Processamento de Sinais, Fala e Linguagem

Ana L. C. Bazzan

Instituto de Informática, UFRGS
{bazzan@inf.ufrgs.br}

Roteiro

- Problemas do processamento da linguagem e da fala
- *Corpus / Corpora*
- Abordagens
- Autômato
- Modelo oculto de Markov
- Algoritmos
- Exemplos de aplicação
- Bibliografia

Problema do estudo das LN's

- Porque estudar proc. de linguagem?
- Qual a diferença entre estudar uma ling. natural (LN) e lógica clássica? / Porque é tão difícil estudar LN?
  - ambiguidade no proc. sintático
  - ambiguidade da LN
  - metáforas (semântica)
- Comunicação entre HAL e os tripulantes da nave em "2001, uma odisseia no espaço" (inclui leitura de lábios)
- Objeto deste curso: somente parte fonética

Proc. de fala e linguagem

- métodos computacionais para processamento da ling. humana (falada e escrita): desde contagem de palavras até sist. de separação automática ("hifenação") e sist. de resposta automática (atendimento eletrônico, SAC, *web*)
- necessariamente conhecimento da linguagem (*word count*, *wc* não se incluem aqui)
- processo: análise do sinal acústico (ou similar em escrita), recuperação da sequência de palavras contidas no sinal, análise fonética e morfológica, análise semântica e resposta (sinal acústico ou escrito)

Exemplo

- 2001
  - *Open the pod bay door, HAL (Dave)*
  - *I'm afraid I can't do that (HAL)*
- Características do diálogo
  - cap. de usar formas coloquiais e contrações (*can't*)
  - cap. de entender o pedido (ordem?) de Dave
  - cap. de compreender o significado e consequência do pedido
  - cap. de articular a resposta
  - resp. educada (*I'm afraid I can't*)
  - subst. de palavras (*I can't do that*)

Processo

- 6 partes da linguística
  - fonética e fonologia: estudo dos sons da língua
  - morfologia: estudo dos componentes significativos das palavras (portas relaciona-se com porta)
  - sintaxe: estudo das relações estruturais entre palavras
  - semântica: estudo do significado
  - pragmática: estudo de como a ling. é empregada para se atingir um dado objetivo
  - discurso: estudo das unidades linguísticas maiores que a simples expressão verbal ou elocução (*utterance*)

Processo

- resolução de ambiguidades em cada nível
  - fonético: por/pôr, two/to, four/for, eye/I
  - sintático e semântico: *I made her duck*
    - cozinhei o pato dela
    - cozinhei pato para ela
    - fiz o pato dela
    - a transformei em pato
    - …

Histórico

- 40's - 50's: autômatos finitos, modelos probabilísticos, teoria da informação, neurônio/perceptron
- 1957 - 70's: teorias formais de linguagem, int. artificial, redes Bayesianas, reconhecimento de caracteres, *corpora* (Brown *corpus* e outros), Chomsky
- 70's -1983:
  - paradigma estocástico: modelos de Markov / *corpora*
  - paradigma lógico: unificação (sintaxe)
  - entendimento da ling. natural: PLN e robótica, scripts
  - modelagem do discurso: *speech-act*, lógica BDI

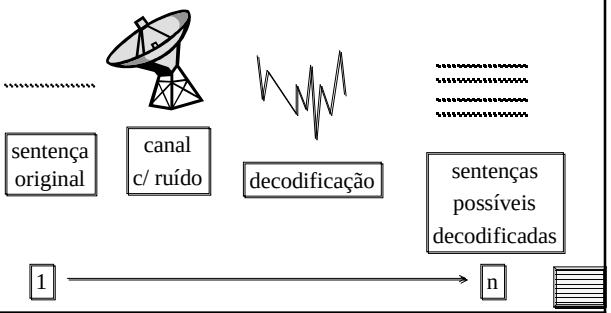
Histórico

- 1983 - 1993: *revival* dos modelos baseados em autômatos e dos modelos probabilísticos (IBM)
- 1994 - 1999:
  - tendência à unificação / métodos híbridos
  - *web* coloca novo desafio: busca e extração de informação baseada em linguagem

Estado da arte

- programas comerciais
  - previsão tempo (Canadá)
    - input: dados meteorológicos
    - output: boletim meteorológico em inglês e francês
  - Babel Fish: tradução no AltaVista (> 10^6 consultas / dia)
  - inf. turística: perguntas em LN sobre restaurantes
  - tutor de leitura para melhorar alfabetização
  - avaliação de monografias de estudantes
  - "narração" em LN de um vídeo de jogo de futebol
  - predição de sequência de palavras (auxílio pessoas com deficiência de fala)

Problema da Decodificação



Métodos e Algoritmos

- máquinas de estado, autômato finito
- linguagens formais, lógica, gramáticas
- teoria da probabilidade
- busca e programação dinâmica
- n-gram e modelo oculto de markov

N-grams

- predição da próxima palavra: tarefa ligada à ambiguidade e ruído
- exemplo:
  - Telefonista, gostaria de fazer uma chamada ….. [a cobrar, internacional, …]
  - problema: encontrar as probabilidades associadas
  - uso: identificação de erros, ajuda à pessoas com deficiências (afasia), atendimento automático
- obs. das palavras ao redor (contexto fonético, não sintático ou semântico!): *I'll meet you in about 15 minuets*

Corpora

- *corpus*: conjunto de palavras usadas para contagem da ocorrência de cada palavra (ou duplas, triplets, etc.)
- exemplos: *Brown corpus*, *Switchboard* (transcrição de conversas telefônicas), catálogo telefônico, conjunto das obras de Sheakspeare, etc.

I need	0.0016	need need	0.000047	# Need	0.000018
I the	0.00018	need the	0.012	# The	0.016
I on	0.000047	need on	0.000047	# On	0.00077
I I	0.039	need I	0.000016	# I	0.079
the need	0.00051	on need	0.000055		
the the	0.0099	on the	0.094		
the on	0.00022	on on	0.0031		
the I	0.00051	on I	0.00085		

Figure 7.7 Bigram probabilities for the words *the*, *on*, *need*, and *I* following each other, and starting a sentence (i.e., following #). Computed from the combined Brown and Switchboard corpora with add-0.5 smoothing.

N-grams

- processo
  - calcular a prob. individual das palavras: *corpora*, dicionários
  - input: n-1 palavras anteriores
  - saída: n-ésima palavra
  - exemplo:
    - "the" é aprox. 7% do Brown *corpus* enquanto que "rabbit" aparece 0,01%
    - *white* …. (the ou rabbit ???)
    - é mais razoável olhar para as palavras em associação com outras palavras vizinhas : p (rabbit | white)

N-grams

- prevêr prox. palavra baseado na "história" do texto (estima a função de prob. *p*): $p(w_n \mid w_1,\ldots,w_{n-1})$
- não é possível encontrar a prob. de w_n dadas todas as palavras $w_1,\ldots,w_{n-1}$ (porque ?)
- hipótese de markov: as últimas (poucas) *k-1* palavras afetam a k-ésima palavra
- ordem do modelo de markov:
  - k = 1 (unigram) [ordem zero]
  - k = 2 (bigram) [ordem 1]
  - k = n (n-gram) [ordem n-1]

N-grams

- exemplo 1
  - S. engoliu a grande ….
  - S. swallowed the large green ….
    - trad. não reproduz totalmente a idéia
    - em português: série de palavras masculinas seria eliminada
- exemplo 2
  - bigram (aprox. por 2 palavras)
  - p (rabbit | Just the other day I saw a white) ≈ p (rabbit | white)
  - p (rabbit | white) = C (white rabbit) / $\sum_w$ C (white <any w>) = C (white rabbit) / C (white)

N-grams

- exemplo 3:
  - Berkeley Restaurant Project
    - usuário faz perguntas sobre restaurantes em Berkeley, CA
      - I'm looking for Cantonese food
      - I want to eat Chinese food for lunch

eat on	.16	eat Thai	.03
eat some	.06	eat breakfast	.03
eat lunch	.06	eat in	.02
eat dinner	.05	eat Chinese	.02
eat at	.04	eat Mexican	.02
eat a	.04	eat tomorrow	.01
eat Indian	.04	eat dessert	.007
eat today	.03	eat British	.001

Figure 6.2 A fragment of a bigram grammar from the Berkeley Restaurant Project showing the most likely words to follow *eat*.

N-grams

<s> I	.25	I want	.32	want to	.65	to eat	.26	British food	.60
<s> I'd	.06	I would	.29	want a	.05	to have	.14	British restaurant	.15
<s> Tell	.04	I don't	.08	want some	.04	to spend	.09	British cuisine	.01
<s> I'm	.02	I have	.04	want thai	.01	to be	.02	British lunch	.01

Figure 6.3 More fragments from the bigram grammar from the Berkeley Restaurant Project.

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

Figure 6.4 Bigram counts for seven of the words (out of 1616 total word types) in the Berkeley Restaurant Project corpus of ≈10,000 sentences.

N-grams

- Uso para gerar texto
  - treinar o modelo sobre um *corpus*
  - classificar todos n-grams de acordo com probabilidade
  - gerar um número randômico
  - pegar n-gram correspondente
- Poder dos n-grams: aumenta com a ordem

Métricas sobre n-grams

- Entropia:
  - mede a quantidade de informação de uma gramática
  - dadas 2 gramáticas e 1 *corpus*, diz qual das gramáticas "casa" com o *corpus*
- função de um conj. χ (de palavras, letras, sinais, etc.) e X é uma var. aleatória sobre χ
- def.:

$$H(X) = -\sum_{x \in \chi} p(x) \cdot \log p(x)$$

Entropia

- exemplo
  - apostar em cavalos (8 por páreo)
  - enviar msg. c/ aposta (+ curta possível)
  - 8 cavalos com igual probabilidade => 3 bits
  - se prob. *a priori* de cada cavalo é: 1/2 (c1), 1/4 (c2), 1/8 (c3), 1/16 (c4), 1/64 (c5 a c8)
  - então *H(x)* sobre os 8 cavalos é: $H(X) = -\sum_{i=1}^{8} p(i) \cdot \log p(i)$
  - *H(x) = 2* bits (média dos bits necessários)
  - codificação: 0 (c1), 10 (c2), 110 (c3), 1110 (c4), etc.

Entropia

- entropia para uma sequência (gramática)
  - seq. de palavras $W=\{\ldots w_0, w_1, w_2, \ldots, w_n\}$ da ling. *L*
  - neste caso X é definido sobre todas seqs. finitas de W com tamanho *b*
  - exemplo:
    - seja a seq. $W_1^n = \{w_1, w_2, \ldots, w_n\}$
    - $H(w_1, w_2, \ldots, w_n) = -\sum_{w \in L} p(W_1^n) \cdot \log p(W_1^n)$

Entropia

- taxa de entropia (entropia por palavra) é a entropia da seq. dividida pelo número de palavras:

$$\frac{1}{n} H(W_1^n) = -\frac{1}{n} \sum_{w \in L} p(W_1^n) \cdot \log p(W_1^n)$$

- entropia de uma linguagem (proc. estocástico L produzindo seq. de palavras de compr. infinito):

$$H(L) = \lim_{n \to \infty} -\frac{1}{n} H(W_1^n)$$

Entropia

- teorema de Shannon-McMillan-Breiman

$$H(L) = \lim_{n\to\infty} -\frac{1}{n} H(W_1^n) =$$

$$\lim_{n\to\infty} -\frac{1}{n} \sum_{W_1^n \in L} p(w_1, w_2, ..., w_n) \cdot \log p(w_1, w_2, ..., w_n) =$$

$$\lim_{n\to\infty} -\frac{1}{n} \log p(w_1 w_2 ... w_n)$$

- interpretação: em 1 seq. suficientemente longa de palavras, qq. outra sub-seq. estará contida; não é necessário somar sobre todas possíveis seqs.

Perplexidade

- segunda métrica: média ponderada do número de escolhas associadas à uma variável aleatória
- perplexidade ou perplexidade cruzada
- dado por 2^H

Autômato Finito Ponderado

- extensão do modelo autômato finito: cada arco associado à probabilidades que indicam quão provável é o caminho
- concepção:
  - Cohen 1989: dicionários e regras (manuais) como as do exemplo para diferentes formas de pronunciação (ou escrita)
  - probabilidades vêm da contagem sobre um *corpus*
  - outras propostas
- base:
  - Bayes: *p(A|B) = [p(B|A) * p(A)] / p(B)*

Autômato Finito: Formalismo

- Definição:
  - sequência de estados $q = (q_0\ q_1\ q_2 \ldots\ q_n)$ cada um correspondendo a um fonema
  - conjunto de probabilidade de transição entre os estados: $a = (a_{01}\ a_{12}\ a_{23}\ a_{34} \ldots)$ indicando a probabilidade de um fonema seguir um outro
  - sequência observada (ou produzida) $O = (o_1\ o_2\ o_3 \ldots o_n)$
  - nodos = estados
  - arcos = probabilidades

Exemplo

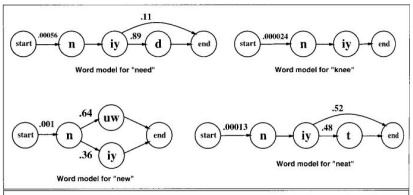


Figure 5.15 Pronunciation networks for the words *need*, *neat*, *new*, and *knee*. All networks are simplified from the actual pronunciations in the Switchboard corpus. Each network has been augmented by the unigram probability of the word (i.e., its normalized frequency from the Switchboard+Brown corpus). Word probabilities are not usually included as part of the pronunciation network for a word; they are added here to simplify the exposition of the forward algorithm.

Problema da Decodificação

- dada uma sequência *O*, um modelo de palavra *w*, calcular qual palavra do modelo pode ter produzido *O* (ou seja *p(O|w) * p(w)*)

Algoritmo *Forward*

function FORWARD(*observations,state-graph*) **returns** *forward-probability*

num-states ← NUM-OF-STATES(*state-graph*)
num-obs ← *length*(*observations*)
Create probability matrix *forward*[*num-states* + 2, *num-obs* + 2]
forward[0,0] ← 1.0
for each time step *t* **from** 0 **to** *num-obs* **do**
  for each state *s* **from** 0 **to** *num-states* **do**
    for each transition *s'* from *s* specified by *state-graph*
      forward[*s'*,*t*+1] ← *forward*[*s*,*t*] * *a*[*s*, *s'*] * *b*[*s'*, o_t]
return the sum of the probabilities in the final column of *forward*

Figure 5.16 The forward algorithm for computing likelihood of observation sequence given a word model. $a[s,s']$ is the transition probability from current state s to next state s', and $b[s',o_t]$ is the observation likelihood of s' given o_t. For the weighted automata that we consider here, $b[s',o_t]$ is 1 if the observation symbol matches the state, and 0 otherwise.

Cálculo

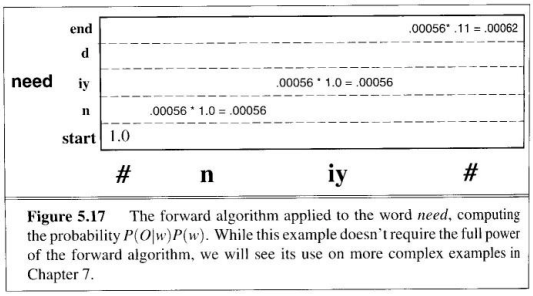


Figure 5.17 The forward algorithm applied to the word *need*, computing the probability $P(O|w)P(w)$. While this example doesn't require the full power of the forward algorithm, we will see its use on more complex examples in Chapter 7.

Algoritmo Viterbi

- motivação:
  - melhorar a eficiência do alg. *forward*:
    - para que calcular prob. em todos os caminhos?
    - para que calcular prob. para todas palavras?
  - Viterbi considera todas palavras simultaneamente para calcular o caminho mais provável

Algoritmo Viterbi

function VITERBI(*observations* of len *T*,*state-graph*) **returns** *best-path*

num-states ← NUM-OF-STATES(*state-graph*)
Create a path probability matrix *viterbi*[*num-states*+2,*T*+2]
viterbi[0,0] ← 1.0
for each time step *t* **from** 0 **to** *T* **do**
  for each state *s* **from** 0 **to** *num-states* **do**
    for each transition *s'* from *s* specified by *state-graph*
      new-score ← *viterbi*[*s*, *t*] * *a*[*s*,*s'*] * $b_{s'}(o_t)$
      if ((*viterbi*[*s'*,*t*+1] = 0) || (*new-score* > *viterbi*[*s'*, *t*+1]))
        then
          viterbi[*s'*, *t*+1] ← *new-score*
          back-pointer[*s'*, *t*+1] ← *s*
Backtrace from highest probability state in the final column of *viterbi*[] and return path

Figure 5.19 Viterbi algorithm for finding optimal sequence of states in continuous speech recognition, simplified by using phones as inputs. Given an observation sequence of phones and a weighted automaton (state graph), the algorithm returns the path through the automaton which has maximum probability and accepts the observation sequence. $a[s,s']$ is the transition probability from current state s to next state s', and $b[s',o_t]$ is the observation likelihood of s' given o_t. For the weighted automata that we consider here, $b[s',o_t]$ is 1 if the observation symbol matches the state, and 0 otherwise.

Entrada e Saída

- entrada:
  - autômato ponderado representando as palavras do modelo
  - conjunto de sequências observadas $O = (o_1\ o_2\ o_3 \ldots o_n)$
- saída:
  - sequência mais provável $q = (q_0\ q_1\ q_2 \ldots q_n)$ e sua probabilidade

Exemplo

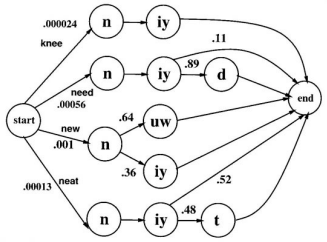


Figure 5.18 The pronunciation networks for the words *need*, *neat*, *new*, *knee* combined into a single weighted automaton. Again, word probabili are not usually considered part of the pronunciation network for a word; t are added here to simplify the exposition of the Viterbi algorithm.

Cálculo

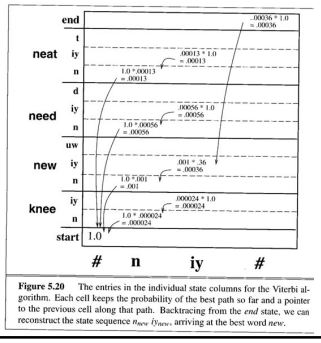


Figure 5.20 The entries in the individual state columns for the Viterbi algorithm. Each cell keeps the probability of the best path so far and a pointer to the previous cell along that path. Backtracing from the *end* state, we can reconstruct the state sequence $n_{new}\ iy_{new}$, arriving at the best word *new*.

Segmentação de Palavras e Frases

- AFP podem ser tb. usados em problemas de segmentação em línguas como japonês, chinês ou em texto falado
- representação de morfemas
  - cada palavra é representada por uma série de arcos representando cada caracter da palavra seguido de um arco representando a probabilidade da palavra

Modelo Oculto de Markov

- hidden markov model (HMM)
- motivação: simplificações feitas no mod. do AFP podem não ser válidas
  - *input* não é uma seq. de símbolos; na realidade a entrada é ambigua
  - símbolos de entrada não correspondem exatamente aos estados do autômato
  - em um HMM não se pode saber exatamente para qual estado mover dado um símbolo pois este não determina o próximo passo de maneira única

Modelo Oculto de Markov

- definição formal:
  - símbolos observados O (não necessariamente do mesmo alfabeto dos estados Q)
  - conj. de estados ($Q = q_1\ q_2\ q_3 \ldots\ q_n$)
  - função de probabilidade de observação (B) cujos valores não se limitam a zero e um (podem assumir qq. valor neste intervalo): $B = b_i\ (o_t)$ ou seja prob. da observ. o_t ser gerada no estado i

Modelo Oculto de Markov

- definição formal (cont.):
  - prob. de transição $A = a_{01}\ a_{12}\ a_{23} \ldots\ a_{n1} \ldots\ a_{nn}$
  - distr. inicial de prob. sobre os estados (π) de forma que π é a prob. de que o HMM se inicie no estado i (se π_j=0, j não pode ser um estado inicial)
  - estados aceitos: conj. de estados legalmente aceitos

Exemplo

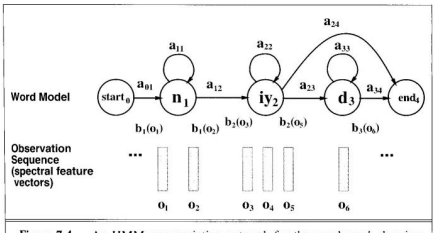


Figure 7.4 An HMM pronunciation network for the word *need*, showing the transition probabilities, and a sample observation sequence. Note the addition of the output probabilities B. HMMs used in speech recognition usually use self-loops on the states to model variable phone durations.

Modelo Oculto de Markov

- característica fundamental: não utiliza todo o espaço de busca (todas sentenças possíveis) mas apenas as que tem alguma chance de parecer com a original
- seja *O* uma seq. de símbolos ou observações individuais (p. ex. discretizando o sinal acústico)
- qual é a sentença mais provável dentre as sentenças da linguagem L dada uma entrada acústica *O* ? ou seja:

$$\hat{W} = \arg\max_{W \in L} p(W \mid O)$$

Modelo Oculto de Markov

- *argmax f(x)* retorna o x t.q. f(x) é máximo
- para uma dada sentença *W* e uma sequência *O*, calcular a sentença de prob. máxima:

$$\hat{W} = \frac{\arg\max_{W \in L} p(O \mid W) \cdot p(W)}{p(O)}$$

Modelo Oculto de Markov

- *p(O|W), p(W) e p(O)* são mais fáceis de obter que *p(W|O)*:
  - *p(W)* vem do n-gram
  - *p(O)* pode ser ignorada (não muda para cada sentença e portanto pode ser desprezada na maximização)
- logo: a sentença *W* mais provável dada uma observ. *O* é calculada pelo produto da sua prob. a priori *p(W)* pela prob. do modelo acústico

$$\hat{W} = \arg\max_{W \in L} p(O \mid W) \cdot p(W)$$

Modelo Oculto de Markov

- algoritmo *forward* para sentenças: intratável
- alternativa: Viterbi

Exemplo

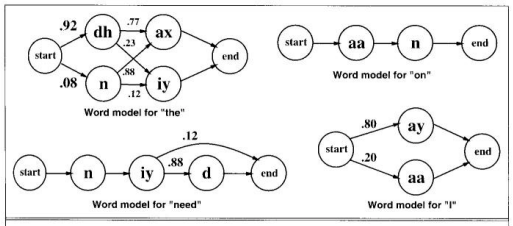


Figure 7.5 Pronunciation networks for the words *I*, *on*, *need*, and *the*. All networks (especially *the*) are significantly simplified.

Exemplo

I need	0.0016	need need	0.000047	# Need	0.000018
I the	0.00018	need the	0.012	# The	0.016
I on	0.000047	need on	0.000047	# On	0.00077
I I	0.039	need I	0.000016	# I	0.079
the need	0.00051	on need	0.000055		
the the	0.0099	on the	0.094		
the on	0.00022	on on	0.0031		
the I	0.00051	on I	0.00085		

Figure 7.7 Bigram probabilities for the words *the*, *on*, *need*, and *I* following each other, and starting a sentence (i.e., following #). Computed from the combined Brown and Switchboard corpora with add-0.5 smoothing.

Exercício

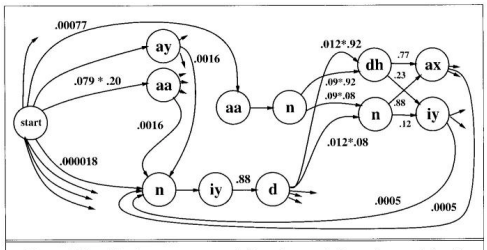


Figure 7.8 Single automaton made from the words *I*, *need*, *on*, and *the*. The arcs between words have probabilities computed from Figure 7.7. For lack of space the figure only shows a few of the between-word arcs.

Exemplo

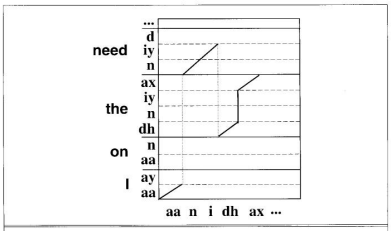


Figure 7.6 An illustration of the results of the Viterbi algorithm used to find the most-likely phone sequence (and hence estimate the most-likely word sequence).

Exemplo

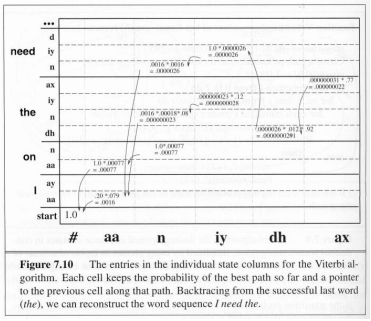
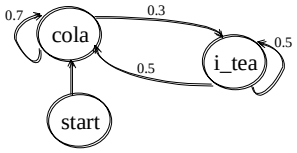


Figure 7.10 The entries in the individual state columns for the Viterbi algorithm. Each cell keeps the probability of the best path so far and a pointer to the previous cell along that path. Backtracing from the successful last word (*the*), we can reconstruct the word sequence *I need the*.

Exercício

- calcular a prob. da seq. de estados {lemonade, iced_tea} na seguinte maq. de refrigerante

$p(x_1 \ldots x_n) = p(x_1) \,.\, p(x_2|x_1) \,.\, p(x_3|x_1,x_2) \ldots p(x_n|x_1,x_2,\ldots x_{n-1})$



prob. de emissão

Resp.: 0.084 = 0.7*0.3*0.7*0.1+0.7*0.3*0.3*0.7+0.3*0.2*0.5*0.7+0.3*0.2*0.5*0.1

Aplicações de HMM

- 3 tipos de problemas:
  - (1) dado um modelo $\mu=(A,B,\pi)$, como calcular (de modo eficiente) quão provável é uma certa observação [$p(O|\ \mu)$] ?
  - (2) dada uma observação O e um modelo μ, como escolher uma sequência de estados $(x_1\, x_2 \ldots x_n)$ que melhor explique estas observações?
  - (3) dada uma sequência O e um conj. de possíveis modelos (obtidos através da variação de parâmetros A,B,π de μ), como encontrar o modelo que melhor explica os dados observados?

Aplicações de HMM

- questões:
  - qual é o melhor modelo? (prob. tipo 1)
  - qual o caminho seguido? (prob. tipo 2)
  - quais parâmetros? (prob. tipo 3)

Resumo

- representação de vários problemas associados à linguagem como seq. de símbolos submetidos à canal com ruídos que deve ser recuperada
- recuperação: considerar todas seqs. possíveis ordenadas por sua prob. condicional (Bayes divide as probs. em *a priori* e prob. em si, obtida por treino do modelo)
- decodificação: encontrar a seq. original que gerou a seq. com ruído
- cálculo da "distância"entre 2 seqs.: *minimum edit sequence* e posterior alinhamento

Resumo

- algoritmos:
  - *forward*: modo eficiente de calcular a prob. de uma seq. observada dado um modelo representado por um autômato
  - Viterbi: modo eficiente de solucionar o problema da decodificação (considera todas strings possíveis e usa Bayes para calcular suas probabilidades de gerar a seq. observada)
- segmentação de palavras: alg. usado para segmentar seqs. expressando frases de línguas sem marcação explícita de palavras (japonês, chinês) ou texto falado

Resumo

- n-gram
  - prob. condicional de uma palavra dadas as n-1 prévias palavras
  - prob. simples pode ser obtida em *corpora* e normalizadas
  - vantagem: inclui o conhecimento léxico
  - desvantagem: dependente do *corpus* usado para treinamento
  - métricas: entropia (quantidade de informação) e perplexidade (para comparar 2 modelos probabilísticos)

Bibliografia

- Russel and Norvig (1995): Art. Int. a Modern Approach (Prentice Hall)
- Jurafsky and Martin (2000): Speech and Language Processing (Prentice Hall)
- Manning and Schütze (1999, 2001): Foundations of Statistical Natural Language Processing (MIT Press)
- Cover and Thomas (1991): Elements of Information Theory (Wiley)