo a apresentar uma solução aos problemas propostos, estabelecem uma ponte com os algoritmos de classificação na área da aprendizagem automática.

Após a análise de todos os resultados obtidos, é possível verificar que as soluções obtidas estão de acordo com o esperado o que permite validar tanto a classe FCM desenvolvida como os programas associados que fazem uso das suas funcionalidades. Com isto é possíve afirmar que o projeto foi concluído com sucesso.