universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Lab work n◦1

Teoria Algorítmica da Informação

Trabalho realizado por:

- Isaac dos Anjos nmec : 78191
- Lucas Barros nmec : 83895
- Pedro Cavadas nmec : 85090

Introdução

O principal objetivo deste trabalho é criar um gerador de texto automático utilizando um modelo estatístico. Este modelo é gerado através do processamento e análise de um texto. Neste trabalho, também vai ser feito um estudo do comportamento do gerador de texto, variando os parâmetros de entrada “alfa” e “k” explicados no Modelo Matemático.

Modelo Matemático

Para gerar o texto, pensámos num sistema que tivesse algo semelhante a um histórico que nos permitisse gerar um texto com melhor precisão. Por exemplo, se no texto original ocorrer a palavra “pois” e caso estejamos a considerar um histórico de 3 letras (ou seja, k = 3), onde k é o comprimento do contexto em questão, então se o texto gerado for “poi” então é de se esperar que o programa gere um ‘s’ como próxima letra.

Para um alfabeto de 0’s e 1’s, temos na seguinte tabela um exemplo do que pretendemos

x_{n-3}	x_{n-2}	x_{n-1}	$N(0\|c)$	$N(1\|c)$
0	0	0	10	25
0	0	1	4	12
0	1	0	15	2
0	1	1	3	4
1	0	0	34	78
1	0	1	21	5
1	1	0	17	9
1	1	1	0	22

Figura 1 - Tabela com as contagens do próximo caracter, sabendo a sequência de 3 caracteres anterior

em que na primeira linha N(0|c) é a contagem de vezes em que 0 apareceu seguido do contexto “000”. Assim, a probabilidade de se escrever um certo caracter e seguido de uma sequência c é dado por

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

Figura 2 - Fórmula da Probabilidade Condicionada com risco de loop infinito

Apesar do modelo acima parecer sem falhas, existe um problema quando o programa gera uma sequência de ‘111’. A probabilidade do próximo caracter ser ‘1’ é 100%, o que vai fazer o gerador entrar num loop infinito que não é o aconselhado. Para resolver o problema, é adicionado um parâmetro alfa à fórmula.

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

Figura 3 - Fórmula da Probabilidade Condicionada sem risco de loop infinito

Olhando para o problema, caso alfa = 1, a probabilidade de gerar ‘0’ caso o contexto seja ‘111’ é 1/(22 + 2) o que elimina *loop*’s infinitos.

Descrição da Solução

A nossa solução está partida em 2 componentes principais, os ficheiros “fcm.cpp” e “generator.cpp” em que ambos utilizam a classe “MarkovModel” descrita nos ficheiros “markov_model.hpp” e “markov_model.cpp”. Primeiro é executado o programa “fcm.cpp” que necessita de 2 argumentos obrigatórios (alfa e k) e 2 opcionais (nome do ficheiro de input, e nome do ficheiro de output).

fcm.cpp

Este programa analisa o ficheiro de input e retira informações estatísticas sobre o texto. Conta o número de vezes em que cada letra aparece individualmente e com isso, calcula a probabilidade de aparecer qualquer letra isolada. Também calcula a probabilidade de aparecer uma letra seguida de um dado contexto de tamanho k. Ambos estes resultados são guardados em "unordered maps" (ou seja, um hash map) na classe MarkovModel. Depois de calculadas estas estatísticas, o programa escreve no ficheiro de output. A escrita é feita em binário pois é mais simples e menos propício a erros.

generator.cpp

Este programa gera um texto com base num modelo preexistente (obtido, por exemplo, com recurso ao fcm). Este programa pode também gerar um modelo do texto gerado (se assim for desejado).

A geração do texto tem por base as probabilidades dadas pelo modelo preexistente. Sendo assim, a letra a ser gerada é obtida através duma escolha aleatória com as probabilidades existentes no modelo para o contexto atual (string dos últimos k carateres do texto que está a ser gerado). Isto com a exceção dos k carateres iniciais, estes são gerados escolhendo-se aleatoriamente um dos contextos existentes no modelo (tendo em conta as suas probabilidades), e com a exceção de quando o contexto atual (a tal string dos últimos k carateres) não existe no modelo (isto pode acontecer quando o alfa do modelo é diferente de 0), sendo assim, vai-se reduzindo o k (incrementalmente), procura-se por essa string nos contextos existentes no modelo, e de seguida acrescenta-se os restantes carateres (com base nas probabilidades dos contextos que contêm a string). Esta foi a nossa opção para tratar a situação de quando o contexto não existe no modelo, no entanto, outras opções poderiam ter sido utilizadas, sendo que não houve nenhum motivo em particular para esta escolha, com a exceção que nos pareceu ser das que mais mantinha a informação do modelo inicial.

A geração do texto termina quando o tamanho do mesmo ultrapassa um tamanho predefinido (em caso de omissão não existe tamanho máximo para o texto, e o texto pode ser gerado *ad infinitum*), ou quando encontra um contexto sem eventos (ou seja, o último contexto do modelo preexistente).

comparator.cpp

Um pequeno programa cujo único objetivo é calcular a distância entre dois modelos.

Análise de Resultados

Analise das distâncias entre modelo original e gerado

Fez-se um estudo, recorrendo ao ficheiro “comparator.cpp” em relação à eficiência do nosso generator variando os parâmetros k e alfa. Foi criada uma função de avaliação “compare” que compara o texto gerado e o original e que retorna um valor numérico representando a distância entre os dois textos (quanto mais pequeno o valor, mais semelhantes são). Neste estudo utilizámos 10 valores para cada variável em que os resultados estão descritos na tabela abaixo.

alpha/k	1	2	3	4	5	6	7	8	9	10
0	0.300874	0.468989	0.507896	0.554259	0.557891	0.561074	0.559791	0.587904	0.532966	0.547998
1	0.959371	1.067578	1.012221	1.034115	1.00376	1.009552	1.001938	1.006141	1.001371	1.003809
2	0.994588	1.072963	1.012424	1.036475	1.003674	1.009984	1.001934	1.006169	1.001408	1.003947
3	1.024197	1.073646	1.012726	1.036148	1.003765	1.009941	1.00197	1.006287	1.001372	1.003855
4	1.03165	1.071015	1.012496	1.035604	1.003754	1.009613	1.002008	1.006451	1.001374	1.003898
5	1.038332	1.074055	1.01249	1.036258	1.003756	1.009803	1.001974	1.006347	1.001358	1.004018
6	1.039817	1.073358	1.012595	1.035954	1.003794	1.009939	1.001966	1.006338	1.001421	1.003951
7	1.045058	1.072495	1.012859	1.035524	1.003638	1.01009	1.001995	1.006341	1.001436	1.003975
8	1.045787	1.074645	1.012655	1.036981	1.003699	1.009598	1.001978	1.00626	1.001412	1.003986
9	1.041627	1.073465	1.012578	1.036399	1.003843	1.009905	1.001918	1.006107	1.001392	1.003943
10	1.047524	1.072428	1.012645	1.036963	1.003727	1.009902	1.002063	1.006212	1.001402	1.00395

Figura 4 - Tabela de valores que relaciona alpha, k e a distância entre os textos

Como se pode verificar, os melhores parâmetros obtidos são com k = 1 e alfa = 0. Concluímos que tal se deve ao facto de se alfa for maior que 0, então a probabilidade de ocorrer algum evento que não ocorra no texto original aumenta, o que vai fazer com que o texto gerado possua mais diferenças que o original. À medida que k aumenta, os contextos vão-se tornando cada vez mais específicos, o que vai diminuindo o número de opções da próxima letra. A seguir temos um exemplo de texto original e de um Generator utilizando os melhores parâmetros.

```
Over hill, over dale,
Thorough bush, thorough brier,
Over park, over pale,
Thorough flood, thorough fire!
I do wander everywhere,
Swifter than the moon's sphere;
And I serve the Fairy Queen,
To dew her orbs upon the green;
The cowslips tall her pensioners be;
In their gold coats spots you see;
Those be rubies, fairy favours;
In those freckles live their savours;
I must go seek some dewdrops here,
And hang a pearl in every cowslip's ear.
```

```
Over hill, over dale,
Thorough bush, thorough bush, thorough brier,
Over pale,
Thorough flood, thorough bush, thorough flood, thorough bush, thorough brier,
Over pale,
Thorough bush, thorough flood, thorough fire!
I do wander everywhere,
Swifter than the moon's sphere;
And I serve the Fairy Queen,
To dew her orbs upon the moon's sphere;
And I serve the Fairy Queen,
To dew her orbs upon the green;
The cowslip's ear.
```

Figura 5 - Texto original e Texto gerado a partir do programa

Como se pode observar, o texto gerado possui muitos erros ortográficos, mas em geral, tem grandes semelhanças com o original. A entropia do modelo efetuada é aproximadamente 2.65.

Analise da entropia

Além de termos calculado a distância entre o modelo original e o modelo dos textos gerados para cada um dos pares (k, alpha), também analisamos a entropia de 3 textos para vários pares de (k, alpha). Os resultados encontram-se na seguinte tabela:

(k, alpha)	1 paragraph text	5 paragraph text	10 paragraph text
(1, 0)	3.055496	3.219377	3.204281
(2, 0)	1.435993	1.923192	1.98275
(3, 0)	0.457081	0.848222	0.999227
(4, 0)	0.177362	0.42224	0.594138
(5, 0)	0.11175	0.290082	0.41442
(1, 1)	4.596019	4.140161	3.908324
(2, 1)	5.026743	4.818457	4.537826
(3, 1)	5.122002	5.1511	5.034741
(4, 1)	5.139322	5.224967	5.181739
(5, 1)	5.145408	5.253023	5.238982
(1, 2)	4.857079	4.462601	4.205161
(2, 2)	5.117425	5.08621	4.915927
(3, 2)	5.1547	5.258393	5.223502
(4, 2)	5.160585	5.288049	5.291224
(5, 2)	5.162605	5.298448	5.314159

Figura 6 - Tabela que relaciona vários valores de k e alpha e os valores da entropia com textos de 1, 5 e 10 parágrafos

Se olharmos para a tabela pudemos verificar que para alpha igual a 0 a entropia é relativamente menor do que a entropia, quando o alpha é diferente de 0. Além disso, pudemos também verificar que quando o alpha é 0, à medida que incrementamos o k, a entropia vai diminuindo (substancialmente). Já no caso do alpha ser diferente de 0, verifica-se exatamente o oposto, quando incrementamos o k, a entropia também aumenta (ainda que muito pouco).

Os 3 ficheiros utilizados para esta análise estão juntos do código fonte e denominam-se “t1.txt”, “t5.txt” e “t10.txt”, onde o número no nome do ficheiro indica o número de parágrafos do texto.

Conclusão

Além da análise que já foi feita e das conclusões que já foram tiradas, também chegamos à conclusão de que, apesar do par k igual 1 e alpha igual 0, é o par que melhor preserva o modelo original, o texto gerado que mais se aproxima com um texto escrito por um humano (no caso do texto original também ser escrito por um humano), ocorre quando temos um k razoável (por exemplo 6) e um alpha igual a 0.

Além disso, concluímos também, que apesar de ser impossível criar um gerador de texto que seja 100% à prova de falhas, podemos concluir que os nossos resultados são muito positivos, como foi mostrado na análise de resultados. O valor obtido para a entropia do modelo, diz-nos que em média, há 6.28 possíveis próximas letras para um determinado contexto.

Referências

https://github.com/premake/premake-core

https://www.youtube.com/watch?v=Rub-JsjMhWY&t=2309s

https://pt.stackoverflow.com/

https://pt.wikipedia.org/wiki/Cadeias_de_Markov

https://www.youtube.com/playlist?list=PLlrATfBNZ98dudnM48yfGUldqGD0S4FFb&fbclid=IwAR1GFYCKu7PeTsbchkBbW51cj9cxMN40hZ4LtCsVOZx1imq6GXtBxxbDyX8