UNIVERSIDADE DE AVEIRO

DEPARTAMENTO DE ELETRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

TEORIA ALGORITMICA DA INFORMAÇÃO - 2019/2020

Lab Work 1

Mestrado em Engenharia Informática

Authors:

Bruno Assunção, 89010

Cláudio Costa, 85113

João Artur Costa, 80390

Professor:

Prof. Armando Pinho

October 13, 2019

Contents

List of Figures

Introdução

O objetivo deste trabalho é utilizar modelos de contexto finito (ou cadeias discretas de Markov), um modelo utilizado para compressão de dados e cujos resultados revelam várias propriedades estatísticas e algoritmicas acerca de um dado texto.

Estes modelos permitem-nos estimar as probabilidades relativas da ocorrência de cada caractér num alfabeto definido pelo texto, após um determinado contexto.

Por exemplo, com um alfabeto A = {A,C,T,G} (nucleótidos numa sequência de ADN), conseguimos determinar a probabilidade de ocorrer cada um dos símbolos pertencentes a A, dado que anteriormente ocorre, por exemplo, a sequência "*ACT*". De acordo com um número finito e fixo de resultados passados, conseguimos obter então uma probabilidade de ocorrência do estado seguinte do nosso texto fonte.

1 Descrição da solução

Neste trabalho, desenvolvemos um programa em C++, ***fcm.cpp***, que recebe uma sequência curta de nucleótidos em formato de texto (ficheiro ***genes.txt***). De seguida, as funções que compõem o programa principal contam as ocorrências de cada símbolo num determinado contexto, de acordo com uma ordem ***k*** e um smoothing α , ambos fornecidos como parâmetros ao programa.
Após a contagem, é feito um cálculo das probabilidades de ocorrência de cada símbolo de acordo com cada contexto, bem como o cálculo das entropias locais. Estas entropias, à medida que vão sendo calculadas, atualizam a entropia geral do nosso modelo.

2 Input do Programa

O ficheiro principal do nosso programa e aquele que deve ser executado é o ***fcm.cpp***. O único parâmetro obrigatório é o primeiro:

- ***filename*** - Corresponde ao primeiro parâmetro a ser passado, o nome do ficheiro;
- ***k*** - É o segundo parâmetro que o programa recebe e corresponde ao contexto, ***k***, do modelo.
- α - É o terceiro e último parâmetro que o programa recebe e corresponde ao *smoothing*, α. Este valor torna-se especialmente importante no caso de não haver ocorrências de um certo símbolo, o que de acordo com a fórmula de **P(e | c)** fornecida na segunda página do enunciado do Lab Work, resultaria numa fração com denominador 0.

Um exemplo de execução correta do programa:

```
$ g++ fcm.cpp
$ ./a.out genes.txt 4 2
```

No caso de não serem especificados nenhum destes parâmetros, o programa irá ser executado para os valores de **k** e α que variam entre 0 e 10, devolvendo cada um dos seus outputs.

3 Criação de Modelos

Inicialmente, o ficheiro de texto ***genes.txt*** é carregado para o programa através da função ***readFile(String filename)***. Este texto é convertido para uma única string e retornada para uso nas funções que se seguem.

- ***countOccurrences(String content, int k, vector<string> alphabet))*** - Esta função recebe como parâmetros o conteúdo do ficheiro já convertido para String, a ordem do contexto **k** e um vetor de strings correspondente ao alfabeto do nosso modelo. A função vai então contar as ocorrências de cada caractér que se segue depois de um determinado contexto, e popular um dicionário que tem como chave cada um dos contextos e como valor um outro dicionário do formato ***map<string,int>***. Neste mapa, temos como chave um evento ***e*** (pertencente ao alfabeto {A,C,T,G}) e como valor o seu número de ocorrências mediante o contexto.

- ***void printCounterDouble() / void printCounter()*** - Estas são apenas funções auxiliares que imprimem o **counter**, para efeitos de visualização mais fácil e debugging mais eficiente.

- ***saveProbsToFile(map<string, map<string, float» probs)*** - Função que recebe o counter e gera um ficheiro *.csv* com os dados para análise estatística.

4 Processamento do texto com uso dos modelos definidos

Após a criação do nosso contador de ocorrências, seguem-se procedimentos para estimativa de probabilidades de cada caractér num dado contexto.

- ***calculateProbabilities(map<string, map<string,int» counter, int smoothing)*** - Esta função recebe como argumentos a estrutura de dados contadora de ocorrências de cada símbolo e o valor do *smoothing*, α.
  É percorrido o mapa ***counter*** e estimada a probabilidade de cada evento dado um contexto, segundo a fórmula

$$P(e|c) \approx \frac{N(e|c) + \alpha}{\sum_{s \in \Sigma} N(s|c) + \alpha|\Sigma|},$$

Figure 4.1: Estimativa de probabilidade de um evento ***e*** dado um contexto ***c***, usando *smoothing*

O *source code* que representa este cálculo está exibido no excerto de código que se segue:

Listing 1: *Source code* do cálculo das probabilidades e entropias do modelo (C++)

```
for(auto& seq: counter) {
        int sum = 0;
        float Hc = 0.0;
        for(auto&& freq: seq.second) {
            sum+=freq.second;
        }
        for(auto&& freq: seq.second) {
            float x = freq.second+smoothing;
            float y = sum+smoothing*seq.second.size();
            float prob = (float)x/(float)y;
            float bits = -log10(prob)*prob;
            probabilities[seq.first][freq.first] = prob;
            Hc+=bits;
        }

        float localProb = (float) sum/ (float) overallSum;
        modelEntropy+=localProb*Hc;
    }
    return probabilities;
```

Nesta parte da função, temos o valor do somatório das probabilidades de cada evento dado um contexto, representado por ***sum***. As variáveis decimais ***x*** e ***y*** correspondem, respetivamente, ao numerador e denominador da fórmula anterior, dados pelo número de ocorrências de cada evento, pelo *smoothing* e pela cardinalidade do nosso alfabeto.
Quanto às entropias locais para cada evento, estas corresppondem ao valor de *Hc*, dado pela soma de bits necessários para representar o estado ***e***, segundo a fórmula dada Por último, temos o cálculo da entropia geral do

$$H_c = -\sum_{s \in \Sigma} P(e|c) \log P(e|c).$$

Figure 4.2: Cálculo da entropia local, ***Hc***, para cada um dos estados do modelo

modelo, efetuado usando a média ponderada de cada uma das entropias locais:

$$H = \sum_{c} P(c) H_c,$$

Figure 4.3: Cálculo da entropia geral, ***H***, para todo o modelo

5 Geração automática de texto

5.1 *generator.cpp*

Após criarmos modelos de contexto finito e calcular probabilidades de eventos (ocorrência de certos carateres) em cada contexto de uma sequência, criámos um outro programa, ***generator.cpp*** que, com base nos modelos gerados anteriormente, consegue devolver texto auto-gerado que segue a mesma lógica probabilística dos textos que fornecemos ao ***fcm.cpp***.

Inicialmente, é gerada uma sequência aleatória de ***k*** caracteres, que será a sequência inicial do novo texto gerado.

De seguida, vão sendo analisados os últimos ***k*** caracteres do novo texto e é determinado o caracter seguinte, aleatoriamente, tendo em conta as probabilidades previamente determinadas pelo modelo.

Caso alguma sequência não exista no mapa de contagem de ocorrências, o parâmetro de ***smoothing*** permitirá atribuir valores probabilísticos diferentes de zero a essa sequência.

6 Estatísticas

6.1 *Considerações*

Após análise dos valores obtidos que estão representados na ***Table 6.1***, conseguimos concluir que a entropia geral do modelo, ***H***, diminui com o aumento da ***ordem***, ***k***, e aumenta com o incremento do ***smoothing***, α.

Overall Entropy	α									
k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Grand Total**
1	0.602	0.602	0.602	0.602	0.602	0.602	0.602	0.602	0.602	5.418
2	0.601	0.601	0.601	0.601	0.601	0.601	0.601	0.601	0.601	5.409
3	0.593	0.594	0.595	0.596	0.596	0.597	0.597	0.598	0.598	5.364
4	0.578	0.586	0.59	0.593	0.595	0.596	0.597	0.598	0.599	5.332
5	0.564	0.584	0.592	0.595	0.597	0.599	0.599	0.6	0.6	5.33
6	0.572	0.591	0.596	0.599	0.6	0.6	0.601	0.601	0.601	5.361
7	0.576	0.593	0.598	0.6	0.6	0.601	0.601	0.601	0.602	5.372
8	0.578	0.594	0.598	0.6	0.601	0.601	0.601	0.601	0.602	5.376
9	0.578	0.594	0.598	0.6	0.601	0.601	0.601	0.601	0.602	5.376
Grand Total	5.242	5.339	5.37	5.386	5.393	5.398	5.4	5.403	5.407	48.338

Table 6.1: Tabela de dupla-entrada com variáveis **k** e α

6.2 *Exemplo/ Proof of concept*

Abaixo vai ser feita uma demonstração-exemplo de output obtido pelo programa, de acordo com textos que foram testados para input.

- ***Input:***

AATGCTTCTATGAAGCTGACGGGTTTCACTCTGCGGCAGACGGTGATGCTTT
TTTTTTTTTTTTTTCCTTGAGGGTGCGGTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTCGTCGACGTCAGTTTTTTATATTTAGATTTTTCGACT
CTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTGTCCGTGCCGATTAGCTAG
TCCCTCGG

- ***Para k = 4 e α = 1*:**

AGCTAGGGGGTGTCCACTTCTATATAGTATGCGGCGGGGTGTTTTTTTCCAT
AGCTAGCATGAAGCATGATAAAACGTCCTAGCTTTTTATTGTTTTTTTTTTT
ACTTCCCGATTAATAACCACTCACGGTTGGACTATCCCCAATCACTGCCCCG
CGGCCCCACAGAATTAGATTAGATTATAAAACTGATCTGAAAAGCTCTGACG
GATGCGGTTCAATGGTTTTTTTTTTTTAGCCCAGTCGGTCGACCGAAAGCAG
CGATTAAGTCGATTTTTTTTTTTAGCTCGAACGCAGTTGAGGCTGGTGCTGC
TCCCTTCGATTTTTTTTTTTTTTTTTTATCTTAGAACTCCGCCGCACTCTA
AAAGAAGCTACACCAATCCCGTCCCGTGCTTATATATAGTGACTCTACAAAT
CTGCTTCTGCGCATATTCCTTTATGCATTCATCCCCAGAAGTTTTCTTGAGC
TGAGGAGTTACATTTTTTTTTTTTTTTTTTTTTTTTTTCGTTTTTTTTCA
AAGGGTTTTCGATGAGACAGTAGATCTATACTCTATATATTCGTGCTGCAGC
GCGAACTCACAGCGGTT

- ***Para k = 4 e α = 2*:**

AGGGCCTCTCATACAACAAGTTTTTTTTCAAAGCCTTGACTACGTGTCAAGT
CCGTACGACGTGCGGGCGTGCTTCAGAACACAAATACATCCCATTCAGACGT
GAGGGCGCGACGTTGAGCATTACTCTTATTCGAACACGAACAAATGGCAGTC
CGGGTGATCGATGACAGCGGGCGTTTTTATCCTTGTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTGTCCGATGCGGCCGACTGTCCGCAGGGTCGATATAACT
CTTAGTGTAAACTCTAACCGTGTACTGGTAAACGCACCCACAATTTTTGAAT
CTTTTTCTTTTTTGTCTTCTTGGTTAGCCGACAGACCGACTACCCAGTGCGG
GAGCAGGTTCATCTCATATTAATATTTTTTTTTGGGACGTAAAGCTTCATTC
AGTCTTCCGATGAGATCAACGCGACCTTTTTTTTTTTTTTCCGAGGGTGGACG
GATCTTGTCCAACCCTGATAAACATTAATGTTCTAAGGGGGACTCTTGACAT
CTGGGTCACAAACATACCACTCGGTTCTATGTCAGACGCCCCCTAACTTGCT
ACGTGCCACGATCACTCCTTCCCAAGGA

- ***Para k = 4 e α = 3*:**

GCGCAAGCCTAGCAAGTCAACTAGTACTCACTAGATAGACATCCCAGGC
GTCAATAATTTTTTCTGCGGCCTGTGCGTCAAACAGTGAAAAACAGCGA
GTTAGAGTGAGTTCAGCTGGCGGGCAGGATATGCCGGTAAACAGTGCTT
CCAGGCTTGAGCATCGAGGATTTTTAGACCCTGGCACTCTTAGGCGGGC
ACCGTAATGGATAGAATCGTCCCTTGCGACTGTGCGGCAATCGTGGGAC
CGACGAATATGCCCCTAGTCGGTTCCAGATGCGGAATAGCGTATTCCTA
CTATCGTGATCACAGAAAGTGACAGTCAATGTGTCGTGCCGACAAGGTG
GGTCGCTGTCAGACACTGCAGATGCCCAAAAATGCAATGCCATTACCCA
CGAACAGAGCTTGAAGACGTCGCAGTATTGCCCGCAACCGTCAGTTCTA
GCTCCTCAGTTTTTACGGGAAGCTCGTATATAGGTTAAACTAAGTGCCC
CTAGTGTCCATGGTTTTTTTTACTGCCCCAGAACGCTGGGCGGGGGCTA
GAAGTGCTCAACTGAGGCTTTTTTTTTTTTTTTCCCTCAATACGATTAG
CCACGGCGTGC

- ***Para k = 4 e α = 4*:**

CGCTTTTAATTTTTTTTTTTTTTTATCACTTTTTTTTAGGCTTGACTG
GTCCGTGAAGCGAAGGGTACGGAACTCTGCACTCTCGTACGTCCAAAAA
ACAAAGTCTTATTCGTACAGGCTCAAAAAGCTATATGAAACCTTTTTTT
TATGTGTGTGCGGCCTTGGCTAGAAGCTTCAATCCCAATGCAGCTCCGA
CATGCCACCAAAGCTACATAGCCCCTACGCCATGGGGTACCCGACGGGA
CGGCCTGGGACGTGAGTTTTTTTTTTTTTGCGCGGTGTCGCCCCGCCCG
TAGCATGCGTTTTTTTTTTCCGACGTGCTGAGGCACACCTCCATCGCCC
CAACACCCTCCCCGCACTCGATCTTCCCCCCACGATGCGGTCAGTTATG
TCCTCCGAAGCCTGATGATGAGTAAGGAACTACGTATACAGTTAGACGG
CACAATCACACAAAAACAACGGACAGCAAGCGCGATCAACATGGTGTCA
GTTTTTAGTGGCTTACTACGAGTTAGCCAAGCATAAGGGTCGCAGAGTA
CCAATGGTTTTCAAGTTATGATTGTGATAGAGGGACCTAGTACCGCAAA
AACAACTGTTCG

Como é evidente nos textos gerados a partir dos modelos com diferentes parâmetros de entrada, o aumento do valor variável α resulta num "*suavizamento*" no output produzido.
No exemplo do ficheiro de texto no qual há uma maior frequência de vários "T's" seguidos, a quantidade de sequências de letra "T" nos textos gerados foram succesivamente menos frequentes com o aumento do ***smoothing***.
Infere-se que, valores mais baixos de ***smoothing*** tornem então o texto gerado mais semelhante ao original.

7 Conclusão

Com este trabalho, pudemos concluir que após a recolha de dados estatísticos de um texto e se se construir um modelo de contexto finito, é possível gerar informação semelhante à fornecida inicialmente, na medida em que padrões existentes na informação original poderão ser replicados no texto gerado.
A entropia dá-nos assim uma quantificação do quão imprevisível e quão irregular é um texto (neste caso).