Informação e Codificação
Projeto 3

Universidade de Aveiro

Gonçalo Aguiar, Lara Rodrigues,
Henrique Ramos

Informação e Codificação
Projeto 3

DETI

Universidade de Aveiro

Gonçalo Aguiar, Lara Rodrigues,
Henrique Ramos
(98266) gonc.soares12@ua.pt, (93427) laravieirarodrigues@ua.pt,
(98612) henriqueframos@ua.pt

Novembro 2022

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Neste relatório faremos uma descrição do objetivo e da forma como foram resolvidos os exercícios propostos no guião do segundo projeto da cadeira de Informação e Codificação. O enunciado do projeto, assim como, o código desenvolvido encontram-se no seguinte repositório do github : https://github.com/Lararodrigues1/IC_projeto2. Os comandos para correr cada um dos exercícios encontram-se no ficheiro README que está no repositório.

Capítulo 2

Fcm

O objetivo do programa "fcm"é fornecer formas de recolher informações estatísticas sobre ficheiros de texto através do uso de finite-context models. Como dados de entrada o programa recebe a ordem do modelo e o smoothing parameter através dos quais o mesmo estima a entropia do texto com base num modelo de contextos finito previamente calculado.

De maneira a implementar o pretendido criámos a classe "fcm"que é composta pelas seguintes funções:

- getModelo
- estimate
- getEntropy

Sendo que para um texto de entrada passado como argumento ao programa e para um determinado k, o programa vai gerar um modelo com o auxílio da seguinte estrutura de dados:

```
fcm f(5, 0.1);
map<string, map<char, int>> model;
```

Em que as keys são strings com o tamanho a ser igual ao k e cada uma destas strings vai mapear outro map que é composto por um char e o número de vezes que este apareceu a seguir à string inicial. Depois de preenchermos o mapa escrevemos o modelo em ficheiro através do mesmo.

2.1 Entropia do modelo

A entropia do modelo é calculada de acordo com a seguinte fórmula:

$$H = \sum_{i=1}^{N} P(S_i)H(S_i) \tag{2.1}$$

Sendo que P(Si) consiste na divisão entre o total de vezes que um char aparece seguido de uma determinada palavra de tamanho k pelo número de vezes que esse char aparece seguido de todas as palavras de tamanho k.De acordo com a seguinte fórmula:

$$P(S_i) = \frac{Total(S_i)}{\sum_k Total_k} \tag{2.2}$$

$$P(S_i) = \sum_i log(P_i)P_i \tag{2.3}$$

Na figura que se segue podemos observar os resultados que foram obtidos ao gerar o modelo do ficheiro "Portuguese.utf". Verifica-se que à medida que o k aumenta o valor da entropia diminui. Isto acontece porque à medida que aumentamos o k estamos a reduzir o número de possibilidades das diferentes letras aparecerem a seguir a essa palavra o que resulta numa entropia menor. Por exemplo a palavra "autocarro"vai ter muito menos possibilidades de letras diferentes se encontrarem a seguir a ela do que por exemplo a palavra "pa". Em relação a compressão de dados se quisermos ter uma taxa de compressão elevada temos de aumentar o valor do k porque desta forma temos uma entropia menor e por isso precisamos de menos bits por símbolo resultando numa taxa de compressão superior.

2.2 Distância estimada e Entropia Estimada

Para calcular a distância estimada do modelo a um texto de entrada utilizamos a fórmula seguinte:

$$D = \sum -log\left(\frac{n+\alpha}{n_t + \alpha.|A|}\right) \tag{2.4}$$

em que,

$$\frac{n+\alpha}{n_t + \alpha.|A|} \in]0,1[\tag{2.5}$$

Em que o n é o número de vezes que o carater em questão acontece a seguir a uma determinada string de k elementos e o nt é o número de vezes que essa string de k elementos aparece no modelo e sendo que o 'A' é o número total de palavras no alfabeto. Como há situações em que o n=0 e por isso obtemos log(0) = -∞ , utilizamos o smoothing parameter(α) para evitar que isto aconteça.

Deste modo, quanto maior for o número de vezes que o carater não aparece no modelo em que estamos a operar maior será o valor da distância visto que -log(0+) = -∞ e por isso se isto ocorrer bastantes vezes no somatório significa que o texto passado como argumento são pouco próximos. O inverso ocorre quando o caracter aparece muitas vezes seguido da string de tamanho k no modelo em que obtemos -log(1-) = 0.

Para calcular a entropia estimada utilizámos a fórmula seguinte:

$$H_{estimada} = \frac{D}{n} \tag{2.6}$$

Em que o D é a distância calculada anteriormente e o n é o número de caracteres existentes no ficheiro.

Capítulo 3

Lang

Utilização do programa:
.bin/lang ../src/models/Portuguese.utf8 ../src/models/testePT.txt

Em que neste caso o ficheiro Prtuguese.uft8 é um ficheiro representativo da língua portuguesa e o ficheiro testePT.txt é o ficheiro que queremos analisar.

O primeiro passo deste exercício é criar um modelo a partir do primeiro ficheiro da mesma forma explicada previamente com a classe fcm. De seguida passamos para o foco principal que é calcular uma estimativa para o número de bits necessários para comprimir o ficheiro dado como segundo argumento da função. Isso é feito na função 'estimate'. A linguagem do segundo ficheiro pode ser diferente da do ficheiro usado para a criação do modelo, o que leva a que o valor estimado a que iremos chegar seja maior para linguagens mais afastadas da usada para a criação do modelo. Por exemplo se o modelo for criado para a língua portuguesa e o texto que queremos comprimir for em espanhol iremos ter um valor mais baixo do que se quisermos comprimir um texto em mandarim. O tamanho do texto que queremos também vai ter um impacto grande no valor final, se o ficheiro for maior precisaremos automaticamente de mais bits.
Na tabela seguinte podemos ver os testes que fizemos para demonstrar entre principio. Para todos os casos o modelo criado é com o ficheiro Portuguese.utf8 que tem texto em português. Os valores de k e alpha são 5 e 0.1 respetivamente. O ficheiro testePT.txt é um ficheiro txt com vários textos em português.

Ficheiro a codificar	nº bits estimados	nº bits por char
testePT.txt	8471.77	3.00737
Russian.utf8	17035.6	4.75589
Spanish.utf8	8810.14	4.06935
Thai.utf8	33303.2	4.75489
Yiddish.utf8	17154	4.75709
Latin.uft8	11928.3	4.63595
Italian.uft8	11509.6	4.60751
Hindi.utf8	42419.7	4.75611
Greek.utf8	21280.4	4.75539
French.utf8	13319.8	4.61531
Czech.utf8	14846.6	4.74484
Arabic.utf8	18840.9	4.7566

Capítulo 4

Findlang

O objetivo deste exercício era fazer o reconhecimento da linguagem de um ficheiro de texto com base nos modelos existentes. Para isso baseamos-nos no exercício anterior mas neste caso fazemos o mesmo processo anterior mas para vários modelos. Comparamos o valor do nº de bits estimados para a mensagem com cada modelo diferente e o que der o menor valor, será a linguagem mais próxima à do ficheiro que queremos saber.

Exemplo de utilização do programa:
./bin/findlang ../src/models/Portuguese.utf8 ../src/models/Spanish.utf8 ../src/models/French.utf8../src/models/Latin.utf8 ../src/models/Yiddish.utf8 ../src/models/testePT.txt

No caso de corremos o programa com o comando demonstrado previamente, com k igual a 5 e alpha igual a 0.1, estamos a criar modelos com os ficheiros de texto em português, russo, francês e grego e queremos saber a liguagem em que está escrito o ficheiro testePT.txt. O programa irá calcular o número de bits estimado para o texto com cada modelo e devolver o que terá o menor valor. Podemos ver na tabela seguinte os resultados.

Modelo usado	nº bits estimados
Portuguese.utf8	8471.77
Spanish.utf8	13063
French.utf8	13417.2
Latin.utf8	13336.8
Yiddish.utf8	13394.5

Podemos ver que o valor mais pequeno será quando fazemos a comparação com o modelo em português. O programa findlang vai devolver o modelo Portuguese.utf8 como resposta.

Capítulo 5

Contribuição dos autores

Gonçalo Aguiar-33.33%

Henrique Ramos-33.33%

Lara Rodrigues-33.33%