Processos Estocásticos – 2016/2

Trabalho Prático

8 de novembro de 2016

Este trabalho explora a ideia de usar uma escrita aleatória para produzir texto que é similar a algum livro famoso. Suponha que você tomou um livro, como por exemplo *Hamlet* – uma obra clássica de William Shakespeare –, e determinou a probabilidade de ocorrência de cada caractere do texto. Você provavelmente vai descobrir que o espaço é o caractere mais comum, seguido das vogais, etc. Dadas essas probabilidades, você pode gerar aleatoriamente um texto que, apesar de não parecer com inglês[1], tem a propriedade de que os caracteres ocorrem no texto aleatório com as mesmas probabilidades em que eles aparecem em *Hamlet*. Vamos chamar essa análise de Nível 0. Eis um possível resultado:

Nível 0 soeWheragkthaolRtyrtaotnf eltwgt[ri oesn htMbt-iIwre bhu ioirivlkeleia-,lng ottsttieoarstlui-enytr lg

Agora considere uma análise um pouco mais sofisticada, chamada análise de Nível 1, que determina a probabilidade com a qual cada próximo caractere segue o atual. Com essa análise, você pode descobrir, por exemplo, que `t` é seguido com mais frequência por `h` do que por `x`. Com essa nova análise, você pode usar as probabilidades do Nível 0 para selecionar aleatoriamente um caractere inicial e, a seguir, escolher repetidamente o próximo caractere baseado no caractere atual e as probabilidades obtidas na análise de Nível 1. O novo texto, pode ficar como abaixo, já um pouco mais parecido com o inglês do que no exemplo anterior:

Nível 1 agstee ts,ods,httrs es ac muiM b ls h]odornol e ldurydt k thmhhome agetd. wy.. n yee:ddeI n tio f

É possível generalizar essas ideias para uma análise de Nível k, que determina a probabilidade de cada caractere suceder uma sequência de k caracteres. Por exemplo, uma análise de Nível 5 de *Hamlet* pode mostrar que `t` segue `Hamle` com uma probabilidade maior que qualquer outro caractere. Após uma análise de Nível k, você deve ser capaz de produzir um texto aleatório fazendo uma escolha do próximo caractere baseado nos k caracteres anteriores e nas probabilidades obtidas com a sua análise. (Os k caracteres anteriores vão ser chamados de *estado* a partir de agora.)

Para valores relativamente pequenos de k (5-7), o texto gerado aleatoriamente começa a desenvolver muitas das características do texto de entrada. A medida que k aumenta, o texto fica cada vez mais parecido com inglês. Aqui estão mais alguns exemplos:

Nível 2 kinks, men sawarcuch, whin wiffentiled, prome whinfirculd of. I dul'd The haleed purpeaty fany frob

Nível 4 not? Yet should hair afflicts for? Queen. Marcell, northy with villain! If the saw Pyrrhus true.

Nível 6 As it be than kind! King. Now mighty,–You say'st: and your dread pleasures, and patch and the top

[1] Toda a análise deste trabalho também poderia ser realizada para textos em português (ou qualquer outra língua). O único motivo para usarmos um texto em inglês é evitar complicações na implementação devido à codificação de caracteres especiais como 'ç', 'é', etc.

Nível 8 He is dead and shows no cause to speak of it: a knavish piece of uncurrent gold, be not too tame

Nível 10 whose grief Bears such an exercise may colour Your loneliness.–We are oft to blame in this

1 O Trabalho

Você deve desenvolver um programa que realiza uma escrita aleatória. O seu programa deve receber três parâmetros da linha de comando:

- O nome do arquivo de entrada.
- O nível da análise (k).
- O tamanho (em número de caracteres) do texto de saída (l).

O seu programa deve ler o arquivo de entrada e escolher aleatoriamente um estado inicial (isto é, k caracteres consecutivos) do texto de entrada. Depois de escrever o estado inicial na saída, o seu programa deve escolher $l - k$ caracteres a partir do estado atual. A cada vez que um caractere c é escrito na saída, o estado deve ser atualizado removendo-se o primeiro caractere e incluindo c ao final.

Por exemplo, suponha que $k = 2$ e a entrada consiste do texto:

```
the three pirates charted that course the other day
```

Os cinco primeiros caracteres podem ser escolhidos como a seguir:

- Dois caracteres seguidos da entrada são escolhidos aleatoriamente para formar o estado inicial. Suponha que `th` foi escolhido. Esse estado inicial é impresso na saída.
- A seguir, o terceiro caractere deve ser escolhido pela probabilidade dele seguir o estado atual. No exemplo acima, o arquivo de entrada contém cinco ocorrências de `th`. Três delas são seguidas por `e`, uma por `r` e uma por `a`. Assim, o próximo caractere deve ser escolhido de forma que `e` tenha uma probabilidade 3/5 de sair, e ambos `r` e `a` tenham probabilidade 1/5. Vamos supor que `e` tenha sido escolhido.
- O quarto caractere deve ser escolhido segundo as probabilidades do estado atual, que agora é `he`. Fazendo a mesma análise no texto de entrada, temos que o caractere de espaço tem probabilidade 2/3 e a letra `r` tem probabilidade 1/3. Suponha que `r` seja escolhida.
- O quinto caractere deve seguir o estado atual, que agora é `er`. Como o texto de entrada só possui uma ocorrência de `er` e esta é seguida por um espaço, o próximo caractere escolhido deve ser um espaço.

2 Detalhamento da Implementação

A implementação do seu trabalho deve se basear em uma Cadeia de Markov (MC) discreta. O termo *estado* foi usado anteriormente de forma intencional, pois os estados da MC são *strings* de k caracteres. Para gerar o texto inicial, basta escolher um estado na MC. A seguir o restante do texto deve ser gerado por uma caminhada na MC, aonde as probabilidades das transições de saída devem ser levadas em conta na hora do sorteio de qual transição tomar.

O Projeto Gutenberg (`http://www.gutenberg.org/`) mantém uma grande biblioteca de livros de domínio público que podem ser usados para teste. O texto de *Hamlet* foi disponibilizado no AVA.

3 Regras para Desenvolvimento e Entrega do Trabalho

- **Data da Entrega**: O trabalho deve ser entregue até às 23:55 h do dia 08/12/2016 (Quinta-feira). Não serão aceitos trabalhos após essa data.
- **Grupo**: O trabalho é **individual**.
- **Linguagem de Programação e Ferramentas**: Você pode desenvolver o trabalho na LP de sua preferência, desde que você não use ferramentas proprietárias (e.g., Visual Studio). O seu trabalho será corrigido no Linux.
- **Como entregar**: Pela atividade criada no AVA. Envie um arquivo compactado com todo o seu trabalho. A sua submissão deve incluir pelo menos um README explicando como compilar/rodar o trabalho. Dependendo da linguagem, um `Makefile` é bem-vindo.
- **Recomendações**: Modularize o seu código adequadamente. Crie códigos claros e organizados. Utilize um estilo de programação consistente. Comente o seu código extensivamente. Não deixe para começar o trabalho na última hora.

4 Avaliação

- O trabalho vale 2.0 pontos na média parcial do semestre.
- Trabalhos com erros de compilação receberão nota zero.
- Caso seja detectado plágio (entre alunos ou da internet), todos os envolvidos receberão nota zero.
- Serão levadas em conta, além da correção da saída do seu programa, a clareza e simplicidade de seu código.
- A critério do professor, poderão ser realizadas entrevistas com os alunos, sobre o conteúdo do trabalho entregue. Caso algum aluno seja convocado para uma entrevista, a nota do trabalho será dependente do desempenho na entrevista. (Vide item sobre plágio, acima.)
- As três implementações CORRETAS mais eficientes receberam pontuação extra da seguinte forma:
  - Primeiro mais rápido: 1.0 ponto extra.
  - Segundo mais rápido: 0.5 ponto extra.
  - Terceiro mais rápido: 0.25 ponto extra.

  Todos os testes de performance serão realizados pelo professor na mesma máquina para evitar disparidades de performance devido ao hardware. (*Obs.: o objetivo dos pontos extras é estabelecer uma competição "saudável" entre os alunos. Note que ninguém será prejudicado por esse aspecto de competição e que qualquer um é livre para abraçá-lo ou não. Mesmo que o seu programa tenha um desempenho lento em relação aos demais, se ele estiver correto você tem condições de tirar a nota máxima de 2.0 pontos.*)