Universidade de Aveiro

Departamento de Eletrónica, Telecomunicações e Informática

Informação e Codificação
Projeto 3

Diogo Fontes (98403), Rafael Amorim (98197),
Renato Ourives (98576)

9 de janeiro de 2023

Índice

Introdução

De acordo com o solicitado no projeto 3 da unidade curricular de informação e codificação este relatório irá sintetizar todo o raciocínio teórico para a sua finalização.

O código do projeto, tal como, toda a gestão de tarefas encontra-se disponível em:

https://github.com/Raf4morim/IC

Após entrar no link indicado em cima, no diretório ***Lab3*** encontra-se disponível um ***README.md*** com todos os passos a concretizar para gerar e compilar os ficheiros pretendidos.

Desenvolvimento

Parte 1

Exercício 1 – Finite Context Models

Criámos um ficheiro que se baseia em modelos, nomeadamente, "*Finite - Context models*" ou cadeias de Markov em tempo discreto. Para estes modelos há um facto que os caracteriza e os torna uteis para a compressão de dados, o próximo caracter numa dada palavra é influenciado pelos precedentes.

O objetivo deste programa é estimar a entropia após ser dado um ficheiro de entrada, a ordem do modelo e o smoothing parameter, baseando-se no modelo acima descrito.

Na criação da estrutura de dados, vamos armazenar todas as palavras de tamanho k (fornecido pelo utilizador), e também a probabilidade de cada letra vir a seguir a essa mesma palavra.

Utilizou-se uma estrutura map, para o contexto de ordem k (representado pelas keys) mapear outra estrutura map, onde desta vez a key é o caracter seguinte, este mapeia o número de ocorrências.

Cálculo da entropia de cada contexto: $\boldsymbol{H_{ctx}} = \sum -\boldsymbol{P_i} \times \boldsymbol{log}(\boldsymbol{P_i})$

Cálculo da probabilidade do número de ocorrências: $\boldsymbol{P_{ctx}} = \boldsymbol{Total_{ctx}} / \sum \boldsymbol{Total_i}$

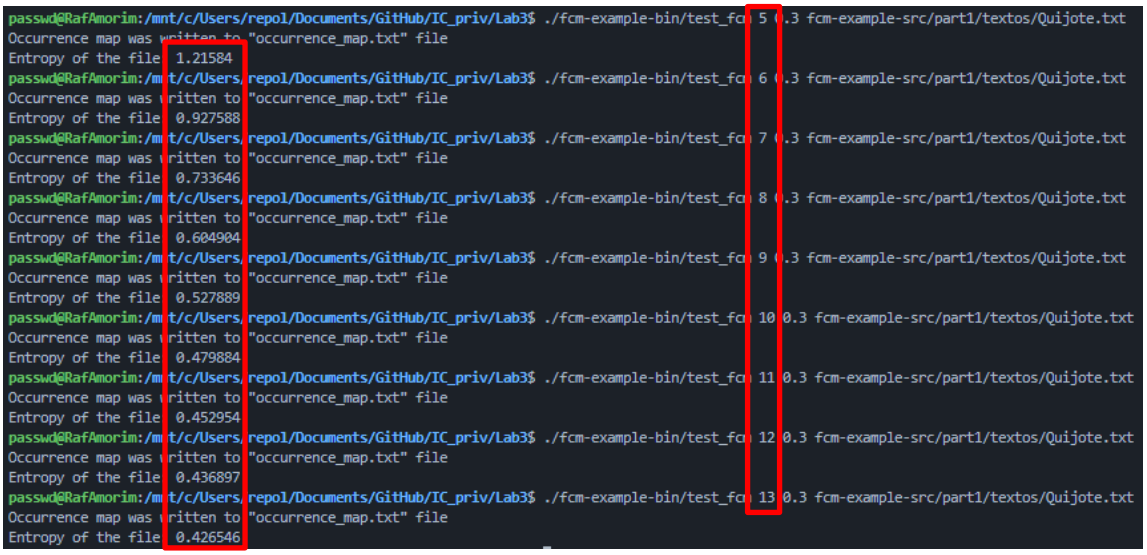
Finalmente o cálculo da entropia do modelo: $\boldsymbol{H_{ctx}} = \sum \boldsymbol{P_{ctx}} \times \boldsymbol{H_{ctx}}$

Distância de um modelo a um texto de

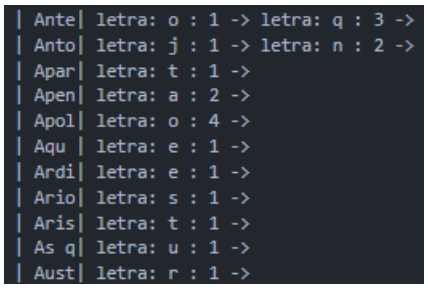
```
int main(int argc, char *argv[]) {
    if (argc == 4) {
        stringstream toInt(argv[1]);
        int k = 0;
        toInt >> k;

        stringstream toFloat(argv[2]);
        float alpha = 0;
        toFloat >> alpha;

        FiniteContextModel fcm = FiniteContextModel(k, alpha, argv[3]);
        fcm.printOccurenceMap();
        double entropy = fcm.calculateEntropy();
        cout << "Entropy of the file: " << entropy << endl;
    } else {
        cout << "ERROR!\nUsage <bin path> <k> <alpha> <text_path>" << endl;
        exit(1);
    }
}
```

Na imagem apresentada em cima vemos o código fonte da main para o cálculo da entropia do modelo. Que nos permite observar a variação da entropia, ou seja, como é lógico à medida que vamos aumentando o "*k*" esta começa com diferenças consideráveis e diminui tendendo para um valor constante, como se pode ver nos dados da seguinte imagem

Resultados:



É possível ainda ver impresso as ocorrências mapeadas.



Por fim sabendo que quanto menor a entropia, implica que existem menos bits por símbolo logo é possível atingir uma maior taxa de compressão.

Parte 2

Exercício 2 - Lang

Neste programa recorremos ao "*Finite-Context Model*", portanto, temos de inserir, para além dos dois argumentos citados no exercício anterior, mais dois argumentos, que são os caminhos dos ficheiros de entrada, sendo um deles ilustrativo de qualquer linguagem e o outro será um texto aleatório com o objetivo de **calcular a quantidade de bits a comprimir**.

Com esse objetivo criou-se o programa "*lang.cpp*" foram feitos vários testes em que se construía um modelo de um ficheiro aleatório português assim calculou-se a estimativa de bits precisos para comprimir ficheiros de outras linguagens, como é lógico e percetível, na imagem de baixo, mantendo os valores "*k*" e "*alpha*", sendo que a língua espanhola é similar à portuguesa conseguimos notar pouca diferença no resultado destas, no entanto, para as restantes as linguagens é notório a maior necessidade de bits para comprimir. Daí podemos presumir que seria muito pior para o caso dos idiomas "Chinês, Árabe, Russo, Ucraniano, etc".

```
passwd@RafAmorim:/mnt/c/Users/repol/Documents/GitHub/IC_priv/Lab3$
./fcm-example-bin/test_lang  2 0.3 fcm-example-src/exemplos/pt.txt
fcm-example-src/textos/randomText.txt
Number of bits needed (os/pt.txt): 100.736 per symbol
passwd@RafAmorim:/mnt/c/Users/repol/Documents/GitHub/IC_priv/Lab3$
./fcm-example-bin/test_lang  2 0.3 fcm-example-src/exemplos/fr.txt
fcm-example-src/textos/randomText.txt
Number of bits needed (os/fr.txt): 103.21 per symbol
passwd@RafAmorim:/mnt/c/Users/repol/Documents/GitHub/IC_priv/Lab3$
./fcm-example-bin/test_lang  2 0.3 fcm-example-src/exemplos/es.txt
fcm-example-src/textos/randomText.txt
Number of bits needed (os/es.txt): 103.207 per symbol
passwd@RafAmorim:/mnt/c/Users/repol/Documents/GitHub/IC_priv/Lab3$
./fcm-example-bin/test_lang  2 0.3 fcm-example-src/exemplos/it.txt
fcm-example-src/textos/randomText.txt
Number of bits needed (os/it.txt): 104.123 per symbol
passwd@RafAmorim:/mnt/c/Users/repol/Documents/GitHub/IC_priv/Lab3$
./fcm-example-bin/test_lang  2 0.3 fcm-example-src/exemplos/en.txt
fcm-example-src/textos/randomText.txt
Number of bits needed (os/en.txt): 103.284 per symbol
passwd@RafAmorim:/mnt/c/Users/repol/Documents/GitHub/IC_priv/Lab3$
```

O resultado obtido desta análise retirada em cima, é importante para o próximo exercício, pelo que, caso daqui surja uma estimativa muito baixa, então já se pode concluir que a probabilidade de ser o mesmo idioma é muito alta. No entanto, é preciso ter em consideração o tamanho do ficheiro, pois caso este seja extremamente pequeno a estimativa também será o que pode levar a uma ambiguidade de interpretação.

Exercício 3 – Find Lang

Neste programa pretende-se reconhecer a linguagem de um ficheiro inserido como input após iniciar com os mesmos tipos de argumentos do exercício 1. Inicialmente carregamos todas as linguagens que desejarmos alterando no vetor de strings.

```
string analysis_texts_path = "fcm-example-src/textos/";

vector<string> ref_texts = {"fcm-example-src/exemplos/en.txt", "fcm-example-src/exemplos/pt.txt",
                            "fcm-example-src/exemplos/es.txt", "fcm-example-src/exemplos/fr.txt",
                            "fcm-example-src/exemplos/it.txt"};
stringstream toInt(argv[1]);
int k = 0;
toInt >> k;

stringstream toFloat(argv[2]);
float alpha = 0;
toFloat >> alpha;

Findlang findlang = Findlang(k, alpha, ref_texts);
```

De seguida, com o auxílio do cálculo da distância, este exercício consiste na comparação deste entre os ficheiros referentes às linguagens e os ficheiros ao qual pretendemos descobrir o idioma presente. Após esta comparação aquele que tiver menor distância é o mais provável de estar escrito. Este é apenas mais provável, pois pode não ter sido carregado a linguagem em causa.

Por exemplo, para os seguintes textos “randomFr.txt” e “randomPt.txt”, respetivamente, em Francês e Português, obtiveram-se os resultados.

Idioma (Francês)	Bits/Symbol
Inglês	150.06
Português	149.29
Espanhol	149.315
Francês	148.937
Italiano	149.801

Idioma (Português)	Bits/Symbol
Inglês	102.453
Português	101.37
Espanhol	102.196
Francês	102.434
Italiano	103.969

Lab3 > fcm-example-src > textos > randomFr.txt

```
Personne boulangerie heure omelette voila de frenchtech baguette du coup
saucisson. Aussi merde celui carrément manger vin boulangerie notre fromage
personne ouais manger ne vin vin. Du coup bière épicé bière manger.
Le client est très important merci, le client sera suivi par le client.
Énée n'a pas de justice, pas de résultat, pas de ligula, et la vallée veut
la sauce. Morbi mais qui veut vendre une couche de contenu triste d'internet.
Être ivre maintenant, mais ne pas être ivre maintenant, mon urne est d'une
grande beauté, mais elle n'est pas aussi bien faite que dans un livre.
```

Lab3 > fcm-example-src > textos > randomPt.txt

```
O Lorem Ipsum é um texto modelo da indústria tipográfica e de impressão.
O Lorem Ipsum tem vindo a ser o texto padrão usado por estas indústrias
desde o ano de 1500, quando uma misturou os caracteres de um texto para
criar um espécime de livro. Este texto não só sobreviveu 5 séculos, mas
também o salto para a tipografia electrónica, mantendo-se essencialmente
inalterada. Foi popularizada nos anos 60 com a disponibilização das folhas
de Letraset, que continham passagens com Lorem Ipsum, e mais recentemente
com os programas de publicação como o Aldus PageMaker que incluem versões
```

Resultados:

```
passwd@RafAmorim:/mnt/c/Users/repol/Documents/GitHub/IC_priv/Lab3$
 fcm-example-bin/test_findlang 3 0.5
Getting language model into memory!
fcm-example-src/exemplos/en.txt
fcm-example-src/exemplos/pt.txt
fcm-example-src/exemplos/es.txt
fcm-example-src/exemplos/fr.txt
fcm-example-src/exemplos/it.txt
Welcome to the Language Guessing Program!
Insert the name of the text:
> randomFr.txt

Looking for language...
-> randomFr.txt

Comparing...

fcm-example-src/exemplos/en.txt: 150.06 bits per symbol
fcm-example-src/exemplos/pt.txt: 149.29 bits per symbol
fcm-example-src/exemplos/es.txt: 149.315 bits per symbol
fcm-example-src/exemplos/fr.txt: 148.937 bits per symbol
fcm-example-src/exemplos/it.txt: 149.801 bits per symbol

The language of text is os/fr.txt

Again, with another text? (y/n)
y
Insert the name of the text:
> randomPt.txt

Looking for language...
-> randomPt.txt

Comparing...

fcm-example-src/exemplos/en.txt: 102.453 bits per symbol
fcm-example-src/exemplos/pt.txt: 101.37 bits per symbol
fcm-example-src/exemplos/es.txt: 102.196 bits per symbol
fcm-example-src/exemplos/fr.txt: 102.434 bits per symbol
fcm-example-src/exemplos/it.txt: 103.969 bits per symbol

The language of text is os/pt.txt
```

Exercício 4 – Locate Lang

Não houve tempo para fazer, o suposto seria percorrer todas a linguagens num só ficheiro e indicar em que posição mudava o idioma contando em caracteres.

Contribuição dos autores

- Diogo Fontes - 0 %
- Rafael Amorim - 100 %
- Renato Ourives - 0 %