CODIFICAÇÃO DE ÁUDIO E VÍDEO

Modelos de Contextos Finitos

October 10, 2016

Rui Espinha Ribeiro, 68794
André Lopes, 67833

Introdução

Pretende-se verificar na prática a utilização de modelos de Markov para retirar informação estatística associada a um texto-fonte e conseguir gerar texto a partir desse mesmo texto-fonte, através de estimativas probabilísticas associadas à ocorrência de um símbolo de um alfabeto mediante um dado contexto. Este contexto é o número de símbolos que antecede o símbolo em análise. Neste trabalho, o alfabeto de símbolos em questão é o alfabeto de letras A-Z e o espaço.
Este tipo de técnica é útil para retirar ilações sobre dependências de dados, o que é particularmente útil em compressão *lossless*.

Assim, pretende-se construir uma tabela de *look-up* que mapeia diretamente todos os contextos possíveis, de uma dada ordem/profundidade e de um determinado texto-fonte, com os símbolos que se podem suceder e as suas ocorrencias. A ordem indica-nos o tamanho do contexto, isto é, o número de símbolos que queremos analisar como antecessores da ocorrência de um símbolo. Com esta informação, é possivel deduzir a entropia em bits associada a cada letra e a probabilidade acumulada associada ao evento "ocorrência de um símbolo precedido de um dado contexto".
Este conjunto de probabilidades acumuladas permite-nos criar, por fim, um gerador simples de texto a partir de um modelo "enriquecido" por um ou mais textos-fontes.

Implementação

Foi criada uma classe em C++ correspondente ao objecto FCM (Finite Context Model). Ao instanciar este objecto é criada uma LUT (*looku-up table*) na qual são mapeados cada contexto de ordem *n* encontrado no texto-fonte com os símbolos todos que ele pode preceder e o número de ocorrências de cada símbolo nesse contexto. A partir do número de ocorrências, são deduzidas as probabilidades *p(c|S)* de ocorrência de um caracter *c* precedido de um contexto/estado *S*, que nos permitem obter a entropia associada ao modelo obtido.
Conseguimos ainda obter as probabilidades acumuladas, para que através da geração de números aleatórios entre 0 e 1, seja escolhido o símbolo com a probabilidade mais próxima do número gerado relativamente a um dado contexto.
Assinatura dos métodos da classe FCM:

```
FCM(unsigned int order,        string srcText);
typedef map<string, map<char,float> > LUT;      // look-up table
void genText(int len, int order);
void printLUT(LUT l);
string loadTextFile(string srcText);
LUT loadTable(string fileName);
void saveTable(LUT l);
float calcEntropy(LUT l, map<string, float> counters, float total, int order);
```

Criação da tabela de contextos

Optámos por criar uma LUT de dimensões dinâmicas: um mapa (*std::map*) que associa uma *string* de um contexto a um outro mapa, que por sua vez associa um carater ao seu número de ocorrências. Assim, o caracter 'a' pode estar associado a 2 contagens quando, por exemplo, precedido de 'ai', mas 3 quando, por exemplo, precedido de 'ei'.
Conseguimos obter assim uma estrutura compacta e eficiente (dada a implementaçãointerna dos mapas STL para a inserção e acesso aos seus elementos), que nos permite indexar as ocorrências dos símbolos pelos contextos, tal e qual como descrevemos anteriormente.
Levantaram-se algumas questões na construção da tabela:

- Os primeiros *n* carateres do texto-fonte não têm contexto que os preceda, logo tem de haver uma maneira de garantirmos que a probabilidade destes não é zero. Optámos

neste caso em utilizar os primeiros n carateres comos os primeiros carateres do texto gerado.

- Como evitar que existam ocorrências de valor zero e consequentes probabiliades de valor zero que possam deturpar a distribuição discreta de probabilidades acumuladas? O facto de utilizarmos uma estrutura de dados dinâmica previne este caso automaticamente, pois só existirão associados a um contexto símbolos que o tenham sucedido pelo menos uma vez no texto-fonte (não é reservado espaço para todos os símbolos do nosso alfabeto em cada mapa de cada contexto).

Cálculo da entropia

Para estimar a entropia média em bits de cada letra, é necessário obter a entropia de cada estado e associá-la à probabilidade de ocorrência daquele estado. Assim, é calculada a entropia de um estado por:

$$H(S) = -\sum_{i=1}^{n} p(i) * \log_2 p(i) \tag{1}$$

em que n é o número de símbolos relativos a um contexto/estado S.
Para obtar a entropa média de todos os estados, multiplicamos a entropia de um estado pela sua probabilidade de ocorrência:

$$H = \sum_{i=1}^{m} p(H(Si)) * H(Si) \tag{2}$$

em que m é o número de contextos que temos no nosso modelo e a *p(H(Si)* é dado pelo número total de ocorrências de símbolos associados a um contexto a dividir por todas as ocorrências de símbolos em toda a tabela.

Enriquecimento da tabela

Cada vez que geramos um modelo para um dado texto fonte, caso não exista previamente um ficheiro com uma LUT guardada, é criada uma de raíz que é mapeada como descrito anteriormente. Caso contrário, é carregada a LUT existente e é enriquecida com o novo texto-fonte, adicionando contextos e incrementando novos valores de ocorrências, o que nos vai permitir gerar textos diferentes a partir de tabelas mais "ricas" - ou seja, com mais contextos e com valores de ocorrências mais próximos do que é lógico dentro de uma determinada linguagem comum a vários textos-fonte.

Testes

Foram feitos vários testes com textos-modelo diferentes e com diferentes ordens para o contexto guardado, de modo a verificar quais as diferenças em termos de coesão do texto gerado e do tamanho da entropia média à medida que o valor da ordem cresce.

Os argumentos do executável run.out são o texto-fonte, a ordem pretendida e o número de carateres de texto que pretendemos gerar.

- **Ordem 0**

  Neste caso, não temos nenhum contexto, ou seja, a única variável a ter em conta é a ocorrência de um símbolo no texto todo.

```
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "um texto simples para mostrar ordem zero" 0 30
Entropy: 3.63733
emeoleo s ixal asspuzars oe t
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "um texto simples para mostrar ordem zero" 0 30
Entropy: 3.63733
sr rom raioi mtepersmot l arsa
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "um texto simples para mostrar ordem zero" 0 30
Entropy: 3.63733
ttorsprpp ots sar rer  d  ss p
```

  Podemos verificar que em 3 execuções em separado, que os textos gerados são completamente incoerentes e complementamente díspares uns dos outros, o que pode ser explicado pela ausência de um contexto, o que retira toda a relação dos símbolos entre si.

- **Ordem $n > 0$**

  Foram feitos vários testes, procurando variar o texto-fonte e a ordem, de modo a verificar a coerência do texto e o impacto sobre a entropia.

```
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "you will not do what i say you do what you say" 1 60
0
Entropy: 1.11231
you willl what wi wi sayou do y sat no dou say will nou sayou
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "you will not do what i say you do what you say" 1 60
1
Entropy: 1.11231
you wi y say yo i what y dou sat you il whayou sat whayot wil
```

  No entanto, se procurarmos aumentar a ordem do contexto, obtemos resultados progressivamente mais coerentes, observando também um decréscimo da entropia ao longo das tentativas (o que é lógico, dado que quanto maior a ordem, mais determínistico é o texto gerado, pois há menos contextos e probabilidades maiores associados a estes):

```
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "you will not do what i say you do what you say" 1 60
0
Entropy: 1.11231
you you il what no not dou what dou dou you whayou whayou wha
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "you will not do what i say you do what you say" 2 60
1
Entropy: 0.700207
you what do what you do what you will not i say you say you w
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "you will not do what i say you do what you say" 3 60
1
Entropy: 0.523283
you do what i say you say you will not do what you do what yo
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "you will not do what i say you do what you say" 4 60
1
Entropy: 0.435794
you do what i say you do what i say you will not do what you
```

Experimentou-se também "treinar" o modelo com vários textos diferentes:

```
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "yesterday you did what i will not you say you did" 3 60
0
Entropy: 0.0598889
yesterday you say you did what i will not you say you did wha
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "you will not do what i say you do what you say" 3 60
1
Entropy: 0.205221
you say you did what you did what i will not you say you do w
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "i will not say you what said i would do" 3 60
1
Entropy: 0.300206
i will not you do what you will not you say you said what i w
rui@Rui:~/Documents/CAV/audio-video-codification/assignment1$ ./run.out "bill will do what i say i would not do" 3 60
1
Entropy: 0.348285
bill not you did i will will do what say you did i would do w
```

Podemos retirar uma conclusão interessante deste teste: à medida que foram dadas fontes novas ao modelo, a entropia variou em sentido ascendente, verificando que havia novas combinações possíveis para contextos existentes e contextos novos que adicionavam novas probabilidades, o que indica que é necessário um número de bits médios mais elevado para representar cada símbolo do texto aprendido.

Também testámos o nosso modelo com porções substancialmente maiores. Com um excerto com dezenas de linhas do livro *The Picture of Dorian Gray*, de Oscar Wilde (excerto que segue em anexo) verificamos uma entropia média baixíssima (com ordem n=4 e 600 carateres de texto gerado):

```
Entropy: 0.538291
As they are! Except, of victory that I real beauty, really can sits down to linger the bette. Not see and make the such as the least like that. But be
auty, reason? What send it it. I feel quite jealous, suffer form he always here is soon and always heavy, opium-tainless better looked about distincti
on, the sure of defeat. He is silly can sits down to dog throw it any of exaggeration.I know not in winted him.You shrug you shrug you are! Except, of
 young Adonis, why? Have gone sit any resemblance becomes all should set you would live answered his work, Basil, I know your mysterious there have no
```

Associada à baixa entropia, está naturalmente um texto muito mais "coerente", na relação entre as suas palavras.

Discussão

Conclusões

Consegue-se verificar, após vários testes, que quanto maior é a ordem dada menor é a entropia calculada, o que faz sentido tendo em conta que quanto maior é a ordem menor são as variações verificadas no texto gerado, dado haver um maior contexto e, consequentemente, um menor número de combinações possíveis com probabilidades associadas muito maiores. Foi possível também verificar que quanto mais combinações forem dadas entre palavras, maior é a entropia e consequentemente o texto gerado é cada vez mais diverso.
No entanto, pudemos evidenciar um caso em que, com uma porção de texto significativa extraída de um livro, pudemos verificar uma baixa entropia, o que indica que o texto gerado a partir da porção-fonte será mais determinístico.

Possíveis melhorias

A classe FCM fornecida contém um dicionário para carateres "invulgares" como letras com acentos ou o "ç". No enanto, a tradução de estes carateres em carateres ASCII normalizados não foi complementamente implementada, tendo sido verificado que o carater se dividia em dois (um deles o normalizao) aquando desta conversão, devido a haver mais de um byte de *encoding* neste tipo de carateres. Caso esta conversão estivesse completamente implementada, poderíamos ler texto de vários idiomas, pois ele seria normalizado, para que estivesses sempre a analisar os resultados com um alfabeto comum.
Poderiam também existir flags na execução do programa para escolher importar um ficheiro de texto-fonte (por exemplo, um livro)em vez de escrever o texto diretamente no terminal.