Laboratório VISGRAF
Instituto de Matemática Pura e Aplicada

Mtodos Probabilisticos para Reconhecimento de Voz

Anderson Mayrink da Cunha
Luiz Velho (orientador)

Technical Report TR-03-04 Relatório Técnico

June - 2003 - Junho

Reconhecimento de Voz

Anderson Mayrink da Cunha

Sumário

Lista de Figuras

Introdução

Esse texto teve origem no curso de leitura em reconhecimento de voz realizado no impa no ano de 2002, sob orientação do professor Luiz Velho.

A área de reconhecimento de voz tem atraído, desde a década de 70, muitos investimentos em pesquisas e grande interesse acadêmico e comercial. Atualmente essa área já está bastante madura e é o problema da percepção humana mais bem estudado.

O reconhecimento de voz tem muitas aplicações comerciais e atualmente existem programas que reconhecem a voz natural em tempo real de modo bastante satisfatório.

Devido a maturidade e eficiência dos algoritmos e métodos de reconhecimento de voz, o seu estudo é útil em outras áreas como visão computacional e reconhecimento em geral.

Esse texto foi dividido em cinco capítulos. No capítulo 1 apresentamos brevemente o que é voz, as dificuldades da tarefa de reconhecimento de voz, uma breve história da área, o modelo Bayesiano e processamento de voz.

O capítulo 2 é dedicado a Cadeias de Markov e principalmente a Hidden Markov Models, que é o principal modelo estatístico para reconhecimento de voz.

No capítulo 3 apresentamos sistemas de reconhecimento de voz, com ênfase em sistemas de reconhecimento de voz para fala natural (grande vocabulário e palavras conectadas).

No capítulo 4 apresentamos o conceito de entropia e no capítulo 5 temos algumas aplicações da entropia em modelagem de linguagem.

Capítulo 1

Preliminares

Neste capítulo vamos apresentar vários conceitos básicos em reconhecimento de voz. Inicialmente apresentamos o que é voz, as dificuldades da tarefa de reconhecimento de voz e uma breve história da área. Apresentamos também um importante método estatístico, o modelo Bayesiano. Por fim temos um sistema de processamento de voz para reconhecimento.

1.1 Voz e Variabilidade da Voz

A voz é produzida quando o ar é expelido pelos pulmões, passa pela traquéia e atinge a laringe, que junto com as cordas vocais tensionadas fazem o ar vibrar em pulsos quasi-periódicos que são modulados pelos tratos vocais e nasais. As diferentes posições dos vários articuladores (língua, boca, etc..) produzem os diferentes sons.

Os fonemas (sons básicos) são produzidos no curto período de tempo em que esses articuladores estão em posições estáveis. Quando esses articuladores se movem para outra posição estável outro fonema é produzido. Existem em torno de 50 fonemas e na fala temos em torno de 8 fonemas por segundo.

A voz (o som) transmitida por uma pessoa é uma onda de pressão que tem frequência máxima por volta de 16 kHz, mas a maior parte da energia está abaixo dos 7 kHz. A voz é convertida em sinal elétrico por um microfone. Um conversor analógico-digital converte esse sinal contínuo em um sinal discreto quantizado em um número fixo, por exemplo 16 bits. Note que num sistema de reconhecimento de voz há uma grande compressão de informações de 50.000 a 100.000 bps do sinal contínuo para torno de 50 bps da produção dos fonemas[1].

O reconhecimento de voz é uma tarefa bastante complexa pois vários fatores influenciam a produção da voz. A mesma palavra pode ser dita pela mesma pessoa em diferentes ocasiões de modos bastante distintos. O fenômeno da co-articulação (interferência na produção do final de um som com o início do som seguinte) altera bastante o sinal de voz. O contexto onde uma palavra é dita pode alterar o seu som. Entre alguns fatores que mudam a forma de onda do sinal de voz podemos citar: uma doença na faringe como uma gripe, por exemplo, a fadiga, o stress, a velocidade da fala, se a fala é espontânea ou a leitura de um texto, a atitude ou a emoção ao falar uma palavra.

[1] pois existem por volta de 50 fonemas e há em torno de 8 fonemas por segundo na fala. Como $2^6 = 64 > 50$, temos por volta de $8.6 = 48$ fonemas/seg.

Diferentes pessoas falam as mesmas palavras de diferentes formas por razões orgânicas como constituição do trato vocal, idade ou sexo ou por razões culturais como sotaque, status social ou educação.

Razões físicas como o meio ambiente, eco, distorção, o microfone ou o tipo de ruído também são fatores fundamentais no reconhecimento de voz.

1.2 Interdisciplinaridade

Outra dificuldade da pesquisa em reconhecimento de voz é seu aspecto multi-disciplinar Várias disciplinas tem sido aplicadas em problemas de reconhecimento de voz, entre elas:

- Processamento de sinais: O processo de extrair informações relevantes do sinal de voz de maneira robusta e eficiente.
- Física (acústica): A compreensão da relação entre o sinal de voz e os mecanismos fisiológicos (o trato vocal humano).
- Reconhecimento de Padrões: O conjunto de algoritmos usados para agrupamento de dados com o objetivo de criar protótipos padrões.
- Teoria da Informação: Procedimentos para estimar parâmetros e avaliação de modelos acústicos estatísticos e modelos de linguagem eficientes.
- Linguística: A relação entre sons (fonologia), palavras na linguagem (sintaxe) e significado das palavras (semântica).
- Ciência da Computação: O estudo de algoritmos eficientes para implementação de vários métodos usados em sistemas de reconhecimento de voz na prática.

1.3 Breve História

O reconhecimento de voz é uma área de pesquisa bastante madura e é um dos problemas da percepção humana mais bem estudados.

As primeiras tentativas de reconhecimento de voz por computadores são da década de 50. Em 1952 (o primeiro computador de uso geral, o ENIAC, foi construído em 1946), pesquisadores dos laboratórios Bell construiram um sistema de reconhecimento de dígitos isolados para um único orador. Este sistema era baseado em medidas espectrais das vogais de cada dígito.

As principais estratégias de reconhecimento de voz nas décadas de 50 e 60 eram segmentar o som em sucessivos fonemas (unidades básicas da pronúncia), identificar os fonemas particulares (baseados em análise espectral) e transcrever o fonema reconhecido em texto.

Na década de 70, o paradigma dominante para reconhecimento de voz de pequeno vocabulário e palavras isoladas (palavras com intervalos entre elas) era o Dynamic Time Warping (DTW). A idéia básica do DTW era deformar (warp) um protótipo de observação (template) de um som nas sequências de sons observadas e chegar a decisão entre várias palavras competidoras de acordo com uma penalização de warping. A penalização mais comum era a distância de Itakura. DTW dava resultados muito bons e nessa época apareceram os primeiros sistemas de reconhecimento

de voz comerciais (para pequeno vocabulário e palavras isoladas). Os principais problemas com DTW eram: (a) a incorporação de modelos de linguagem não era natural, (b) a construção de protótipos sintéticos não foi resolvido e (c) não foi achado uma formulação estatística unificada incorporando todos os módulos de um sistema de reconhecimento de voz. DTW não é baseado em idéias de modelamento de sinais estatístico em estrito senso. DTW é melhor classificado com um método simplificado, não-paramétrico em que várias sequências de referência são usadas para caracterizar a variação entre diferentes sons.

Na década de 80 foi introduzido em reconhecimento de voz o método estatístico baseado em Hidden Markov Models (HMM), que não tem os problemas de DTW e possibilita reconhecimento de voz em grandes vocabulários e de palavras conectadas, como na fala natural. Desde então a abordagem de DTW foi essencialmente abandonada.

Na década de 90 apareceram sistemas de reconhecimento de voz baseados em redes neurais ou sistemas híbridos de redes neurais e HMM. No entanto, HMM continua sendo atualmente o principal modelo para reconhecimento de voz.

Neste texto apresentamos somente reconhecimento de voz com HMM. Uma boa fonte de referência para DTW ou redes neurais, assim como para as seções anteriores é [2].

1.4 Percepção e Estatística

Como já foi dito na seção anterior, o modelo atualmente usado em reconhecimento de voz é estatístico. Além dos resultados práticos obtidos com esse modelo, é interessante ressaltar que a estatística é a base da percepção humana. A principal razão para ser difícil aceitar esse fato é que não estamos conscientes de 99% das ambiguidades que manipulamos a cada segundo. O que os sentidos realmente percebem não chega a nossa consciência. O que se torna consciente é o sinal sensorial realçado com nossas expectativas e memórias, que completam cada elemento da percepção. Os experimentos psicofísicos de Warren [5] (obtidos em [4]) constituem um bom exemplo desse fato. Nesse experimento foram gravadas várias frases e alguns fonemas foram substituídos por ruído. Essas frases modificadas foram ouvidas por outras pessoas que não perceberam a falta de nenhum fonema, mas acreditavam ter ouvido o fonema que fazia a frase semanticamente consistente. A tabela abaixo indica esse experimento.

Som real	Som percebido
the ?eel is on the shoe	the heel is on the shoe
the ?eel is on the car	the wheel is on the car
the ?eel is on the table	the meal is on the table
the ?eel is on the orange	the peel is on the orange

Esse experimento mostra que o sinal real não chega à consciência e que a escolha da percepção é uma questão de probabilidade, isto é, chega à consciência o som mais provável segundo regras semânticas quardadas na nossa memória ou critérios pessoais (inconscientes).

1.5 Formulação Matemática

Antes de apresentar o modelo específico para reconhecimento de voz vamos apresentar um modelo estatístico mais geral, o modelo Bayesiano. Para isso é necessário:

- Um conjunto de variáveis aleatórias $X = (A, W)$ onde A é o sinal observado e W são variáveis ocultas, que descrevem eventos ou objetos causados por A.
- Um modelo estocástico que permite descrever a variabilidade e o ruído no sinal.
- Parâmetros específicos θ do modelo estocástico que melhor descreve a classe dos sinais que tentamos decifrar.

Existem 3 problemas básicos que apresentamos agora:

1. O cálculo de $P(X|\theta) = P(A, W|\theta) = P(A|W, \theta)P(W|\theta)$.
2. O treinamento do modelo, isto é, a obtenção dos parâmetros θ do modelo que melhor descreve a classe dos sinais que tentamos decifrar. Esses parâmetros devem ser obtidos a partir de dados reais $\{X^\alpha\}$. Devemos escolher θ que maximiza $\Pi_\alpha P(X^\alpha|\theta)$.
3. Obtenção das variáveis ocultas, isto é, dado um sinal A e os parâmetros θ do modelo, maximizar $P(W|A, \theta)$. Pela regra de Bayes, temos que:

$$P(W|A, \theta) = \frac{P(A|W, \theta)P(W|\theta)}{P(A|\theta)}$$

Como o fator $P(A|\theta)$ é comum para qualquer W, basta maximizar $P(A|W, \theta)P(W|\theta)$. Isto é, devemos obter:

$$W^* = \arg\max_W P(A|W, \theta)P(W|\theta)$$

Vamos agora especificar o modelo Bayesiano para sinais de voz.

Seja A a evidência acústica, o dado de entrada de voz. Como estamos trabalhando com computadores digitais, sem perda de generalidade, assumimos que A é uma sequência de símbolos tomados de um (possivelmente grande) alfabeto $\mathcal{A}$:

$$A = a_1 a_2 ... a_m \qquad a_i \in \mathcal{A}$$

onde os a_i são gerados no tempo. Seja

$$W = w_1 w_2 ... w_n \qquad w_i \in \mathcal{W}$$

onde W denota uma sequência de n palavras, cada uma pertence a um dicionário fixo $\mathcal{W}$.

Como já comentado, o principal modelo usado para reconhecimento de voz é baseado em HMM. Os 3 problemas básicos são os já descritos acima. O problema 3, que descreve qual a sequência de palavras que melhor corresponde à entrada acústica pode ser assim escrita:

$$W^* = \arg\max_W P(A|W)P(W)$$

onde omitimos os parâmetros θ do modelo.

O objetivo do próximo capítulo é descrever HMM e um modelo mais simples no qual HMM são baseados: Cadeias de Markov.

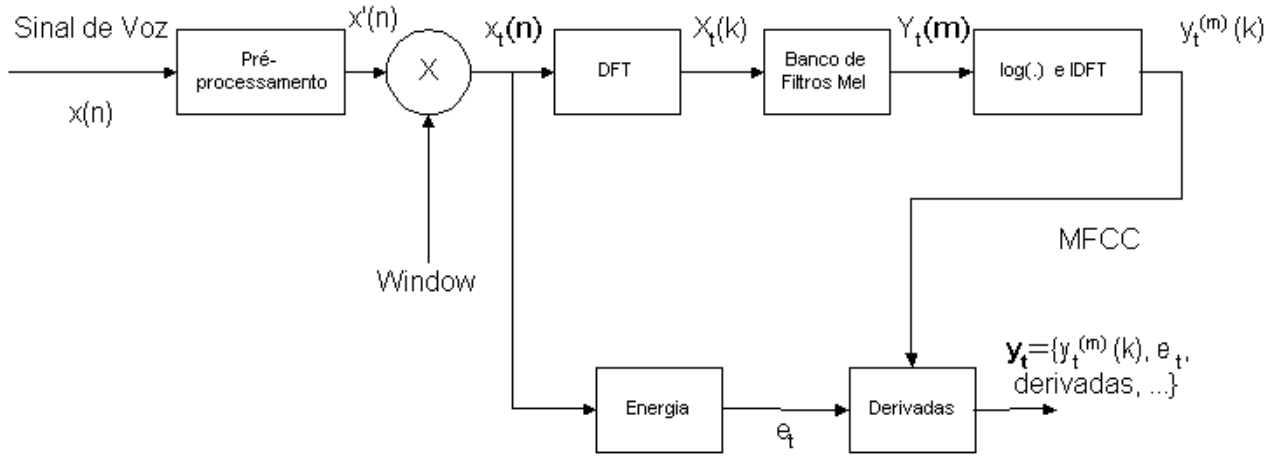

Figura 1.1: Sistema de Processamento de Voz

1.6 Processamento Acústico

O processamento acústico é parte importante para uma boa performance de um sistema de reconhecimento de voz. Vamos apresentar nessa seção um sistema simples de processamento de voz. Maiores detalhes podem ser obtidos em [2] ou [3].

Na figura 1.1 temos um sistema de processamento de voz (baseado em [3]). Nesse sistema não há discretização dos dados de saída, o que pode ser feito com um algoritmo de quantização vetorial, se necessário. Descrevemos brevemente a seguir alguns dos componentes desse sistema.

1.6.1 Pré-processamento

As altas frequências, apesar da pequena amplitude em relação às baixas frequências do sinal, possui informações relevantes. A pré-ênfase é usada para amplificar as altas frequências do sinal e pode ser feita com um filtro cuja função de transferência é:

$$H(z) = 1 - a.z^{-1} \qquad 0 \leq a \leq 1$$

No domínio do tempo o sinal de saída $x'(n)$ é relacionado com o sinal de entrada $x(n)$ com a fórmula:

$$x'(n) = x(n) - a.x(n-1)$$

Um valor típico usado é $a = 0.95$, que dá $20dB$ de amplificação para altas frequências.

Pode ser necessário, para aumentar a performance de um sistema de reconhecimento de voz, eliminar grandes intervalos de silêncio do sinal. Para isso um detector simples baseado na energia é suficiente.

1.6.2 Janela (Windowing)

Os métodos de avaliação espectral dão bons resultados no caso de sinais estacionários (sinais cujas características estatísticas são invariantes em relação ao tempo). Para voz, isto vale somente para períodos curtos de tempo. Por isso fazemos um"windowing" do sinal $x'(n)$ em uma sequência de frames $x'_t(n)$, para serem posteriormente processados individualmente. Os frames $x'_t(n)$ podem ser calculados da forma:

$$x'_t(n) = w(n)x'(n - tQ) \qquad 0 \leq n < N, \qquad t = 1, ..., T$$

onde Q é o tamanho da janela, que depende da aplicação. Um tamanho de janela Q de 20 a 50 ms é geralmente usado em reconhecimento de voz.

O formato de janela retangular tem o problema de introduzir distorção no sinal (a transformada de Fourier é a função *sinc*). Uma janela comumente usada em reconhecimento de voz é a janela de Hamming definida como:

$$\begin{cases} w(n) = 0.54 - 0.46\cos(\frac{2\pi n}{N-1}), & n = 0, 1, ..., N-1 \\ w(n) = 0, & \text{caso contrário.} \end{cases}$$

1.6.3 Banco de Filtros e MFCC

Após calcular a transformada de Fourier dos frames, vamos obter as features espectrais da voz com um banco de filtros, que calcula a integral do espectro em faixas de frequência definidas.

Um banco de filtros comumente usado em reconhecimento de voz é o banco de filtros Mel, que é constituído de um total de 24 filtros passa-faixa, com 10 filtros uniformemente espaçados no eixo de frequência até 1 kHz. Acima de 1 kHz as faixas estão distribuidas segundo uma escala logaritmica. Este tipo de distribuição simula o processamento do ouvido humano. O banco de filtros Mel é composto de filtros triangulares (no domínio de frequência) transladados.

Após o banco de filtros, calculamos o logaritmo da magnitude desses valores, pois há vantagens teóricas em tal procedimento e segue o processamento do ouvido humano. Após o cálculo do logaritmo aplicamos a transfomada inversa de Fourier, obtendo assim o MFCC (Mel Frequency Cepstrum Computation). Sendo $Y_t(m)$ a saída do banco de filtros, calculamos o MFCC da seguinte forma:

$$y_t^{(m)}(k) = \sum_{m=1}^{M} (\log |Y_t(m)|) \cos(\frac{k\pi}{M}(m - \frac{1}{2})), \qquad k = 0, ..., L$$

Em reconhecimento de voz, normalmente são descartados alguns dos últimos coeficientes do MFCC. O número de coeficientes de MFCC mantidos em geral é menor que 15. Isto provoca uma suavização no sinal.

O vetor de saída do processamento de voz é composto do MFCC, da energia do sinal e das derivadas (calculadas a partir de diferenças) do MFCC e da energia.

1.6.4 Quantização Vetorial

Como já foi dito, se desejamos trabalhar com observações discretas é necessário um algoritmo de quantização vetorial. Vamos descrever o algoritmo k-means, que é um caso particular do

algoritmo EM. O k-means (e suas variantes, por exemplo, o algoritmo LBG) é o algoritmo de quantização vetorial mais usado para propósitos gerais e também para reconhecimento de voz.

Vamos formular o problema de quantização vetorial: Seja X um conjunto de N pontos de R^d. Achar L (com $L \ll N$) classes tal que a distorção (soma das distâncias de cada ponto x_j ao centróide de sua classe) seja minimizada.

Para aplicar o algoritmo EM devemos procurar dados incompletos do problema. Podemos ver o dado $X = \{x_j\}$ como incompleto. O dado completo é:

$$z_j = \{x_j, y_{1j}, y_{2j}, \dots, y_{Lj}\}, \qquad j = 1 \cdots N$$

onde y_{ij} indica se x_j pertence à classe ω_i. Isto é, $y_{ij} = 1$ se x_j pertence à classe ω_i e $y_{ij} = 0$ caso contrário.

O algoritmo k-means é composto dos passos:

1. Inicialização: Obter um conjunto de centróides iniciais, de modo aleatório ou usando algum tipo de critério.

2. Passo **E** (Expectation): Dado as classes ω_i (seus centróides z_i) e a função distância, calcular as (esperanças das) variáveis y_{ij}. Basta classificar os pontos x_j:

$$y_{ij} = 1 \Leftrightarrow x_j \in \omega_i \Leftrightarrow dist(x_j, z_i) \leq dist(x_j, z_k), \forall k.$$

3. Passo **M** (Maximization): Calcular as novas classes ω_i (seus centróides z_i) que minimizam a distorção. Basta calcular o novo codebook $\{z_i\}$.

$$z_i = \frac{1}{n_i} \left(\sum_{x_j \in \omega_i} x_j \right)$$

   onde n_i é o número de vetores x_j na classe ω_i.

4. Iteração: Se a distorção decrescer muito pouco com a iteração, terminar a execução e retornar o novo codebook. Se não, ir para o passo **E** (com o novo codebook calculado no passo **M**).

Capítulo 2

Processos de Markov

Os processos de Markov têm aplicações em diversas áreas e se caracterizam pelo fato de ser sem memória, isto é, toda a história passada está completamente resumida no valor atual do processo. Nesse capítulo vamos estudar dois tipos de processos de Markov unidimensionais: Cadeias de Markov e, principalmente, Hidden Markov Models, que é o principal modelo estatístico para reconhecimento de voz.

2.1 Cadeias de Markov

Em 1907, Markov definiu e investigou propriedades que são hoje conhecidas como processos (ou cadeias) de Markov. A principal característica dos processos de Markov é que o modo que toda a história passada afeta o futuro está completamente resumida no valor atual do processo. Nos interessa principalmente cadeias de Markov em tempo discreto, que definimos a seguir.

Considere um sistema cuja evolução seja descrita por um processo estocástico $\{X(n) = X_n, n = 1, 2, \dots\}$, consistindo de uma família de variáveis aleatórias. O valor s_n assumido pela variável aleatória X_n é chamado de *estado* do sistema no tempo discreto n. O conjunto de todos os valores que as variáveis aleatórias podem assumir é chamado de espaço de estados do sistema. Se a estrutura do processo estocástico é tal que a distribuição de probabilidade condicional de X_n depende somente do valor de X_{n-1} e é independente de todos os valores anteriores, dizemos que o processo é uma cadeia de Markov. Mais precisamente:

Definição 1 *Uma sequência de variáveis aleatórias $X_1, X_2, \dots$ é uma cadeia de Markov em tempo discreto se para todo tempo n ($n = 1, 2, \dots$) e para todo o espaço de estados do sistema temos que:*

$$P[X_n = s_n \mid X_1 = s_1, X_2 = s_2, \dots, X_{n-1} = s_{n-1}] = P[X_n = s_n \mid X_{n-1} = s_{n-1}]$$

Podemos pensar na cadeia de Markov como um modelo gerador consistindo de estados ligados entre si por transições possíveis. Em cada unidade de tempo n um estado particular é visitado e o modelo coloca na saída o símbolo associado àquele estado.

Se a probabilidade condicional $P[X_n = s_j \mid X_{n-1} = s_i]$ for independente do tempo n, chamamos a cadeia de Markov de homogênea. Todas as cadeias de Markov a partir daqui são homogêneas. Nesse caso temos que $P_{ij} = P[X_n = s_j \mid X_{n-1} = s_i]$ é independente de n e então podemos organizar os P_{ij} numa matriz de transição de probabilidades.

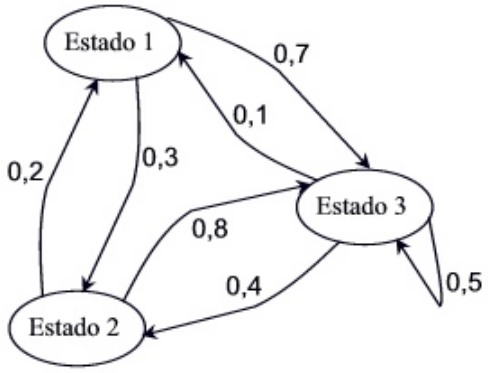

Figura 2.1: Exemplo de cadeia de Markov

Uma cadeia de Markov fica totalmente determinada pela matriz $A = \{a_{ij} = P[X_n = s_j \mid X_{n-1} = s_i]\}$ e pelo vetor π de probabilidades iniciais. Denotamos os parâmetros da cadeia de Markov por $\theta = \{A, \pi\}$.

Exemplo 1: Seja uma partícula que a cada unidade de tempo está em um entre três locais distintos (estados 1, 2 e 3). O novo estado da partícula depende somente do estado atual de acordo com a matriz de probabilidades

$$A = \begin{bmatrix} 0 & 0,3 & 0,7 \\ 0,2 & 0 & 0,8 \\ 0,1 & 0,4 & 0,5 \end{bmatrix}$$

Todos os elementos dessa matriz estão entre 0 e 1 e a soma dos elementos de cada linha é 1.

Esta é uma cadeia de Markov homogênea. A matriz de transição de probabilidades também pode ser expressa no grafo da figura 2.1.

Exemplo 2: Na figura 2.2 temos um exemplo simples de modelagem do clima com cadeias de Markov. O cálculo de $P(X|\theta)$ e a obtenção dos parâmetros θ do modelo a partir do treinamento é muito simples.

Vamos supor, por exemplo, que $X = S, N, C, C, N, S, S, N$ (onde S =sol, C =chuva e N =nublado). Como o estado futuro só depende do estado atual, temos que:

$$P(X|\theta) = \pi_S A_{SN} A_{NC} A_{CC} A_{CN} A_{NS} A_{SS} A_{SN}$$

Dado a sequência de observação X acima, podemos estimar os parâmetros da cadeia de Markov $\theta = \{A, \pi\}$ com as fórmulas:

$$\begin{cases} a_{ij} = P[X_n = s_j \mid X_{n-1} = s_i] = \frac{\text{n}^0 \text{ de transições de } s_i \text{ para } s_j}{\text{n}^0 \text{ de vezes no estado } s_i} \\ \pi_i = 1, \text{ se } X_1 = s_i \text{ e } \pi_i = 0, \text{ caso contrário.} \end{cases}$$

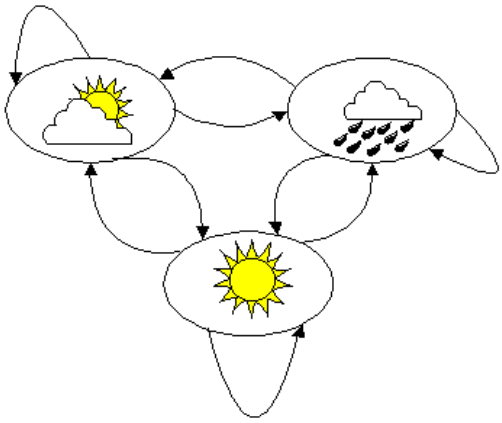

Figura 2.2: Cadeia de Markov para modelagem do clima

Daí temos que a seguinte estimativa para o modelo:

$$\begin{cases} A_{SS} = \frac{1}{3}, & A_{SC} = \frac{0}{3}, & A_{SN} = \frac{2}{3} \\ A_{CS} = \frac{0}{2}, & A_{CC} = \frac{1}{2}, & A_{CN} = \frac{1}{2} \\ A_{NS} = \frac{1}{2}, & A_{NC} = \frac{1}{2}, & A_{NN} = \frac{0}{2} \\ \pi_S = 1, & \pi_C = 0, & \pi_N = 0 \end{cases}$$

Note que como o tamanho da sequência X é curto, a estimativa não é confiável.

2.2 Hidden Markov Models (HMM)

Como já foi comentado, para modelar as características naturais de um sinal de voz ou de escrita à mão, a abordagem tradicional na área de reconhecimento de voz (ou escrita) tem sido utilizar HMMs, que descreveremos agora. Um Modelo Oculto de Markov (Hidden Markov Model - HMM) é um modelo de Markov (cadeia de Markov) onde os estados do modelo não são conhecidos, mas apenas o sinal emitido em cada um dos estados. Um exemplo de HMM, obtido em [6] (assim como as figuras 2.3, 2.4 e 2.5), está descrito a seguir: Considere urnas com bolas pretas, brancas e cinzas como na figura 2.3. Vamos supor que a mão que pega a bola nas urnas segue um processo de Markov. Temos conhecimento das bolas selecionadas (como na figura 2.4), mas não de que urna essas bolas foram retiradas. Esse processo é modelado por um HMM cujos parâmetros são as probabilidades iniciais e de transição da cadeia de Markov e ainda a probabilidade de retirarmos bolas pretas, brancas ou cinzas em cada um dos estados. Representamos graficamente esse HMM de 3 estados (urnas) e 3 símbolos de saídas (tipos de bolas em cada urna) na figura 2.5.

Vamos tratar somente de HMMs com número de estados e de unidades de tempo (tamanho da observação) finitos. O sinal O emitido em dada estado pode assumir os valores definidos pelo alfabeto, que inicialmente assumimos como sendo finito. Na última subseção apresentamos brevemente o caso do sinal O ser contínuo.

Um outro exemplo de HMM é uma partícula que para cada $t = 1, 2, \ldots, T$ muda para um local (ou estado) entre os N possíveis. Em cada um dos estados, a partícula emite um sinal (de

Figura 2.3: Urnas com bolas pretas, brancas e cinzas

Figura 2.4: Bolas obtidas

um alfabeto de tamanho M). Temos que o estado futuro da partícula depende somente do estado atual, e não dos estados anteriores ou do tempo t. Não conhecemos os estados da partícula, mas somente os sinais emitidos por ela. Abaixo temos algumas notações.

- N é o número de estados do modelo, ou locais onde a partícula pode ir. Para simplificar a notação denominamos o conjunto de estados $\mathcal{S} = \{1, 2, \ldots, N\}$.
- M é o número total de símbolos distintos, o tamanho do alfabeto de sinais que a partícula emite. $V = \{v_1, v_2, \ldots, v_M\}$ é o alfabeto.
- T é a quantidade de unidades de tempo (tamanho da observação).
- $Q = \{q_1, q_2, \ldots, q_T\}$ onde q_t é o estado do modelo no tempo t.
- $O = \{O_1, O_2, \ldots, O_T\}$ onde O_t é símbolo observado no tempo t.
- $\pi = \{\pi_i\}, i = 1, \ldots, N.$ onde $\pi_i = P[q_1 = i]$ é a probabilidade de i ser o estado inicial do experimento.
- $A = \{a_{ij}\}$ é uma matriz $N \times N$, onde $a_{ij} = P[q_{t+1} = j \mid q_t = i]$ é a probabilidade da partícula ir do estado i para o estado j. Os a_{ij}'s são independentes do tempo t.
- $B = \{b_i(k)\}$ é uma matriz $N \times M$,onde $b_i(k)$ é a probabilidade do símbolo v_k ser observado no estado i.

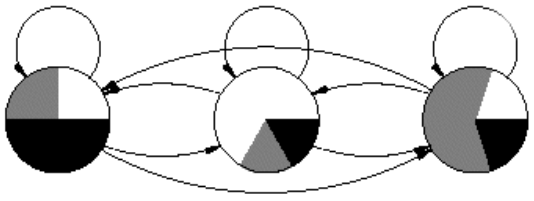

Figura 2.5: Representação do HMM com 3 estados e 3 símbolos.

- $\lambda = (A, B, \pi)$ é a notação compacta para um HMM.

Nas subseções seguintes apresentamos algoritmos para os três principais problemas de HMM na maioria das aplicações que são calcular: (1) $P[O \mid \lambda]$, a probabilidade da ocorrência da observação O dado o modelo λ. (2) $\arg\max_{Q} P[Q \mid O, \lambda]$, a sequência de estados mais provável dado a observação O e o modelo $\lambda = (A, B, \pi)$ e (3) $\arg\max_{\lambda} P[O \mid \lambda]$, quais os parâmetros do modelo $\lambda = (A, B, \pi)$ a partir de dados de treinamento O.

2.2.1 Cálculo de $P[O \mid \lambda]$

- *Dado um modelo $\lambda = (A, B, \pi)$ como calcular $P[O \mid \lambda]$, a probabilidade da ocorrência da observação $O_1, O_2, \ldots, O_T$?, isto é, $P[O \mid \lambda] =$?*

O modo mais imediato de se calcular $P[O \mid \lambda]$ é achar $P[O \mid Q, \lambda]$ para um estado fixado $Q = q_1, q_2, \ldots, q_T$ e somar as probabilidades sobre todos os estados possíveis, isto é:

$$P[O \mid \lambda] = \sum_{Q} P[O, Q \mid \lambda] = \sum_{Q} P[O \mid Q, \lambda] P[Q, \lambda]$$

onde $P[O \mid Q, \lambda] = b_{q_1}(O_1) b_{q_2}(O_2) \cdots b_{q_T}(O_T)$ e $P[Q, \lambda] = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T}$.

Para cada estado temos $2T - 1$ multiplicações para o cálculo de $P[O \mid Q, \lambda] P[Q, \lambda]$ e temos N^T estados. Daí temos na ordem de $2TN^T$ multiplicações para o cálculo de $P[O \mid \lambda]$. Isso é inviável computacionalmente pois, por exemplo, um modelo com 10 estados e 100 instantes de tempo (observações), isto é, $N = 10, T = 100$, temos ordem de 10^{102} multiplicações. Para executar esse cálculo mais rapidamente usamos o procedimento abaixo.

Algoritmo Forward

A variável forward $\alpha_t(i)$ é a probabilidade da observação da sequência parcial $O_1, O_2, \ldots, O_t$ e que no tempo t tenhamos o estado i.

$$\alpha_t(i) = P[O_1, O_2, \ldots, O_t, q_t = i \mid \lambda]$$

$\alpha_t(i)$ pode ser calculada recusirvamente da seguinte forma:

$$\begin{cases} \alpha_1(i) = \pi_i b_i(O_1) & i = 1, \ldots, N \\ \alpha_{t+1}(j) = b_j(O_{t+1}) \left(\sum_{i=1}^{N} \alpha_t(i)\, a_{ij} \right) & t = 1, 2, \ldots, T-1 \quad , \quad j = 1, \ldots, N \end{cases}$$

Temos que

$$P[O \mid \lambda] = \sum_{i=1}^{N} \alpha_T(i)$$

Esse método de cálculo de $P[O \mid \lambda]$ envolve ordem de $N^2 T$ multiplicações.

A dedução da fórmula $\alpha_{t+1}(j) = b_j(O_{t+1})\left(\sum_{i=1}^{N}\alpha_t(i)\,a_{ij}\right)$ deve ser feita com o devido cuidado pois $P[AB] = P[A]P[B]$ só se A e B são independentes.

$\alpha_{t+1}(j) = P[O_1,\ldots,O_t,O_{t+1},q_{t+1}=j \mid \lambda] =$

$\sum_{i=1}^{N} P[O_1,\ldots,O_t,O_{t+1},q_{t+1}=j \mid q_t=i,\lambda]P[q_t=i \mid \lambda] =$

$\sum_{i=1}^{N} P[O_1,O_2,\ldots,O_t \mid q_t=i,\lambda]P[q_t=i \mid \lambda]P[O_{t+1},q_{t+1}=j \mid q_t=i,\lambda] =$

$\sum_{i=1}^{N} P[O_1,\ldots,O_t,q_t=i \mid \lambda]P[O_{t+1} \mid q_{t+1}=j,q_t=i,\lambda]P[q_{t+1}=j \mid q_t=i,\lambda] = b_j(O_{t+1})\left(\sum_{i=1}^{N}\alpha_t(i)\,a_{ij}\right)$

O cálculo da variável forward com o algoritmo descrito acima envolve problema de precisão numérica (underflow) pois, por exemplo, se temos um alfabeto de tamanho 100 e 100 observações ($T = 100$), α_{t+1} é da ordem $T = 100$ vezes menor que α_t. Daí temos que $P[O \mid \lambda]$ é da ordem de 10^{-200}.

Para resolver este problema temos que escalar esse algoritmo. Esse escalamento consiste basicamente em cada passo do algoritmo criar uma variável auxiliar que indica o valor de $\sum_i \alpha_t(i)$.

Algoritmo Forward Escalado

O cálculo da variável forward com o algoritmo descrito na subseção acima envolve problema de precisão numérica (underflow) pois, por exemplo, se temos um alfabeto de tamanho 100 e 100 observações ($T = 100$), α_{t+1} é da ordem $T = 100$ vezes menor que α_t. Daí temos que $P[O \mid \lambda]$ é da ordem de 10^{-200}.

Para resolver este problema temos que escalar esse algoritmo. Esse escalamento consiste basicamente em cada passo do algoritmo criar uma variável auxiliar que indica o valor de $\sum_i \alpha_t(i)$.

Descrevemos a seguir o algoritmo forward escalado.

1. Inicialização:

$\widetilde{\alpha}_1(i) = \pi_i b_i(O_1), i = 1\cdots N$

$c_1 = 1/\left(\sum_{i=1}^{N}\widetilde{\alpha}_1(i)\right)$

$\widehat{\alpha}_1(i) = c_1\widetilde{\alpha}_1(i), i = 1\cdots N$ (escalado)

2. Indução

$\widetilde{\alpha}_{t+1}(j) = b_j(O_{t+1})\left(\sum_{i=1}^{N}\widehat{\alpha}_t(i)\,a_{ij}\right), t = 1,\ldots,T-1 \quad , \quad j = 1,\ldots,N$

$c_{t+1} = 1/\left(\sum_{i=1}^{N}\widetilde{\alpha}_{t+1}(i)\right), t = 1,\ldots,T-1$

$\widehat{\alpha}_{t+1}(j) = c_{t+1}\widetilde{\alpha}_{t+1}(i), t = 1,\ldots,T-1 \quad , \quad j = 1,\ldots,N$ (escalado)

3. Finalização

$P[O \mid \lambda] = 1/\left(\prod_{t=1}^{T} c_t\right)$

isso é válido pois $\widehat{\alpha}_t(i) = \left(\prod_{\tau=1}^{t} c_\tau\right)\alpha_t(i)$ e $\sum_{i=1}^{N}\widehat{\alpha}_t(i) = 1$ logo $P[O \mid \lambda] = \sum_{i=1}^{N}\alpha_T(i) = \sum_{i=1}^{N}\widehat{\alpha}_T(i) / \left(\prod_{t=1}^{T} c_t\right) = \frac{1}{\prod_{t=1}^{T} c_t}\sum_{i=1}^{N}\widehat{\alpha}_T(i) = 1/\left(\prod_{t=1}^{T} c_t\right)$

Como $P[O \mid \lambda]$ pode ser muito pequeno, calculamos o seu logaritmo:

$$\log P[O \mid \lambda] = -\sum_{t=1}^{T} \log c_t$$

2.2.2 Cálculo da Sequência Ótima de Estados

- *Dado um modelo $\lambda = (A, B, \pi)$ e a sequência de observação $O_1, O_2, \ldots, O_T$. Qual a sequência de estados $Q = (q_1, q_2, \ldots, q_T)$ para que $P[Q \mid O, \lambda]$ seja maximizada?, isto é,* $\arg\max_Q P[Q \mid O, \lambda] =$?

Como $P[Q \mid O, \lambda] = \frac{P[Q,O|\lambda]}{P[O|\lambda]}$ e $P[O|\lambda]$ é constante em relação à seqüência de estados Q, basta calcular:

$$Q^* = \arg\max_Q P[Q, O \mid \lambda]$$

Para executar o cálculo acima de modo eficiente usamos o algoritmo de Viterbi, que é um algoritmo de programação dinâmica e é muito parecido com o algoritmo descrito para o cálculo de $P[O \mid \lambda]$. Para o cálculo do algoritmo de Viterbi precisamos da variável auxiliar $\delta_t(i)$ definida por:

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P[q_1, q_2, \ldots, q_t = i, O_1, \ldots, O_t \mid \lambda]$$

Podemos calcular $\delta_{t+1}(i)$ a partir da seguinte relação indutiva:

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] b_j(O_{t+1})$$

onde $\delta_1(i) = P[q_1 = i, O_1|\lambda] = \pi_i b_i(O_1)$.

O valor de $\max_Q P[Q, O \mid \lambda]$ pode ser calculado com a relação:

$$P^* = \max_Q P[Q, O \mid \lambda] = \max_i [\delta_T(i)]$$

Temos ainda que traçar o caminho de volta para achar a seqüência de estados:

$$q_T^* = \arg\max_i [\delta_T(i)] \text{ e } q_t^* = \arg\max_i [\delta_t(i) a_{i q_{t+1}^*}]$$

Como no caso do algoritmo forward, é necessário algumas modificações para o cálculo escalado do algoritmo de Viterbi, o que evita underflow em computadores de precisão finita. Para maiores detalhes consulte [2].

2.2.3 Treinamento de HMM

- *Quais os parâmetros* $\lambda = (A, B, \pi)$ *de um HMM para que* $P[O \mid \lambda]$ *seja maximizado?, isto é,* $\arg\max_{\lambda} P[O \mid \lambda] = ?$

O treinamento de um modelo estatístico (problema 3) é de longe o mais difícil e a existência de um algoritmo eficiente para esse problema é condição fundamental para a aplicabilidade desse modelo estatístico. Esse é o caso de HMM, pois existe o eficiente algoritmo de Baum-Welch, que é um caso particular do algoritmo EM (Expectation-Maximization).

O treinamento de HMM é um problema de otimização contínua de várias variáveis (os parâmetros $\lambda = (A, B, \pi)$) e pode ser resolvido aplicando métodos de otimização baseados no gradiente. Uma opção melhor é o algoritmo de Baum-Welch (ou EM) que consiste dos seguintes passos:

1. Inicialização: Obter parâmetros $\lambda = (A, B, \pi)$ iniciais do modelo. Podemos fazer isso de modo aleatório ou usar algum tipo de estimativa para a inicialização do HMM.

2. Passo **E** (Expectation): Calcular as esperanças matemáticas das variáveis ocultas, dado λ. No caso de HMM, o número de vezes nos estados, transição dos estados e outras variáveis indicadas no passo **M**.

3. Passo **M** (Maximization): Calcular novos parâmetros $\hat{\lambda}$ para que $P[O \mid \hat{\lambda}]$ seja maximizado, assumindo os valores das variáveis ocultas no passo **E**. Se já conhecemos os valores dos estados ocultos (obtidos no passo **E**), o cálculo dos parâmetros λ do modelo para que $P[O \mid \lambda]$ seja máximo é simples (análogo para cadeias de Markov):

$$\begin{cases} a_{ij} = \frac{\text{número de transições do estado } i \text{ para o estado } j}{\text{número de vezes no estado } i} \\ b_j(k) = \frac{\text{número de vezes no estado } j \text{ com o símbolo } v_k}{\text{número de vezes no estado } j} \\ \pi_i = \text{número de vezes no estado } i \text{ no tempo } t = 1 \end{cases}$$

4. Iteração: Se $P[O \mid \hat{\lambda}] - P[O \mid \lambda]$ é maior que um valor fixo, ir para o passo **E** (com os novos parâmetros $\hat{\lambda}$). Se não, terminar a execução e retornar os novos parâmetros $\hat{\lambda}$.

Pode-se provar que $P[O \mid \hat{\lambda}] \geq P[O \mid \lambda]$ e que a sequência de modelos $\hat{\lambda}_i$ obtidos com o algoritmo EM converge para λ^*, um máximo local de $\arg\max_{\lambda} P[O \mid \lambda]$. O algoritmo de Baum-Welch converge localmente (mas não globalmente) para um máximo. No entanto, Baum-Welch é rápido e geralmente obtemos um bom máximo local.

Para maiores detalhes e a dedução das fórmulas explícitas para os parâmetros do HMM, assim como o Baum-Welch escalado e para observações contínuas, consulte [1] ou [2].

2.2.4 HMM para Observações Contínuas

Apesar de ser possível discretizar o sinal de voz com um algoritmo de quantização vetorial, uma melhor performance em reconhecimento de voz é obtida trabalhando diretamente com o sinal de voz contínuo. Para isso é necessário que o HMM lide com funções contínuas de densidade

de distribuição $b_i(O)$ (que define a densidade de distribuição do sinal emitido no estado i). Em reconhecimento de voz é comum usar a função densidade de distribuição $b_i(O)$ como uma soma de Gaussianas:

$$b_j(O) = \sum_{k=1}^{M} c_{jk} \mathcal{N}(O, \mu_{jk}, U_{jk})$$

onde c_{jk} é o coeficiente da k-ésima mistura do estado j e $\mathcal{N}$ é a função Gaussiana (ou distribuição normal) multidimensional definida por:

$$\mathcal{N}(X, \mu, U) = \frac{1}{(2\pi)^{d/2}|U|^{1/2}} \exp(-\frac{1}{2}(X-\mu)^T U^{-1}(X-\mu))$$

onde d é a dimensão, U é a matriz de covariância e μ é vetor média d-dimensional.

Os algoritmos para os 3 problemas de HMM contínuas são parecidos com o caso discreto. Para maiores detalhes, consulte [1] ou [2].

Capítulo 3

Sistemas de Reconhecimento de Voz

Este é o capítulo central desse texto. Nele vamos apresentar os 3 tipos básicos de reconhecimento de voz. Fazemos essa classificação de acordo com dois critérios:

- Palavras isoladas ou conectadas: Quando as palavras são isoladas (há uma pequena pausa entre elas), podemos separá-las e o sistema de reconhecimento de voz vai se ocupar com uma palavra de cada vez. Quando as palavras são conectadas, como na fala natural, o reconhecimento de voz se torna bem mais complexo pois (1) Não sabemos o número de palavras no sinal de voz. No máximo obtemos um limite inferior e superior para o número de palavras. (2) O fenômeno da co-articulação (interferência na produção do final de um som com o início do som seguinte) altera bastante o sinal de voz. (3) As fronteiras das palavras não são bem definidas, tornando difícil, ou mesmo impossível, especificar precisamente as fronteiras de uma palavra devido a co-articulação.

- Pequeno ou grande vocabulário: Quando o vocabulário é pequeno a unidade básica para a modelagem de HMM é a palavra. Já quando o vocabulário é grande (>1000 palavras), se torna impraticável que a unidade básica sejam as palavras. Necessitamos de unidades menores como o fonema, por exemplo.

Temos 3 tipos básicos de reconhecimento de voz que estudaremos nesse capítulo:

1. Reconhecimento de voz com pequeno vocabulário e palavras isoladas: É o tipo mais simples e é útil em aplicaçoes tipo comande-e-controle, onde é dito uma palavra por vez (com pausas entre os palavras do comando) e um tipo de ação é executado.

2. Reconhecimento de voz com pequeno vocabulário e palavras conectadas: Nesse caso já reconhecemos a fala natural conectada, mas com um pequeno vocabulário. Esse problema é bem mais complexo que o reconhecimento de voz com palavras isoladas.

3. Reconhecimento de voz com grande vocabulário (e palavras conectadas): É a forma mais geral de reconhecimento de voz. É onde tentamos reconhecer a fala natural. Nesse caso os algoritmos em geral não podem achar uma solução por busca exaustiva. É um problema de grande interesse nas pesquisas atuais em reconhecimento de voz.

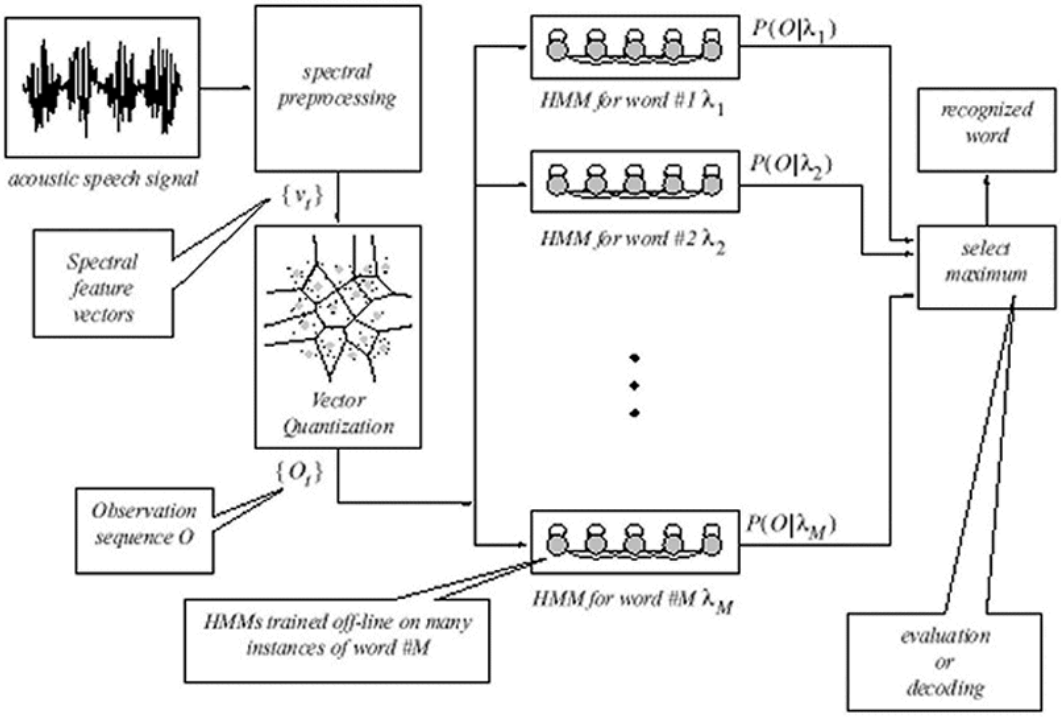

Figura 3.1: Sistema de Reconhecimento de Voz para pequeno vocabulário e palavras isoladas.

3.1 Pequeno Vocabulário e Palavras Isoladas

Vamos descrever um sistema bem simples de reconhecimento de voz para palavras isoladas e pequeno vocabulário baseado em [2] ou em [6]. A figura 3.1 (obtida em [6]) descreve esse sistema. Nesse sistema a entrada é o som de uma palavra X. Calculamos a probabilidade da ocorrência de X para os HMMs das palavras $1, 2, ..., M$ (já treinados). A palavra i escolhida pelo sistema é o do HMM em que obtemos a máxima probabilidade $P(X|HMM - i)$. Se a entrada é o som de várias palavras, o pré-processamento deve separá-las (pelo silêncio entre elas) pois o sistema só reconhece uma palavra por vez.

O treinamento do sistema (a obtenção dos parâmetros para cada um dos HMMs) é feito com o algoritmo de Baum-Welch e vários sons das palavras $1, 2, ..., M$ por vez. Essa arquitetura é bem geral (um sistema de reconhecimento de escrita com letras isoladas é feito de modo análogo).

3.2 Pequeno Vocabulário e Palavras Conectadas

O reconhecimento de voz com palavras conectadas e pequeno vocabulário é bem mais complexo que o de palavras isoladas pelas razões já expostas na seção 1. Vamos a seguir definir a notação usada para em seguida expor o algoritmo de programação dinâmica em dois níveis.

Note que a notação usada foi baseada em distâncias (como usada em [2]) é mais natural

para a técnica de DTW. A mudança da notação de distâncias para probabilidades (para aplicar HMM) é imediata e não será feito aqui.

3.2.1 Notação

Seja $\mathcal{T}$ o dado de entrada (sinal acústico) devidamente processado como uma sequência de vetores espectrais.

$$\mathcal{T} = \{t(1), t(2), ..., t(M)\}$$

onde o intervalo entre os frames dos vetores espectrais $t(i)$ é definida no processamento de voz (em torno de 20 a 50 ms).

Seja $\mathcal{R}$ o vocabulário, composto de V palavras

$$\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_V\}$$

onde cada $\mathcal{R}_i$ tem a duração de N_i frames, que denotamos por:

$$\mathcal{R}_i = \{r_i(1), r_i(2), ..., r_i(N_i)\}$$

O problema de reconhecimento de voz conectado consiste em achar a sequência ótima de palavras $\mathcal{R}^*$ do vocabulário que melhor se casa com o dado de entrada $\mathcal{T}$. Denotamos $\mathcal{R}^*$ por:

$$\mathcal{R}^* = \{\mathcal{R}_{q^*(1)} \oplus \mathcal{R}_{q^*(2)} \oplus ... \oplus \mathcal{R}_{q^*(l)}\}$$

onde $q^*(l)$ está entre 1 e V. Denotamos uma sequência qualquer de palavras por:

$$\mathcal{R}^s = \{\mathcal{R}_{q(1)} \oplus \mathcal{R}_{q(2)} \oplus ... \oplus \mathcal{R}_{q(L)}\} = \{r^s(n)\}_{n=1}^{N^s}$$

A distância entre $\mathcal{T}$ e $\mathcal{R}^s$ pode ser calculada da forma:

$$D(\mathcal{T}, \mathcal{R}^s) = \sum_{m=1}^{M} dist(t(m), r^s(m))$$

Com essa notação, dado um sinal de entrada $\mathcal{T}$, a distância mínima entre todos os $\mathcal{R}^s$ e $\mathcal{T}$ é:

$$D^* = \min_{\mathcal{R}^s} D(\mathcal{T}, \mathcal{R}^s)$$

e o problema de reconhecimento de voz conectado consiste em achar

$$\mathcal{R}^* = \arg\min_{\mathcal{R}^s} D(\mathcal{T}, \mathcal{R}^s)$$

3.2.2 Programação Dinâmica em dois níveis

O cálculo de D^* e $\mathcal{R}^*$ pelo método de força bruta é impraticável pois se o sinal de entrada $\mathcal{T}$ é composto de L palavras e temos um dicionário de V palavras, temos V^L possíveis sequências de palavras (sem contar o fato de que só temos estimativas do número mínimo e máximo de palavras de $\mathcal{T}$). Apresentamos a seguir o algoritmo de programação dinâmica em dois níveis, que é um dos vários algoritmos para o cálculo de D^* e $\mathcal{R}^*$.

O algoritmo de programação dinâmica em dois níveis, como o própio nome diz, quebra o cálculo em duas partes. Na primeira parte temos o cálculo da distância de uma porção arbitrária do dado de entrada $\mathcal{T}$ às palavras do vocabulário.

Sejam b e e os pontos inicial e final de um intervalo de $\mathcal{T}$. Devemos calcular a distância $\widehat{D}(v, b, e)$ desse intervalo a uma palavra v do vocabulário. Esse cálculo deve ser feito para todos os valores possíveis de b e e e para todas as palavras v do vocabulário, isto é, devemos calcular:

$$\widehat{D}(v, b, e) = \min \sum_{m=b}^{e} dist(t(m), r_v(m))$$

onde $1 \leq b < M, \qquad b < e \leq M, \qquad 1 \leq v \leq V.$

Após esse cálculo, fixados b e e, devemos achar para qual palavra v do vocabulário temos o menor valor entre as distâncias $\widehat{D}(v, b, e)$, isto é, devemos calcular:

$$\tilde{D}(b, e) = \min_{1 \leq v \leq V} \widehat{D}(v, b, e)$$

A palavra que dá melhor casamento no intervalo $[b, e]$ é:

$$\tilde{N}(b, e) = \arg\min_{1 \leq v \leq V} \widehat{D}(v, b, e)$$

O primeiro nível de cálculo consiste em achar a matriz de melhores valores $\tilde{D}(b, e)$. No segundo nível de cálculo utilizamos a programação dinâmica para calcular a menor distância para todo o dado de entrada $\mathcal{T}$. Utilizamos a seguinte recursão para esse cálculo.

$$\bar{D}_l(e) = \min_{1 \leq b < e} [\tilde{D}(b, e) + \bar{D}_{l-1}(b - 1)]$$

O cálculo deve ser feito para $l = 1$ até $l \leq L_{MAX}$, onde L_{MAX} é o número máximo de palavras contido no dado $\mathcal{T}$. A solução final é:

$$D^* = \min_{1 \leq l \leq L_{MAX}} \bar{D}_l(M)$$

3.3 Grande Vocabulário

Vamos analisar somente reconhecimento de voz com grande vocabulário e palavras conectadas (reconhecimento de voz com grande vocabulário e palavras isoladas não tem interesse prático). Vamos usar o modelo Bayesiano já visto no capítulo 1 e que revisamos aqui. No modelo Bayesiano é necessário:

- Um conjunto de variáveis aleatórias $X = (A, W)$ onde A é o sinal observado, a evidência acústica, o dado de entrada de voz (processado) e W é uma sequência de palavras, são as variáveis ocultas que descrevem eventos causados por A.

  Assumimos que A é uma sequência de símbolos a_i (gerados no tempo):

  $$A = a_1, a_2, ..., a_m$$

  W é uma sequência de n palavras, cada uma pertence a um dicionário fixo (e finito) $\mathcal{W}$.

  $$W = w_1 w_2 ... w_n \qquad w_i \in \mathcal{W}$$

- Um modelo estocástico, no nosso caso HMM, que permite descrever a variabilidade e o ruído no sinal.

- Parâmetros específicos θ do HMM que melhor descrevem a classe dos sinais (voz) que tentamos decifrar.

O problema básico em reconhecimento de voz é obter W a partir de A. No capítulo 1 vimos que podemos obter, segundo o modelo, qual a sequência de palavras mais provável W^* com a entrada acústica A:

$$W^* = \arg\max_W P(A|W, \theta)P(W|\theta)$$

Omitindo na fórmula acima os parâmetros θ do modelo temos:

$$W^* = \arg\max_W P(A|W)P(W) \tag{3.1}$$

Nas próximas seções descrevemos brevemente os três componentes principais de um sistema de reconhecimento de voz, que são:

1. Modelagem Acústica, onde modelamos o sinal de voz e calculamos $P(A|W)$, a probabilidade da ocorrência da acústica A dado W.

2. Modelagem da Linguagem, onde modelamos a linguagem e calculamos $P(W)$.

3. Hipóteses de Busca, que trata dos métodos para o cálculo da fórmula 3.1.

3.4 Modelagem Acústica

Na modelagem acústica tratamos de modelar a acústica do sinal de voz para calcular $P(A|W)$, a probabilidade da ocorrência da acústica A dado W.

Quando temos um grande vocabulário (de 1000 a 20000 palavras) é inviável ter palavras como a unidade básica para modelamento com HMM. É necessário modelar (com HMM) uma unidade menor que a palavra. As principais razões para isso são: (1) O treinamento de modelos de palavras inteiras requer um conjunto de treinamento com um número suficientemente grande de amostras de cada palavra. (2) O modelamento da influência do contexto entre palavras (palavras

ditas antes ou depois alteram o som da própria palavra) ou inter-palavras (influência dos sons consecutivos dentro das palavras) requer um número muito grande de possibilidades a serem analisadas.

Há várias possibilidades de unidades menores que a palavra, entre elas:

- Sílabas ou Semi-sílabas: Usamos a noção linguística de sílaba (um núcleo vogal com consoantes (ou agrupamento de consoantes) opcionais antes ou depois da vogal) para modelar as unidades baseadas em similaridade acústica. O problema é que há um número grande de sílabas em uma língua (em inglês há aproximadamente 10.000 sílabas). Outra opção é usarmos semi-sílabas, que são partes de uma sílaba. Em inglês há aproximadamente 2.000 semi-sílabas.

- Bases Sintéticas: Podemos usar unidades sintéticas, que são completamente auto-organizadas a partir dos dados. Não é necessário nenhum conceito fonético ou linguístico e é uma ferramenta útil para modelar novas palavras que não pertencem ao vocabulário.

- Fonemas: Existem em torno de 50 fonemas na língua inglesa. É a unidade mais usada para reconhecimento de voz e é a única que discutiremos aqui.

3.4.1 Modelos Acústicos Fonéticos

Para se construir HMMs para palavras devemos:

1. Construir um dicionário fonético para todas as palavras do vocabulário. Isto é, obter os fonemas (com base na pronúncia) de cada palavra v do vocabulário, obtendo a sequência $\Phi(v) = \varphi_1, \varphi_2, ..., \varphi_n$, onde φ_i pertence a um alfabeto fonético pré-determinado. Chamamos $\Phi(v)$ de base fonética de v. Para palavras com pronúncias diferentes devemos ter múltiplas bases fonéticas. Existem vários alfabetos e dicionários fonéticos como o ISA (International Phonetic Alphabet) ou o ARPABET (da DARPA, agência de fomento americana que financiou várias pesquisas em reconhecimento de voz). Por exemplo, num alfabeto essencialmente idêntico ao ARPABET, temos a seguinte transcrição das palavras:

   above - ax/ b/ ah/ v

   carry - k/ ae/ r/ iy

   hours - aw/ w/ axr/ z

2. Construir um HMM para cada símbolo do alfabeto fonético. Em reconhecimento de voz é comum utilizar a estrutura left-right, onde as únicas transições permitidas são para o próprio estado e para o estado seguinte, como vemos na figura 3.2. Comumente usa-se de 3 a 5 estados para se modelar um fonema.

3. O HMM de uma palavra v é a concatenação dos HMMs dos fonemas dessa palavra ($\Phi(v) = \varphi_1, \varphi_2, ..., \varphi_n$), como vemos na figura 3.3.

4. O HMM da sentença inteira $W = w_1, w_2, ..., w_n$ é a concatenação dos HMMs das palavras $w_1, w_2, ..., w_n$, como vemos na figura 3.4.

Figura 3.2: HMM letf-right com 5 estados

Figura 3.3: Modelo de uma palavra - concatenação de HMMs dos fonemas

5. Para o treinamento dos fonemas usamos o algoritmo de Baum-Welch. Os dados de treinamento são a leitura por usuários de um texto preparado W. Vamos treinar o HMM composto relativo a todo o texto W, levando em conta que o modelo de um fonema específico aparece em vários lugares de W. As transições de um mesmo fonema tem de ser considerados a mesma transição, não importa a sua posição em W.

3.4.2 Definição Alternativa de HMM

Uma definição alternativa de HMM usada em reconhecimento de voz (principalmente por pesquisadores da IBM, veja [1]) é a HMM com saída nas transições e transições nulas. HMM com saída nas transições quer dizer que os estados não emitem nenhum tipo de sinal e as observações são emitidas nas transições entre os estados. Uma transição nula não emite nemhuma observação e é denotada por uma linha tracejada. A utilidade dessa definição alternativa de HMM pode ser vista na figura 3.5 onde modelamos um fonema com pronúncias diferentes e a opção de omissão de alguns estados.

Podemos também ter um estado inicial onde o HMM sempre começa, evitando assim a necessidade do vetor π como parâmetro do HMM.

3.5 Modelagem da Linguagem

É claro que em uma linguagem certas palavras são muito mais prováveis que outras e que a probabilidade de ocorrência de certas palavras depende das palavras anteriores. Temos que levar

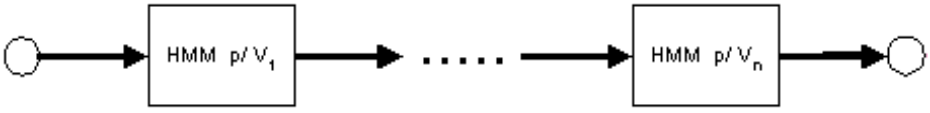

Figura 3.4: Modelo de uma sentença - concatenação de HMMs de palavras

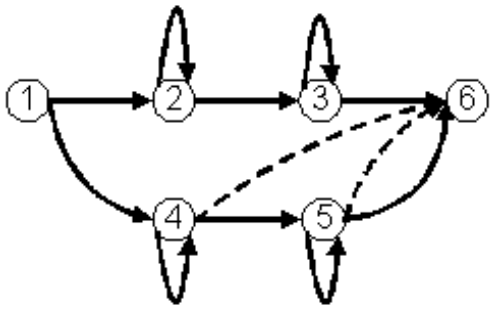

Figura 3.5: HMM de um fonema usando definição alternativa

em conta esses fatos e modelar a linguagem para calcularmos $P(W)$.

Seja $W = w_1, w_2, ..., w_n$. Pelo teorema de Bayes podemos decompor $P(W)$ em:

$$P(W) = \prod_{i=1}^{n} P(w_i|w_1, w_2, ..., w_{i-1})$$

onde $P(w_i|w_1, w_2, ..., w_{i-1})$ é a probabilidade de w_i ser dito após as palavras $w_1, w_2, ..., w_{i-1}$, que são chamadas de história (de W).

Infelizmente o cálculo de $P(w_i|w_1, w_2, ..., w_{i-1})$ é muito caro computacionalmente e é razoável supor que, por exemplo, a quinta palavra de uma frase depende muito pouco da primeira palavra dita. Diante disso, temos os modelos de linguagem de unigramas, bigramas e trigramas que restringem a história passada. No modelo de linguagem de unigrama não levamos em conta a história (as palavras ditas anteriormente), isto é,

$$P(W) = \prod_{i=1}^{n} P(w_i) = P(w_1)P(w_2)P(w_3)...P(w_n)$$

O modelo de linguagem de bigramas leva em conta somente a última palavra:

$$P(W) = P(w_1)\prod_{i=2}^{n} P(w_i|w_{i-1}) = P(w_1)P(w_2|w_1)P(w_3|w_2)...P(w_n|w_{n-1})$$

Já no modelo de linguagem de trigramas levamos em conta as duas últimas palavras:

$$P(W) = P(w_1)P(w_2|w_1)\prod_{i=3}^{n} P(w_i|w_{i-1}, w_{i-2})$$

Em sistemas de reconhecimentos de voz atuais praticamente é usado somente o modelo de linguagem de trigramas, que permanece no estado da arte. Modelos de linguagem de unigramas ou bigramas perdem muito em performance e modelos mais sofisticados, que levam em conta três ou mais palavras da história tem uma complexidade muito alta.

Para estimar a probabilidade de ocorrência de trigramas devemos calcular a frequência relativa para todos os trigramas do dicionário:

$$P(w_3|w_1, w_2) = f(w_3|w_1, w_2) = \frac{C(w_1, w_2, w_3)}{C(w_1, w_2)} \tag{3.2}$$

onde $C(w)$ indica o número de vezes que a palavra w ocorre no texto de treinamento e $f(.)$ indica a frequência relativa.

Um problema na fórmula acima é que mesmo para textos grandes muitos trigramas nunca ocorrem, o que acarreta que textos W que possuem esse trigrama terão probabilidade $P(W) = 0$. Um exemplo desse fato é o experimento conduzido por pesquisadores da IBM na década de 70: Um texto baseado em um dicionário de 1.000 palavras foi dividido em um subconjunto de testes e outro de treinamento (de 300.000 e 1.500.000 de palavras, respectivamente). Foi verificado que 23% dos trigramas do subconjunto de testes não apareceram no subconjunto de treinamento. Isto se deve ao fato de que em um dicionário de 1.000 plavras existem $1000^3 = 1$ bilhão de trigramas. Para calcular a probabilidade de ocorrência de trigramas pela frequência relativa de sua ocorrência, como na fórmula 3.2, o tamanho do texto de treinamento deve ser muito grande, o que é inviável.

Para se resolver esse problema é necessário suavizar as frequências dos trigramas. Um modo simples de se fazer isso é interpolar as frequências relativas dos unigramas, bigramas e trigramas:

$$P(w_3|w_1, w_2) = \lambda_3 f(w_3|w_1, w_2) + \lambda_2 f(w_3|w_2) + \lambda_1 f(w_3) \tag{3.3}$$

com $\lambda_1 + \lambda_2 + \lambda_3 = 1$ e os λ_i são positivos.

Um modelo de linguagem operando de acordo com a fórmula 3.3 pode ser modelado por um HMM e diante disso podemos calcular os valores λ_1, λ_2 e λ_3 com o algoritmo de Baum-Welch. No entanto, existem métodos mais eficientes do que a abordagem padrão por HMM para o cálculo de λ_1, λ_2 e λ_3, como vemos em [1].

Devemos separar o dado de treinamento em duas partes. A primeira parte (muito maior) é usada para estimar as frequências relativas e a segunda parte (muito menor) é usada para estimar os parâmetros λ_1, λ_2 e λ_3.

Um exemplo do modelo de linguagem de trigramas operando de acordo com a fórmula 3.3 pode ser visto em [1], onde foi calculado qual a ordem (indicada entre parênteses) no vocabulário de cada uma das palavras na frase abaixo:

- We(9) need(7) to(1) resolve(98) all(9) the(3) important(1641) issues(9) within(66) the(1) next(1) two(2) days(7).

3.6 Hipóteses de Busca

Nesta seção vamos atacar o principal problema de reconhecimento de voz, que é achar qual a sentença W^* mais provável dado a evidência acústica A, que num modelo Bayesiano pode ser calculado com a fórmula 3.1. Existem dois métodos principais para esse cálculo: a busca de Viterbi e busca em árvore, que descrevemos a seguir.

3.6.1 Algoritmo de Viterbi

O algoritmo de Viterbi, já discutido no capítulo 1 e que revisamos aqui, calcula qual a sequência de estados mais prováveis Q^* de um HMM de parâmentros λ e dado a evidência acústica A.

$$Q^* = \arg\max_Q P[Q, A \mid \lambda]$$

O algoritmo de Viterbi é baseado em programação dinâmica onde usamos a variável auxiliar $\delta_t(i)$ definida por:

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} P[q_1, q_2, \ldots, q_t = i, A_1, \ldots, A_t \mid \lambda]$$

Podemos calcular $\delta_{t+1}(i)$ a partir da seguinte relação indutiva:

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] b_j(A_{t+1}) \tag{3.4}$$

onde $\delta_1(i) = P[q_1 = i, A_1 | \lambda] = \pi_i b_i(A_1)$.

Podemos calcular qual a sequência ótima de estados até o tempo t com a relação:

$$P_t^* = \max_s [\delta_t(s)] \tag{3.5}$$

O valor de $\max_Q P[Q, A \mid \lambda]$ pode ser calculado com a relação:

$$P^* = \max_Q P[Q, A \mid \lambda] = \max_i [\delta_T(i)]$$

E traçamos o caminho de volta para achar a seqüência de estados:

$$q_T^* = \arg\max_i [\delta_T(i)] \quad e \quad q_t^* = \arg\max_i [\delta_t(i) a_{i q_{t+1}^*}]$$

3.6.2 Busca de Viterbi

A busca de Viterbi consiste em achar o melhor caminho num grafo (que depende do modelo de linguagem) com o algoritmo de Viterbi.

No modelo mais simples de unigramas, a busca de Viterbi consiste em achar o caminho mais provável no HMM da figura 3.6. Note a utilidade de modelagem por HMM pois a complexa tarefa de achar a sentença W^* mais provável dado a evidência acústica A é reduzida ao algoritmo de Viterbi em um (grande) HMM.

No modelo de linguagem de bigramas, a busca de Viterbi consiste em achar o caminho mais provável no HMM composto. Na figura 3.7 temos esse HMM composto para um vocabulário de duas palavras.

Para o modelo de linguagem de trigramas, o HMM composto fica mais complexo, com número de estados proporcional ao quadrado do tamanho do dicionário $|\mathcal{W}|$. Em [1] temos a figura do HMM composto para trigramas.

O problema da busca de Viterbi é que o número de estados é nuito grande (proporcional a $|\mathcal{W}|$ para modelos de linguagem de unigramas ou bigramas e proporcional a $|\mathcal{W}|^2$ para trigramas). Supondo que temos em média 20 estados para a modelagem de uma palavra e o tamanho do dicionário é $|\mathcal{W}| = 10.000$, temos 200.000 estados para unigramas (ou bigramas) e 2.10^9 estados para trigramas.

Um método simples e funcional (beam search) é estabelecer um limiar (threshold) no algoritmo de Viterbi, isto é, todos os caminhos que no tempo t e no estado i tem probabilidade menor que o limiar são considerados pouco prováveis e são eliminados (assim como continuações desse caminho). O limiar é calculado com:

$$\tau_i = \frac{P_t^*}{K}$$

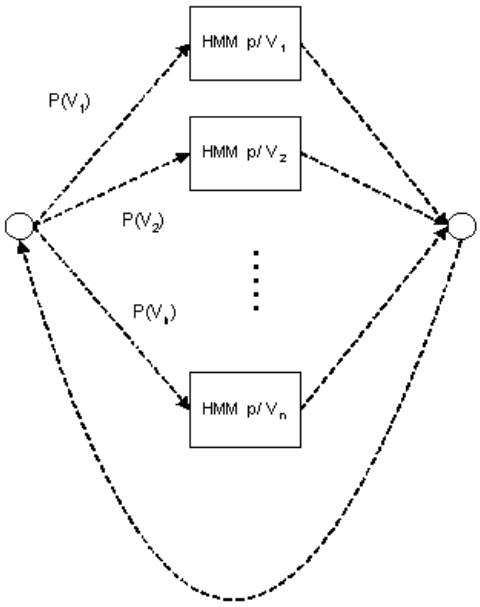

Figura 3.6: HMM composto para modelo de linguagem de unigramas

onde P_t^* é calculado com a fórmula 3.5 e K é uma constante determinada experimentalmente. $K = 100$ é um valor comum.

Essa estratégia reduz drasticamente o número de estados no cálculo da recursão da fórmula 3.4. É claro que nesse caso não há garantia que vamos obter realmente o melhor caminho no HMM composto (figura 3.6 ou 3.7). Esta estratégia funciona bem para modelos de linguagem de unigramas ou bigramas. Para modelos de linguagem de trigramas, devido ao grande número de estados são necessárias estratégias adicionais como processamento em vários passos para acelerar o cálculo, como pode ser visto em [1].

3.6.3 Busca em árvore

Podemos achar a sentença W^* mais provável dado a evidência acústica A com uma busca em árvore. Como W é uma sequência de palavras $w_1, w_2, ..., w_n$, esse problema consiste em achar o melhor caminho na árvore cujos ramos (em cada nível) são as palavras do vocabulário (na ordem de $|\mathcal{W}| = 10.000$).

Essa árvore tem um número muito grande de ramos (no nível i temos $|\mathcal{W}|^i$ ramos) e portanto é necessário uma estratégia (como foi feito na subseção anterior) para eliminar ramos pouco prováveis. Apresentamos a seguir o algoritmo A*, que é uma estratégia de busca em árvore.

Toda busca em árvore é baseada na maximização de um critério de avaliação g. Seja $w_1^n = w_1, w_2, ..., w_n$ uma sequência de palavras que maximiza $g(.)$, com $g(w_1^k)$ definida para $k = 1, 2, ..., n$.

Vamos definir uma função auxiliar F tal que se $F(w_1^n) \geqslant F(w_1^k)$ então $g(w_1^n) \geqslant g(w_1^k || z_1^{n-k})$ para todo z_1^{n-k} (o símbolo $||$ indica a concatenação das sequências w_1^k e z_1^{n-k}). Daí temos que se $F(w_1^n) \geqslant F(w_1^k)$ então nenhuma continuação do caminho w_1^k pode ser melhor que w_1^n. Nesse caso todas as continuações de w_1^k podem ser completamente eliminadas como candidatas a melhor

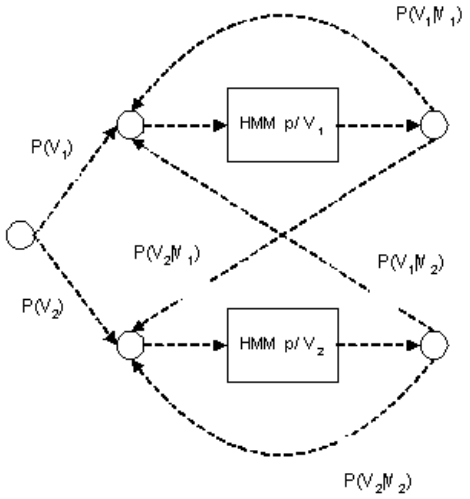

Figura 3.7: HMM composto para modelo de linguagem de bigramas com vocabulário de duas palavras

caminho.

O algoritmo A* é implementado usando estruturas de pilha. Qual a função F usada para reconhecimento de voz assim como algoritmos de multi-pilhas e o algoritmo de Fast Match, que são usados para acelerar a busca podem ser vistos em [1].

Capítulo 4

Entropia

Neste capítulo, que é independente dos demais, apresentamos o conceito de entropia na física, assim como na teoria da informação de Shannon. Mostramos aplicações da entropia no problema de codificação de símbolos e em árvores de decisão. Apresentamos também o conceito de entropia condicional, informação mútua, divergência e o princípio de máxima entropia.

4.1 Segunda Lei da Termodinâmica

A ciência da termodinâmica começou na primeira parte do século XIX com o estudo dos gases e na tentativa de melhorar a eficiência de máquinas a vapor, mas seus conceitos são válidos para sistemas físicos de qualquer natureza. A termodinâmica é uma ciência experimental, baseada em um pequeno número de princípios, que são generalizações feitas a partir da experiência. Ela diz respeito somente a propriedades macroscópicas ou em grande escala da matéria. A termodinâmica desempenha um papel fundamental da física e continua válida mesmo com os avanços da física no século XX.

Na primeira parte do século XIX pesquisadores observaram que vários processos ocorrem espontaneamente. Por exemplo, se dois corpos com temperaturas diferentes são colocados em contato, esses corpos vão chegar a uma temperatura em comum. O contrário (dois corpos com a mesma temperatura são colocados em contato e um corpo fica mais quente e o outro mais frio) nunca acontece. A segunda lei da termodinâmica determina em que sentido um processo ocorre. Seu enunciado conciso é: *A entropia do universo sempre aumenta.*

A segunda lei da termodinâmica diz que em qualquer processo que ocorra (no tempo) em um sistema isolado (que não troca com o exterior massa nem energia), a entropia não diminui. É a única das leis físicas que determina uma direção à seta do tempo e é de importância fundamental na física, como podemos observar pelas palavras dos físicos Arthur Eddington e Albert Einstein.

"The law that entropy always increases – the second law of thermodynamics – holds I think, the supreme position among the laws of Nature. If someone points out to you that your pet theory of the universe is in disagreement with Maxwell's equations - then so much worse for Maxwell equations. If it is found to be contradicted by observation - well these experimentalists do bungle things sometimes. But if your theory is found to be against the second law of Thermodynamics, I can give you no hope; there is nothing for it but to collapse in deepest humiliation". Arthur Eddington[12].

"[A law] is more impressive the greater the simplicity of its premises, the more different are the kinds of things it relates, and the more extended its range of applicability. Therefore, the deep impression which classical thermodynamics made on me. It is the only physical theory of universal content, which I am convinced, that within the framework of applicability of its basic concepts will never be overthrown". Albert Einstein[13].

O conceito de entropia e a segunda lei da termodinâmica foram descobertos pelo físico Clausius, que definiu a variação de entropia como a variação do calor recebido sobre a temperatura.

$$dS = \frac{dQ}{T}$$

Quando colocamos em contato dois corpos de temperaturas T_1 e T_2, com $T_1 \geqslant T_2$, temos que o calor flui do corpo mais quente (tempetatura T_1) para o mais frio (o contrário nunca acontece). A variação de entropia nesse processo é sempre positiva pois:

$$dS = -\frac{dQ}{T_1} + \frac{dQ}{T_2} \geqslant 0$$

Baseado nesse tipo de argumento Clausius postulou, em 1865, a segunda lei da termodinâmica.

Da termodinâmica estatística temos a fórmula de Boltzmann, que define a entropia de um sistema físico:

$$S = k \log W$$

onde S denota a entropia. $k = 1,381.10^{-16}$ é a constante de Boltzmann e W é o número de microestados.

Os microestados são os possíveis valores que determinam o estado de cada partícula, por exemplo, a energia cinética. A mecânica quântica diz que W é um número inteiro e finito.

O próprio Boltzmann notou que a entropia mede a desorganização de um sistema físico. A identificação de entropia com informação foi clarificada com a explicação de Szilard para o paradoxo do demônio de Maxwell (descrita no próximo parágrafo) e com a teoria da informação de Shannon.

Maxwell propôs a sequinte experiência: Seja uma câmara isolada dividida em duas partes A e B separadas por uma partição que não conduz energia nem matéria. Essa partição tem uma portinhola móvel (sem atrito) que permite a passagem de partículas entre as câmaras. Em cada uma das partes temos amostras de um gás à mesma temperatura. A temperatura do gás é determinada por uma "média" estatística das velocidades das partículas do gás, mas em uma amostra de gás há partículas lentas e rápidas (respectivamente com velocidades menores ou maiores que a "média" estatística.). Um demônio vai controlar a portinhola deixando passar de A para B somente partículas lentas e deixando passar de B para A somente partículas rápidas. Com isso a temperatura de A vai aumentar e a de B vai diminuir, o que é um fato que vai contra a intuição e contradiz a segunda lei da termodinâmica.

Szilard em 1929 propôs a seguinte explicação para o paradoxo do demônio de Maxwell. O demônio deve estar muito bem informado sobre a velocidade de todas as partículas do gás. Szilard identificou informação com entropia e mostrou que a informação disponível aumenta a entropia do sistema câmara+demônio. A idéia de seu argumento é que, pela mecânica quântica, a energia é quantizada e identificando um pacote de energia com um bit de informação sobre uma

partícula, ele calculou a entropia total do sistema câmara+demônio e mostrou que ela aumenta. Esse argumento mostra que qualquer sistema que manipula informações (um cérebro ou um computador) pode ser considerado como uma máquina térmica e aumenta a entropia total do sistema. Segundo [10], um computador (na década de 80) dissipava energia na ordem de 10^{10} vezes desse limite físico fundamental.

4.2 Informação em um evento

É conveniente explicitar o que entendemos por informação com o exemplo das frases abaixo.

1. Amanhã vai fazer sol.
2. Amanhã vamor ter um maremoto.
3. ASDFR SDFFG ER GHJHGS

A frase 1 contém pouca informação pois esse é um evento comum. Já a frase 2 contém muita informação pois é extremamente improvável que ela ocorra (pelo menos no Rio de Janeiro). A frase 3, ao contráro das frases 1 e 2, não tem um significado. A frase 3, no entanto, contém muita informação pois na língua portuguesa é muito improvável que ocorram tais combinações de letras.

Para esclarecer o argumento acima vamos supor que observamos um sistema com um número finito de eventos possíveis $S_1, S_2, ..., S_n$. Cada evento S_i tem possibilidade p_i de ocorrer, com

$$p_1 + p_2 + ... + p_n = 1$$

Nesse contexto temos que intuitivamente o evento da frase 3 é muito improvável e que deve conter mais informação que o das frases 1 ou 2, que são mais prováveis de ocorrer no espaço de eventos (frases num texto, por exemplo). Quanto maior a probabilidade de um evento, menor a sua quantidade de informação.

Pode ser provado que sob certas condições desejáveis[1], a única função possível para medir a quantidade de informação de um evento S_i é (a menos de constantes):

$$I(S_i) = -\log p_i$$

A informação de S_i pode ser interpretada como o espaço (em bits ou em símbolos) necessário para representar a ocorrência S_i. Por exemplo, seja uma variável x no intervalo $[0, 1]$ e fazemos uma medida e obtemos que $x = 0.23$. São necessários 2 dígitos decimais para representar uma medida com precisão de um intervalo de comprimento 0.01. Nesse caso temos um ganho de informação com essa medida de 2 dígitos decimais $= -\log_{10} 0.01$. Para uma medida de x com 5 casas decimais é necessário um intervalo de comprimento igual a 10^{-5}, o que está de acordo com a fórmula da informação de um evento: $I = 5 = -\log 10^{-5}$.

Em um sistema binário a informação é expressa em número de bits e a base do logaritmo deve ser 2. Para informação em dígitos decimais, a base do logaritmo deve ser 10.

[1] Essas condições são equivalentes às exigidas por Shannon na seção 4.3.

Num sistema com um número finito de eventos possíveis $S_1, S_2, ..., S_n$, onde cada evento S_i tem possibilidade p_i de ocorrer, a esperança da quantidade de informação é:

$$H = \sum_{i=1}^{n} p_i I(S_i) = -\sum_{i=1}^{n} p_i \log p_i$$

Na próxima seção vamos chegar a essa mesma fórmula com a abordagem de Shannon, que é em essência a mesma abordagem desta seção.

4.3 Teoria da Informação

A teoria da informação teve início com Shannon[7], em 1948, que a apresentou num contexto de engenharia de comunicações. A teoria da informação nasceu da necessidade de quantificar a informação contida em uma mensagem. Abaixo temos um pequeno excerto de seu famoso paper [7], onde são descritas as propriedades desejáveis para a entropia, ou informação, ou medida da quantidade de escolha na seleção de um evento.

"Suppose we have a set of possible events whose probabilities of occurrence are $p_1, p_2, \ldots, p_n$ ($p_1 + p_2 + \ldots + p_n = 1$). These probabilities are known but that is all we know concerning which event will occur. Can we find a measure of how much choice is involved in the selection of the event or of how uncertain we are of the outcome?

If there is such a measure, say $H(p_1, p_2, \ldots, p_n)$, it is reasonable to require of it the following properties:

1. H should be continuous in the p_i.

2. If all the p_i are equal, $p_i = \frac{1}{n}$, then H should be a monotonic increasing function of n. With equally likely events there is more choice, or uncertainty, when there are more possible events.

3. If a choice be broken down into two successive choices, the original H should be the weighted sum of the individual values of H. The meaning of this is illustrated in Fig. 4.1. At the left we have three possibilities $p_1 = \frac{1}{2}, p_2 = \frac{1}{3}, p_3 = \frac{1}{6}$. On the right we first choose between two possibilities each with probability $\frac{1}{2}$, and if the second occurs make another choice with probabilities $\frac{2}{3}, \frac{1}{3}$. The final results have the same probabilities as before. We require, in this special case, that

$$H(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}) = H(\frac{1}{2}, \frac{1}{2}) + \frac{1}{2} H(\frac{2}{3}, \frac{1}{3})$$

The coefficient $\frac{1}{2}$ is because this second choice only occurs half the time."

A propriedade 3 indica que se por exemplo, os números de 00 a 99 são equiprováveis, a informação que o número de saída é o 57 é igual a informação que o algarismo das dezenas do múmero é 5 somado à informação que o algarismo das unidades é 7.

Shannon mostrou que a única função H que satisfaz essas 3 propriedades é da forma:

$$H = -k \sum_{i} p_i \log p_i$$

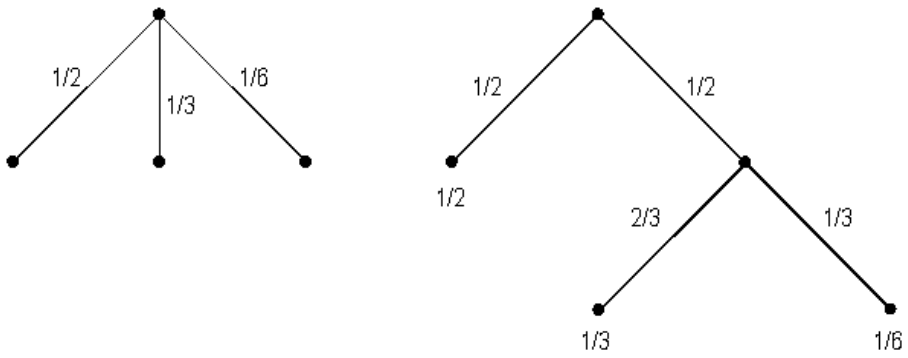

Figura 4.1: Decomposição de uma escolha oriunda de três possibilidades.

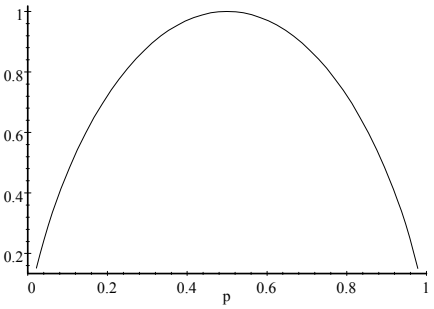

Figura 4.2: Gráfico da entropia para $(p, 1-p)$.

onde k é uma constante positiva, que sem perda de generalidade tomamos $k = 1$.

Seja $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ uma distribuição de probabilidades, com $p_i \geqslant 0$ e $\sum_{i=1}^{n} p_i = 1$. Denotamos a entropia de $\mathbf{p}$ por:

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log p_i$$

Como $p_i \log p_i$ não é definido para $p_i = 0$ tomamos $0 \log 0 = \lim_{p \to 0} p \log p = 0$.

Na figura 4.2 temos o gráfico da entropia em bits (log na base 2) para as duas variáveis $(p, 1-p)$, isto é, $H = -p \log_2 p - (1-p) \log_2 (1-p)$.

4.4 Propriedades da Entropia

A entropia possui várias propriedades interessantes.

- Para as n distribuições degeneradas $\Delta_1 = (1, 0, ..., 0), \Delta_2 = (0, 1, ..., 0), ..., \Delta_n = (0, 0, ..., 1)$, temos que $H(\Delta_i) = 0$, o que é desejável, pois nesse caso não há nenhuma incerteza na saída.

- Para quaisquer distribuições de probabilidades $\mathbf{p}$ não degeneradas temos $H(\mathbf{p}) > 0$, o que é desejável, pois há incerteza (ou informação) na saída.
- $H(\mathbf{p})$ não altera se trocarmos $p_1, p_2, \ldots, p_n$ de posição entre eles. Esta propriedade é desejável pois a ordenação dos $p_i's$ da distribuição de probabilidades não pode alterar a entropia.
- A entropia não se altera com a inclusão de um evento impossível, isto é:

$$H(p_1, p_2, \ldots, p_n, 0) = H(p_1, p_2, \ldots, p_n)$$

- $H(\mathbf{p})$ é uma função côncava com máximo para $p_1 = p_2 = \ldots = p_n = \frac{1}{n}$. Nesse caso, o valor da entropia é:

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log p_i = -\sum_{i=1}^{n} \frac{1}{n} \log \frac{1}{n} = -n\left(\frac{1}{n} \log \frac{1}{n}\right) = \log n$$

- A fórmula da entropia termodinâmica $S = k \log W$ pode ser deduzida da fórmula da entropia de Shannon, pois na termodinâmica todos os microestados são equiprováveis e o número de microestados acessíveis é W. Nesse caso a entropia de Shannon é:

$$H(\mathbf{p}) = -\sum_{i=1}^{W} \frac{1}{W} \log \frac{1}{W} = -W\left(\frac{1}{W} \log \frac{1}{W}\right) = \log W = S$$

4.5 Codificação de Símbolos

Uma aplicação interessante da teoria da informação é o problema de codificação ótima de um sinal. Seja uma fonte de comunicação que só emite os símbolos $A, B, ..., Z$ e conhecemos as suas probabilidades de emissão pela fonte: $p_A, p_B, \ldots, p_Z$. Qual o número mínimo de bits que precisam ser usados em média para codificar o conjunto de símbolos de saída da fonte de informação?

Para que tenhamos um número mínimo de bits é necessário usar um código de tamanho variável para cada símbolo, com menos bits para o código de símbolos mais frequentes. Não desejamos "vírgulas" (que é um novo símbolo) entre palavras.

Vamos ilustrar o problema de codificação de símbolos com uma fonte que só emite os símbolos A, B, C, D com as respectivas probabilidades $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}$. Na tabela 1 temos duas codificações possíveis.

Símbolo	Probabilidade	Codificação α	Codificação β
A	1/2	1	00
B	1/4	01	01
C	1/8	001	10
D	1/8	000	11

Tabela 1 - Codificação α e β dos símbolos A, B, C, D.

Uma condição suficiente para que a codificação seja unicamente decifrável é que nenhum código seja prefixo de outro código (é o caso dos códigos da tabela 1). Uma codificação que

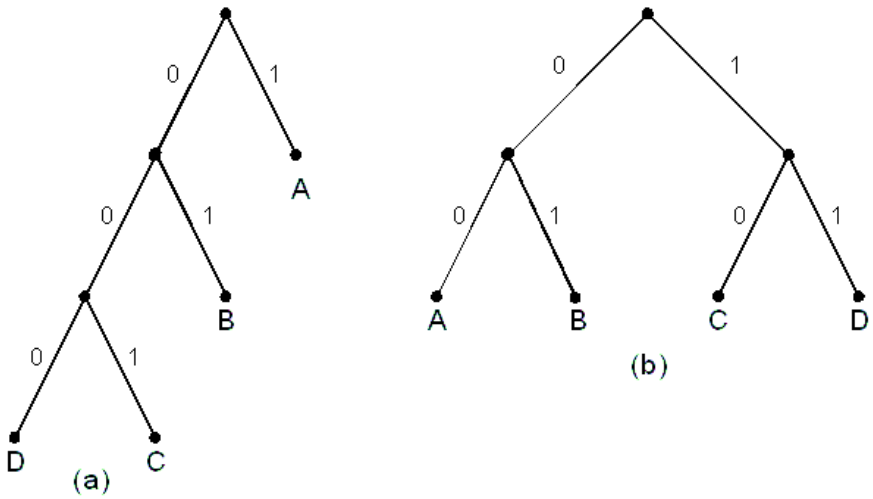

Figura 4.3: Árvores das codificações (a) α (b) β.

satisfaz essa condição pode ser diagramado na forma de uma árvore. Na figura 4.3 temos as árvores das codificações α e β.

Seja $l(x)$ o tamanho em bits do código do símbolo x e $p(x)$ a sua probabilidade. O comprimento médio da codificação pode ser calculado com a fórmula:

$$L = \sum_x l(x)p(x)$$

Shannon demonstrou em seu teorema fundamental para canais sem ruído[7] que não existe uma codificação tal que seu comprimento médio L seja menor que a entropia $H(X)$ e que dado $\varepsilon > 0$, existe uma codificação tal que $L < H(X) + \varepsilon$.

O comprimento médio das codificações α e β da tabela acima e a entropia (em bits) são respectivamente:

$$L_\alpha = \frac{1}{2}.1 + \frac{1}{4}.2 + \frac{1}{8}.3 + \frac{1}{8}.3 = \frac{7}{4}$$
$$L_\beta = \frac{1}{2}.2 + \frac{1}{4}.2 + \frac{1}{8}.2 + \frac{1}{8}.2 = 2$$
$$H = -\frac{1}{2}.\log\frac{1}{2} - \frac{1}{4}.\log\frac{1}{4} - 2.\frac{1}{8}\log\frac{1}{8} = \frac{1}{2} + \frac{1}{2} + \frac{3}{4} = \frac{7}{4}$$

Note que a codificação α é ótima pois $L_\alpha = H$.

4.6 Árvores de Decisão

Vamos apresentar o conceito de árvores de decisão com um exemplo. Imagine um jogo com dois participantes P e R. O jogador R escolhe em segredo um objeto O e o jogador P tem que descobrir qual é esse objeto fazendo perguntas ao R, que responderá dizendo SIM ou NÃO.

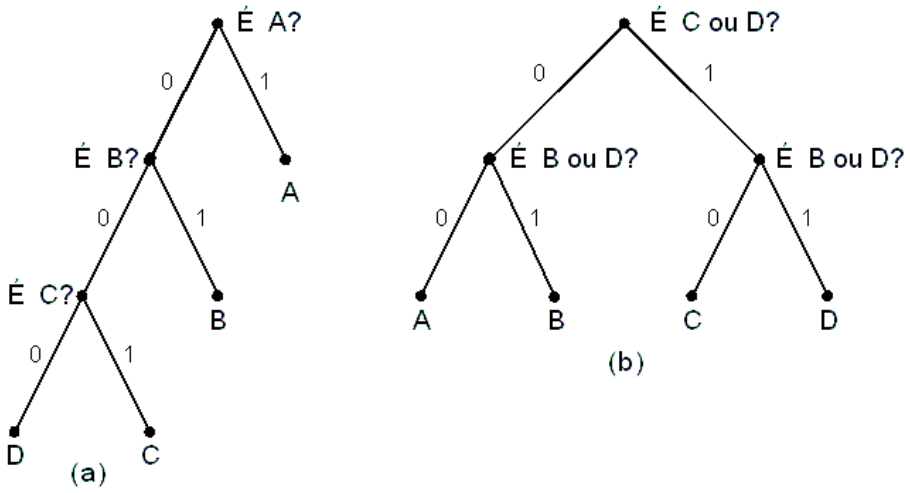

Figura 4.4: Árvores dos esquemas de perguntas (a) e (b)

Simplificando o problema supomos que só há 4 objetos possíveis para R escolher: A, B, C, D e que conhecemos as probabilidades com que R escolhe os objetos:

$$p_A = \frac{1}{2}, p_B = \frac{1}{4}, p_C = \frac{1}{8}, p_D = \frac{1}{8}$$

Na figura 4.4 temos dois esquemas de perguntas. A árvore da figura 4.4(a) indica que inicialmente fazemos a pergunta: "O objeto escolhido é o A?". Se a resposta for SIM (representado por "1" na figura) sabemos com certeza que o objeto é o A. Se a resposta for NÃO ("0"), fazemos a pergunta "O objeto é o B?" e assim por diante.

Note que esse problema de árvores de decisão é equivalente ao problema de codificação de símbolos, como indica a semelhança das figuras 4.3 e 4.4.

A Codificação α da tabela 1 na seção 4.5 é a codificação das perguntas na árvore da figura 4.4(a). Por exemplo, o código "1" do símbolo A na tabela indica a certeza do símbolo A se temos resposta positva à primeira pergunta. O código "01" do símbolo B na tabela indica a certeza do símbolo B se a primeira resposta é negativa e a segunda resposta é positiva. A Codificação β da tabela 1 é a codificação das perguntas da árvore da figura 4.4(b).

Os resultados da seção 4.5 se aplicam aqui e portanto o número médio de perguntas não pode ser menor que a entropia. O esquema de perguntas da figura 4.4(a) é ótimo pois temos em média $\frac{7}{4}$ perguntas, que é o valor da entropia. No esquema de perguntas da figura 4.4(b) temos 2 perguntas para saber qual o objeto escolhido.

4.7 Entropia de Variáveis Aleatórias

Uma variável aleatória X é definida pelo alfabeto de possíveis valores que ela pode assumir: $\{x_1, x_2, \ldots, x_n\}$ e pela sua distribuição de probabilidades $\mathbf{p} = (p_1, p_2, \ldots, p_n)$, onde $p_i = P\{X = x_i\}$

A esperança de uma variável aleatória X, governada pela distribuição $\mathbf{p}$, é:

$$E(X) = \sum_{i=1}^{n} p_i x_i$$

A entropia de X é:

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i$$

Podemos considerar uma nova variável aleatória $Z(X) = -\log P\{X\}$, logo:

$$E(Z) = E(-\log P\{X\}) = -\sum_{i=1}^{n} p_i . \log p_i = H(X)$$

Desta fórmula temos que a entropia é uma média e o valor $-\log p_x$ pode ser visto como a quantidade de informação suprida pela saída x.

Sejam X e Y duas variáveis aleatórias. A entropia da variável aleatória de duas dimensões $Z = (X, Y)$ é:

$$H(X,Y) = -\sum_{x,y} P\{X = x, Y = y\} \log P\{X = x, Y = y\}$$

Temos que:

$$H(X,Y) \leq H(X) + H(Y)$$

onde a igualdade só vale se X e Y são independentes.

Para estimar a entropia de uma fonte de comunicação baseado nas suas saídas devemos considerar cada saída no tempo i como a realização de uma variável aleatória X_i. Quando a fonte é constante (as variáveis X_i tem a mesma distribuição) e sem memória (os valores passados não alteram o presente) temos que:

$$P\{X_1 = x_1, X_2 = x_2, ..., X_n = x_n\} = \prod_{i=1}^{n} P\{X_i = x_i\}$$

Então:

$$H(X_1, X_2, ..., X_n) = nH(X)$$

Calculamos a entropia por unidade de saída com a fórmula:

$$H = \frac{1}{k} \lim_{k \to \infty} H(X_1, X_2, ..., X_k)$$

4.8 Entropia Condicional

A entropia condicional é uma média ponderada de entropias.

$$H(X|Y) = -\sum_{y} P\{Y = y\} H\{X|Y = y\}$$

A entropia condicional pode ser calculada com:

$$H(X|Y) = -\sum_{x,y} P\{X = x, Y = y\} \log P\{X = x|Y = y\}$$

Duas propriedades importantes da entropia condicional são:

$$H(X,Y) = H(X) + H(Y|X)$$

$$H(X_1|X_2, ..., X_k, X_{k+1}) \leqslant H(X_1|X_2, ..., X_k)$$

A primeira equação pode ser interpretada da seguinte forma: A incerteza sobre duas variáveis aleatórias é igual a incerteza da primeira variável mais a incerteza média da segunda variável com a primeira variável conhecida. Já na segunda equação temos que a incerteza decresce quanto mais informação é conhecida.

4.9 Informação Mútua

A informação mútua entre duas variáveis aleatórias X e Y é definida como:

$$I(X;Y) = H(X) - H(X|Y)$$

A informação mútua é simétrica pois:

$$I(X;Y) = H(X) - H(X|Y) = \sum_{x,y} P(x,y) \log \frac{P(x|y)}{P(x)} =$$

$$= \sum_{x,y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)} = H(X) + H(Y) - H(X,Y) = I(Y;X)$$

Do cálculo acima temos também que:

$$0 \leqslant I(X;Y) \leqslant \min\{H(X), H(Y)\}$$

Se X não depende de Y $(H(X|Y) = H(X))$ temos que $I(X;Y) = 0$. Se X determina Y ou Y determina X temos que $I(X;Y) = \min\{H(X), H(Y)\}$.

Finalmente, notamos que

$$I(X;Y) = \sum_{x,y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)} = E\left[\log \frac{P(x,y)}{P(x)P(y)}\right]$$

Podemos interpretar $\log \frac{P(x,y)}{P(x)P(y)}$ como a informação mútua entre os simbolos x e y.

4.10 Divergência

Muitas vezes é necessário medir a distância entre duas distribuições de probabilidade $\mathbf{p}$ e $\mathbf{q}$. A medida mais comum é a desenvolvida por Kullback e Leibler em 1951 que é chamada de distância (ou divergência ou medida) de Kullback-Leibler, entropia relativa, discriminante ou simplesmente divergência. A divergência é definida por:

$$D(\mathbf{p};\mathbf{q}) = \sum_{i=1}^{n} p_i . \log \frac{p_i}{q_i}$$

onde assumimos que se $q_i = 0$ então $p_i = 0$ e que $0 \log \frac{0}{0} = 0$.

A divergência não é simétrica, isto é, $D(\mathbf{p};\mathbf{q}) \neq D(\mathbf{q};\mathbf{p})$. Em aplicações em que a simetria dessa medida é importante, podemos usar a divergência simétrica definida por:

$$J(\mathbf{p};\mathbf{q}) = D(\mathbf{p};\mathbf{q}) + D(\mathbf{q};\mathbf{p})$$

Como a entropia, a divergência tem várias propriedades interessantes, como:

1. $D(\mathbf{p};\mathbf{q}) = 0$ só se $\mathbf{p} = \mathbf{q}$. Se $\mathbf{p} \neq \mathbf{q}$ então $D(\mathbf{p};\mathbf{q}) > 0$.
2. $D(\mathbf{p};\mathbf{q})$ é contínua e convexa em $\mathbf{p}$ e $\mathbf{q}$.
3. $D(\mathbf{p};\mathbf{q})$ é permutacionalmente simétrica, isto é, $D(\mathbf{p};\mathbf{q})$ não se altera se os pares $(p_1, q_1), \ldots, (p_n, q_n)$ são permutados.
4. Se $\mathbf{U} = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$ é a distribuição uniforme então

$$D(\mathbf{p};\mathbf{U}) = \sum_{i=1}^{n} p_i . \log \frac{p_i}{1/n} = \log n + \sum_{i=1}^{n} p_i . \log p_i = \log n - H(\mathbf{p})$$

5. $I(X;Y) = \sum_{x,y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)} = D(\mathbf{p};\mathbf{q})$, onde $\mathbf{p} = P(x,y)$ e $\mathbf{q} = P(x)P(y)$.

4.11 Variáveis Aleatórias Contínuas

Nesta seção daremos as definições de entropia, entropia condicional, divergência e informação mútua para variáveis aleatórias contínuas. Elas são parecidas, basicamente trocando o símbolo $\sum$ por $\int$.

Seja uma variável aleatória contínua X com distribuição de probabilidade $f_X(x)$. A entropia de X, chamada entropia diferencial é definida por

$$H(X) = -\int f_X(x) \log f_X(x) dx = -E\left[\log f_X(x)\right]$$

A entropia diferencial tem várias propriedades da entropia ordinária (para variáveis discretas), no entanto ela não é uma medida absoluta da aleatoriedade de X, como no caso discreto (intuitivamente uma variável contínua tem informação infinita).

A informação mútua para variáveis contínuas tem a mesma definição que para variáveis discretas:

$$I(X;Y) = H(X) - H(X|Y)$$

onde a entropia condicional é definida por:

$$H(X|Y) = -\int\int f_{X,Y}(x,y) \log f_X(x|y) dxdy$$

Como no caso discreto, também vale a relação:

$$I(X;Y) = \int\int f_X(x,y) \log \frac{f_X(x|y)}{f_X(x)} dxdy = \int\int f_{X,Y}(x,y) \log \frac{f_{X,Y}(x,y)}{f_X(x) f_Y(y)} dxdy$$

A divergência é definida por:

$$D\left(f_X; g_X\right) = \int f_X(x) \log \frac{f_X(x)}{g_X(x)} dx$$

A divergência para variáveis contínuas, como para variáveis discretas, tem algumas propriedades únicas:

- $D\left(f_X; g_X\right) = 0$ só se $f_X = g_X$. Se $f_X \neq g_X$ então $D\left(f_X; g_X\right) > 0$.
- A divergência é invariante em relação as seguintes variações nas componentes do vetor $\mathbf{x}$: Permutação da ordem na qual as componentes estão arranjadas, escalamento de amplitude e transformação não linear monótona.

E também vale a relação:

$$I(X;Y) = \int\int f_{X,Y}(x,y) \log \frac{f_{X,Y}(x,y)}{f_X(x) f_Y(y)} dxdy = D\left(f_{X,Y}; f_X f_Y\right)$$

4.12 Princípio da Máxima Entropia - MaxEnt

Na natureza, a entropia de um sistema sempre aumenta (ou melhor, nunca diminui) pela segunda lei da Termodinâmica. No entanto, em geral, a entropia de um sistema físico tende para a distribuição de máxima entropia, respeitando as restrições físicas que o sistema impõe. O Princípio da Máxima Entropia é baseado nessa idéia, isto é, procurar a distribuição de máxima entropia, respeitando as restrições impostas.

Seja X uma variável aleatória que pode assumir os valores $x_1, x_2, \ldots, x_n$, mas com as correspondentes probabilidades $p_1, p_2, \ldots, p_n$ desconhecidas. No entanto conhecemos algumas restrições sobre a distribuição de probabilidades como por exemplo a média, variância ou algum momento de X. Como escolher as probabilidades $p_1, p_2, \ldots, p_n$?

Esse problema em geral tem infinitas soluções. Devemos procurar uma função a ser otimizada para que consigamos uma única solução. O princípio da máxima entropia, proposto formalmente por Jaynes em 1957, oferece uma solução para esse problema.

O princípio da máxima entropia, que chamaremos de MaxEnt, diz que das distribuições de probabilidades que satisfazem as restrições, devemos escolher aquela com a máxima entropia.

O MaxEnt é um princípio muito geral e que tem aplicações em diversos campos de conhecimento. Suas primeiras aplicações foram em mecânica estatística e termodinâmica. Existem aplicações do MaxEnt em diversas áreas como planejamento urbano, economia, teoria de filas, análise espectral, reconhecimento de padrões, processamento de imagens, modelos de crescimento de populações, modelos de linguagem e muitas outras áreas. Uma importante aplicação em termodinâmica estatística é descobrir a distribuição de probabilidades que descrevem os (micro)estados de um sistema baseado em medidas (valores médios de funções, que são as restrições do problema) feitas em escala macroscópica.

Uma explicação intuitiva para o MaxEnt é que se escolhermos uma distribuição com entropia menor que a máxima, esta redução de entropia vem do uso de informações não fornecidas, mas usadas conscientemente ou inconscientemente. Por exemplo, se não temos nenhuma restrição, a distribuição escolhida pelo MaxEnt é a uniforme. Se por alguma razão não escolhemos a distribuição uniforme, isto é, se para algum i e j temos $p_i > p_j$, então foi usada alguma informação não fornecida para causar essa assimetria na distribuição de probabilidades.

MaxEnt pode ser formulado matematicamente como: Seja X uma variável aleatória com o alfabeto $\{x_1, x_2, \ldots, x_n\}$ e com a distribuição de probabilidades $\mathbf{p} = (p_1, p_2, \ldots, p_n)$. Queremos maximizar a medida de Shannon sujeita a várias restrições lineares.

$$\begin{cases} \arg\max H(X) = -\sum_{i=1}^{n} p_i \log p_i \\ \text{sujeito a } p_i \geqslant 0, \forall i \;\; \text{e} \;\; \sum_{i=1}^{n} p_i = 1 \;\; \text{e} \;\; \sum_{i=1}^{n} p_i g_{ri} = a_r, r = 1, 2, ..., m. \end{cases}$$

Como o domínio é convexo e $H(X)$ é côncava, temos um problema de otimização convexa, o que garante a existência de uma única solução e facilidade e eficiência computacional para o problema.

O problema acima tem uma estrutura particular que garante que na solução temos $p_i \geqslant 0, \forall i$ (pois os p_i são exponenciais, como veremos a seguir). Dessa forma não precisamos nos preocupar com as restrições de desigualdade.

Aplicando o método de KKT (ou dos multiplicadores de Lagrange) temos que a solução é:

$$p_i = \exp(-\lambda_0 - \lambda_1 g_{1i} - \lambda_2 g_{2i} - \lambda_m g_{mi}), \qquad i = 1, 2, ..., n.$$

onde $\lambda_0, \lambda_1, \lambda_2, ..., \lambda_m$ são os multiplicadores de Lagrange que podem ser calculados com as equações:

$$\begin{cases} \sum_{i=1}^{n} \exp(-\lambda_0 - \sum_{j=1}^{m} \lambda_j g_{ji}) = 1 \\ \sum_{i=1}^{n} g_{ri} \exp(-\lambda_0 - \sum_{j=1}^{m} \lambda_j g_{ji}) = a_r, \qquad r = 1, 2, ..., m. \end{cases}$$

A formulação do MaxEnt para variáveis aleatórias contínuas é semelhante ao caso discreto. Seja uma variável aleatória contínua e sua distribuição de probabilidades contínua $f(x)$. Se conhecemos somente os valores esperados $E[g_1(x)] = a_1, E[g_2(x)] = a_2, ..., E[g_m(x)] = a_m$ de x,

temos, em geral, infinitas distribuições satisfazendo esses momentos. De acordo com MaxEnt devemos escolher a distribuição de máxima entropia, que é:

$$f(x) = \exp(-\lambda_0 - \lambda_1 g_1(x) - \lambda_2 g_2(x) - \lambda_m g_m(x))$$

onde $\lambda_0, \lambda_1, \lambda_2, ..., \lambda_m$ são os multiplicadores de Lagrange, que podem ser calculados com as equações:

$$\begin{cases} \int f(x)dx = 1 \\ \int f(x)g_r(x)dx = a_r, \qquad r = 1, 2, ..., m. \end{cases}$$

Para maiores informações, consulte [8].

4.13 Distribuições obtidas com MaxEnt

A seguir daremos o domínio da variável aleatória e quais os momentos conhecidos. Com a aplicação do MaxEnt e usando as fórmulas da seção anterior, seja no caso discreto ou contínuo, obtemos a distribuição de probabilidade de máxima entropia (DPME).

Boa parte das distribuições da estatística podem ser caracterizadas a partir de certos momentos prescritos e a aplicação do MaxEnt (ou do MinxEnt, comentado na seção 4.14).

1. Se o domínio de x é $[a, b]$ e não há nenhuma restrição (exceto as restrições naturais da distribuição de probabilidades), a DPME é a distribuição uniforme. A distribuição uniforme é caracterizada pela ausência de restrições para a entropia.

2. Se o domínio de x é $[a, b]$ e temos a média aritmética $m = E[x]$, a DPME é a distribuição exponencial truncada dada por:

   $$f(x) = ce^{-kx}$$

   onde c e k podem ser calculados com as fórmulas:

   $$c\int_a^b e^{-kx}dx = 1 \qquad \text{e} \qquad c\int_a^b xe^{-kx}dx = m$$

3. Se o domínio de x é $[0, \infty)$ e temos a média $m = E[x]$ a DPME é a distribuição dada por:

   $$f(x) = \frac{1}{m}e^{-x/m}$$

   e a máxima entropia é:

   $$S_{MAX} = 1 + \ln m$$

4. Se o domínio de x é $[0, \infty)$ e temos a média aritmética $m = E[x]$ e a média geométrica $E[\ln x]$, a DPME é a distribuição gama dada por:

   $$f(x) = \frac{a^\gamma}{\Gamma(\gamma)}e^{-ax}x^{\gamma-1}$$

5. Se o domínio de x é $(-\infty, \infty)$ e temos a média $m = E[x]$ e a variância $E[(x-m)^2] = \sigma^2$, a DPME é a distribuição normal.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2\right)$$

cuja entropia é:

$$S_{MAX} = \ln \sigma\sqrt{2\pi e}$$

6. Se o domínio de x é $(-\infty, \infty)$ e temos o momento $E[x]$, a DPME é a distribuição de Laplace:

$$f(x) = \frac{1}{\sigma} \exp(-\frac{|x|}{\sigma})$$

7. Uma importante aplicação em termodinâmica estatística é descobrir a distribuição de probabilidades que descrevem os (micro)estados de um sistema baseado em medidas (valores médios de funções, que são as restrições do problema) feitas em escala macroscópica.

   Sejam $p_1, p_2, \ldots, p_n$ as probabilidades da partícula no sistema ter energias $\epsilon_1, \epsilon_2, ..., \epsilon_n$, respectivamente. A única informação que temos sobre o sistema é sua energia média $\widehat{\epsilon}$, isto é:

$$p_1\epsilon_1 + p_2\epsilon_2 + \ldots p_n\epsilon_n = \widehat{\epsilon}$$

   e as restrições naturais da probabilidade:

$$p_i \geqslant 0, \forall i \quad \text{e} \quad \sum_{i=1}^{n} p_i = 1$$

   De acordo com o MaxEnt obtemos a distribuição de Maxwell-Boltzmann da mecânica estatística:

$$p_i = e^{-\mu\epsilon_i}/(\sum_{i=1}^{n} e^{-\mu\epsilon_i}), \qquad i = 1, 2, ..., n.$$

   onde $\mu = \frac{1}{kT}$ (T é a temperatura e k é a constante de Boltzmann). Outras distribuições da mecânica estatística, como a de Bose-Einstein ou a de Fermi-Dirac também são obtidas com o MaxEnt, mudando apenas as restrições (macroscópicas) que temos sobre o sistema.

4.14 Princípios de Otimização de Entropia

O MaxEnt é o mais importante princípio de otimização de entropia. Existem vários outros princípios de otimização de entropia, dependendo do problema a ser atacado. Em [8] temos uma descrição abrangente desses princípios. Vamos apresentar brevemente o segundo mais importante

princípio de otimização de entropia: o princípio de mínima entropia cruzada, que chamaremos de MinxEnt.

O MinxEnt é utilizado quando desconhecemos a distribuição de probabilidades $\mathbf{p}$ mas temos (como no caso do MaxEnt) restrições relativas a $\mathbf{p}$ e também temos uma distribuição de probabilidades a priori $\mathbf{q}$ e queremos escolher $\mathbf{p}$ satisfazendo as restrições e mais próxima possível de $\mathbf{q}$.

O MinxEnt pode ser enunciado concisamente como: De todas as distribuições satisfazendo a dadas restrições, escolher aquela mais próxima da distribuição a priori. A medida de distância usual para distribuições de probabilidades é a divergência. Com essa medida, o MinxEnt diz que devemos minimizar a divergência $D(\mathbf{p};\mathbf{q})$ sujeito as restrições do problema.

Como a divergência $D(\mathbf{p};\mathbf{q})$ é convexa e o domínio (em geral) é convexo, temos (como no caso do MaxEnt) um problema de otimização convexa, o que garante a existência de uma única solução e a eficiência computacional para o solução do problema.

Um fato interessante é que se a distribuição a priori $\mathbf{q}$ não é dada, o mais matural é escolhermos a distribuição de probabilidades $\mathbf{p}$ mais próxima possível da distribuição uniforme $\mathbf{U} = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$. Como já vimos na seção de Divergência, temos que:

$$D(\mathbf{p};\mathbf{U}) = \sum_{i=1}^{n} p_i . \log \frac{p_i}{1/n} = \log n + \sum_{i=1}^{n} p_i . \log p_i = \log n - H(\mathbf{p})$$

Daí temos que minimizar a divergência em relação à distribuição uniforme equivale a maximizar a entropia. Logo MaxEnt é um caso particular de MinxEnt em que minimizamos a distância em relação a distribuição uniforme. Podemos unir MaxEnt e MinxEnt em um princípio geral:

"De todas as distribuições de probabilidades satisfazendo a dadas restrições, escolher a mais próxima de uma dada distribuição a priori. Se essa distribuição a priori não é fornecida, escolher a distribuição mais próxima da distribuição uniforme."

Capítulo 5

Entropia e Modelagem de Linguagem

Neste capítulo vamos apresentar algumas aplicações do conceito de entropia em modelagem de linguagem. A principal referência desse capítulo é [1], onde há demonstrações de fatos enunciados aqui, explicações adicionais e outras aplicações de entropia em modelagem de linguagem.

5.1 Modelos de Linguagem

Estamos interessados aqui em estimar qual a probabilidade de ocorrência de uma frase numa lingua. Essa estimativa deve ser baseada em um texto, provavelmente longo, que seja representativo das frases que tentamos analisar. Para o cálculo da probabilidade de uma frase é necessário criar um modelo de linguagem. Podemos construir um modelo de linguagem assumindo que a probabilidade de ocorrência de uma palavra só dependa das palavras ditas anteriormente. Formalizando essa idéia, seja W uma sequência de palavras, cada uma delas pertencentes a um dicionário fixo $\mathcal{V}$.

$$W = w_1 w_2 ... w_n, \qquad w_i \in \mathcal{V}$$

Temos que a probabilidade $P(W)$ pode ser calculada da forma:

$$P(W) = \prod_{i=1}^{n} P(w_i | w_1 w_2 ... w_{i-1})$$

Dada uma palavra w_i, as palavras $w_1 w_2 ... w_{i-1}$ são chamadas de história de w_i, que denotamos por h_i. A história é uma sequência de palavras, onde cada uma delas pertence a um dicionário fixo $\mathcal{V}$. A quantidade de histórias possíveis é um número, em geral, extremamente grande.

Para que possamos calcular $P(W)$ é necessário algum tipo de simplificação. Uma simplificação muito comum é a classificação de equivalência da história. Seja Φ um mapeamento de histórias em M classes de equivalência. $\Phi\left(w_1 w_2 ... w_{i-1}\right)$ denota a classe de equivalência da história $w_1 w_2 ... w_{i-1}$. Daí temos que $P(w_i|h_i)$ pode ser calculado com:

$$P(w_i|h_i) = P(w_i|w_1 w_2 ... w_{i-1}) = P\left(w_i|\Phi\left(w_1 w_2 ... w_{i-1}\right)\right)$$

Logo podemos aproximar o cálculo de $P(W)$ com a fórmula:

$$P(W) = \prod_{i=1}^{n} P(w_i|\Phi\left(w_1 w_2 ... w_{i-1}\right))$$

No modelo de linguagem n-grama, a palavra w_i só depende das $n-1$ últimas palavras. O modelo de linguagem n-grama é uma aproximação de Markov de ordem n, em que o estado atual só depende dos $n-1$ anteriores. Já apresentamos um modelo de linguagem comum em reconhecimento de voz, que é o modelo de trigramas, onde:

$$P(w_i|w_1w_2...w_{i-1}) = P\left(w_i|\Phi\left(w_1w_2...w_{i-1}\right)\right) = P(w_i|w_{i-2}w_{i-1})$$

Mesmo no modelo de linguagem de trigramas, o número de histórias é muito grande pois se, por exemplo, o tamanho do vocabulário $|\mathcal{V}|$ é igual a 10.000, temos $|\mathcal{V}|^3 = 10^{12}$ combinações possíveis de w_{i-2}, w_{i-1} e w_i

5.2 Árvores de Decisão

Um exemplo simples de árvores de decisão foi apresentado no capítulo de entropia, onde mostramos a semelhança de árvores de decisão com codificação de sinais.

Nesta seção estamos interessados em estimar qual a probabilidade da ocorrência de uma palavra w dada a sua história h. Por exemplo, seja a frase: *"O reconhecimento de voz ....."*. Queremos descobrir a probabilidade de uma palavra w ser continuação dessa frase. Intuitivamente temos que o verbo *"é"* é uma palavra bastante provável, assim como a conjunção aditiva *"e"*. Já a palavra *"paz"* é muito pouco provável pois a frase *"O reconhecimento de voz paz"* não é comum.

Para calcular a probabilidade de uma palavra w devemos fazer perguntas à sua história h, como por exemplo: "a última palavra é voz?", ou "a penúltima palavra é calor?" ou ainda: "a ante-penúltima palavra é reconhecimento?". No caso do exemplo, a resposta à primeira e terceiras perguntas é positiva. Já a resposta à segunda pergunta é negativa.

É possível fazer perguntas sobre a função gramatical de palavras da história, como por exemplo, "a última palavra é um verbo?". No entanto nos restringiremos à perguntas do tipo: "a palavra de posição m da história é v_i?".

Formulando matematicamente essa idéia, seja $w_{-m}w_{-m+1}...w_{-1}$ a história h. Queremos estimar $P(w|w_{-m}w_{-m+1}...w_{-1}) = P(w|h)$ com perguntas do tipo: "a palavra w_{-m} da história é v?"

Na figura 5.1 temos uma árvore de decisão. Inicialmente fazemos a pegunta Q_{10}. Se a resposta é sim (representado por "1" na figura), fazemos a pergunta Q_{21}. Se a resposta é não ("0") fazemos a pergunta Q_{20}, e assim por diante.

Cada pergunta divide a história em duas partes, fazendo assim uma classificação de equivalência da história. Por exemplo, a pergunta Q_{10} particiona a história em duas classes de equivalência Φ_{20} e Φ_{21}, que são relacionadas com Y_{20} e Y_{21}, que são os subconjuntos da história cuja resposta à Q_{10} é respectivamente negativa ou positiva. Outras perguntas são feitas criando assim novas classificações de equivalência da história.

Podemos criar uma árvore de decisão com os seguintes passos:

1. De todas as possíveis perguntas, achar a pergunta Q_{10} que irá classificar todas as histórias h em duas classes de equivalência Φ_{20} e Φ_{21} tal que o modelo de linguagem resultante $P(W) = \prod_{i=1}^{n} P(w_i|\Phi(h_i))$ seja o melhor possível.

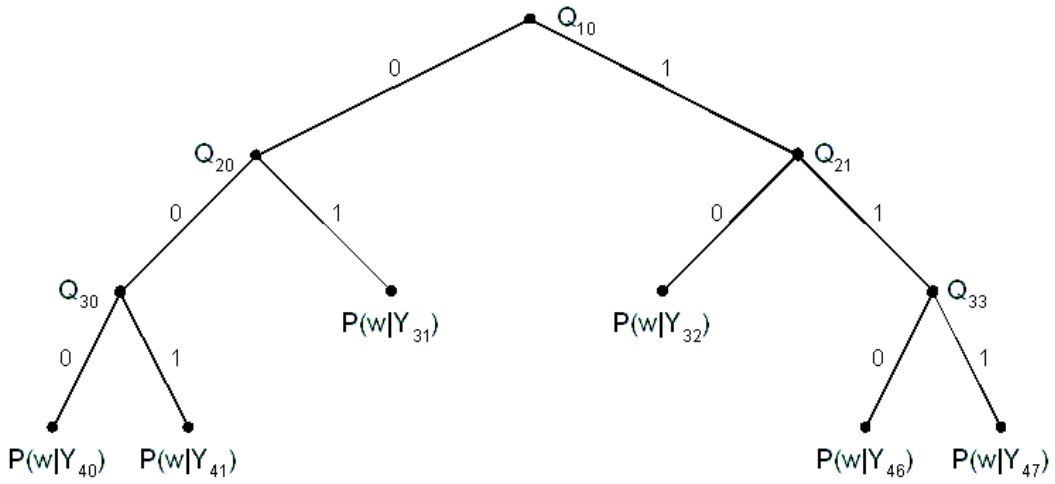

Figura 5.1: Exemplo de árvore de decisão.

2. Com Q_{10} fixado, achar a melhor tentativa de pergunta Q_{20} que particiona a classe de equivalência Φ_{20} nas classes Φ_{30} e Φ_{31} de forma que o modelo de linguagem resultante seja o melhor possível.

3. Continuando com Q_{10} fixado, achar a melhor tentativa de pergunta Q_{21} que particiona a classe de equivalência Φ_{21} nas classes Φ_{32} e Φ_{33} de forma que o modelo de linguagem resultante seja o melhor possível.

4. Das perguntas Q_{20} e Q_{21} fixar aquela cujo modelo resultante seja o melhor possível.

5. Supondo que Q_{21} tenha sido escolhido e fixado no passo 4, devemos pesquisar qual a melhor entre as perguntas Q_{20}, Q_{32} e Q_{33}. Fixar essa melhor pergunta e repetir esse procedimento com as perguntas que restarem.

Não foi explicado no algoritmo acima qual o critério de escolha da melhor pergunta nem foi proposto um critério de parada da construção da árvore. Esse é o objetivo da próxima seção.

5.3 Entropia e Árvores de Decisão

A entropia pode ser usada como um critério para escolher a melhor pergunta em uma árvore de decisão. De uma maneira geral, seja uma variável aleatória X com entropia $H(X)$. Uma pergunta Q é uma variável aleatória que só pode assumir os valores 0 ou 1 com respectivas probabilidades $P(Q = 1)$ e $P(Q = 0)$. Ao se formular uma pergunta Q, mais informação é conhecida. A entropia condicional $H(X|Q)$ pode ser interpretada como a entropia do processo estocástico quando é formulada a pergunta Q. A entropia condicional $H(X|Q)$ é uma média ponderada de entropias e pode ser calculada com:

$$H(X|Q) = P(Q = 1)H(X|Q = 1) + P(Q = 0)H(X|Q = 0)$$

A informação mútua é definida como:

$$I(X;Q) = H(X) - H(X|Q)$$

e pode ser interpretada como a quantidade de informação ganha ao se formular a pergunta Q.

A informação mútua pode ser usada como o critério de eficiência de uma pergunta pois quanto maior a informação mútua $I(X;Q)$, maior a quantidade de informação ganha ao se formular a pergunta Q. Como $H(X)$ é fixado, maximizar a informação mútua $I(X;Q)$ equivale a minimizar a entropia condicional $H(X|Q)$. Daí temos que quanto melhor a pergunta Q, menor é a entropia condicional $H(X|Q)$.

Árvores de decisão em modelagem de linguagem são criadas a partir de um texto de treinamento que é dividido em duas partes. A parte maior é o texto de desenvolvimento e é usado para a criação da árvore. Uma parte menor, que é chamado de texto de checagem da validação cruzada, é visto como um dado não observado e será usado para verificar se os resultados obtidos com o texto de desenvolvimento são capazes de modelar bem dados diferentes do próprio texto de desenvolvimento ou se é necessário continuar a criar novas perguntas, refinando a construção da árvore. Esse tipo de método é comumente chamado de método de validação cruzada.

A partir do texto de treinamento podemos calcular a probabilidade de uma palavra w condicionada a uma classe de equivalência Φ_{ij} do seguinte modo:

$$P(w|\Phi_{ij}) = f(w|Y_{ij}) = \frac{C(w|Y_{ij})}{C(Y_{ij})}$$

onde f denota a frequência relativa de w em Y_{ij}, que é um subconjunto do texto de treinamento caracterizado pela classe de equivalência Φ_{ij}. $C(w|Y_{ij})$ indica o número de vezes da ocorrência de w em Y_{ij} e $C(Y_{ij})$ indica o número de palavras em Y_{ij}.

O tamanho relativo do conjunto Y_{ij} é definido como:

$$f(Y_{ij}) = \frac{C(Y_{ij})}{|T|}$$

onde $|T|$ indica o tamanho do texto de treinamento (de desenvolvimento ou de checagem).

O critério usado para a eficácia das perguntas é minimizar a entropia empírica (calculada no texto de desenvolvimento) condicionada à árvore que é dada por:

$$H = \sum_{i,j} f(Y_{ij}) \sum_{w} f(w|Y_{ij}) \log f(w|Y_{ij})$$

onde o somatório é feito nas folhas da árvore.

Na figura 5.1, por exemplo, a entropia condicional é dada por:

$$H = \sum_{i=0,1,6,7} f(Y_{4i}) \sum_{w} f(w|Y_{4i}) \log f(w|Y_{4i}) + \sum_{i=1,2} f(Y_{3i}) \sum_{w} f(w|Y_{3i}) \log f(w|Y_{3i})$$

A função de entropia cruzada é definida por:

$$H^c = \sum_{i,j} g(Y_{ij}) \sum_{w} g(w|Y_{ij}) \log f(w|Y_{ij})$$

onde $f(w|Y_{ij})$ indica a frequência relativa calculada no texto de desenvolvimento e $g(Y_{ij})$ e $g(w|Y_{ij})$ indicam frequências relativas calculadas no texto de checagem.

H^c pode ser interpretado como a entropia condicional quando "achamos" que a distribuição de probabilidades do sistema é $f(w|Y_{ij})$ mas a distribuição real é $g(w|Y_{ij})$. Esse erro na estimativa da distribuição de probabilidades é custoso e temos que $H^c > H$.

Com o conceito de entropia e a validação cruzada, o procedimento de construção de árvores de decisão da seção anterior fica da forma:

1. De todas as possíveis perguntas, achar a pergunta Q_{10} que irá classificar todas as histórias h em duas classes de equivalência Φ_{20} e Φ_{21} tal que o valor resultante H calculado no texto de desenvolvimento é mínimo. Chamamos este valor de H_d. Com a questão Q_{10} calculamos H^c no texto de checagem e chamamos esse valor de H_0^c.

2. Com Q_{10} fixado, achar a pergunta Q_{20} que particiona a classe de equivalência Φ_{20} nas classes Φ_{30} e Φ_{31} resultando num valor mínimo de H (calculado no texto de desenvolvimento). Chamamos este valor de H_{20}.

3. Com Q_{10} fixado, achar a pergunta Q_{21} que particiona a classe de equivalência Φ_{21} nas classes Φ_{32} e Φ_{33} resultando num valor mínimo de H (calculado no texto de desenvolvimento). Chamamos este valor de H_{21}.

4. Das perguntas Q_{20} e Q_{21} rejeitar aquela com maior valor de H_{2i}. Seja Q_{2j} a questão mantida. Se $H_d - H_{2j} > \varepsilon$ então calculamos H_{2j}^c no texto de checagem. Se $H_0^c - H_{2j}^c > \delta$ então tomamos a questão Q_{2j} como permanente e atribuimos $H_d = H_{2j}$ e $H_0^c = H_{2j}^c$. Caso contrário, isto é, $H_d - H_{2j} < \varepsilon$ ou $H_0^c - H_{2j}^c < \delta$ rejeitamos Q_{2j}.

5. O padrão de construção da árvore já está formado. Assumindo que Q_{21} tenha sido escolhido e fixado no passo 4, devemos pesquisar entre as perguntas Q_{20}, Q_{32} e Q_{33} qual delas reduz o valor da entropia no texto de desenvolvimento em pelo menos ε e que também reduz o valor da entropia cruzada no texto de checagem em pelo menos δ. Fixar essa pergunta. Repetir esse procedimento com as perguntas que restarem até que nenhuma questão seja mais aceita.

6. Com a árvore já formada, recalcular as frequências relativas baseados nos conjuntos Y_{ij} das folhas da árvore em todo o texto de treinamento (texto de desenvolvimento + texto de checagem).

Podemos usar um critério mais restritivo (e mais rápido) na construção da árvore usando os passos abaixo ao invés dos passos 4 e 5.

- Para $j = 1, 2$: Se $H_d - H_{2j} < \varepsilon$ rejeitamos Q_{2j} e marcamos Q_{2j} como não podendo nunca mais ser postulada. Caso contrário, calculamos H_{2j}^c no texto de checagem e se $H_0^c - H_{2j}^c < \delta$ rejeitamos Q_{2j} e marcamos Q_{2j} como não podendo nunca mais ser postulada. Caso contrário ($H_d - H_{2j} > \varepsilon$ e $H_0^c - H_{2j}^c > \delta$), fixamos Q_{2j} como permanente e tomamos $H_d = H_{2j}$ e $H_0^c = H_{2j}^c$.

- O padrão de construção da árvore já está formado. A cada profundidade da árvore dividir as folhas que satisfazem os critérios de reduzir o valor da entropia no texto de desenvolvimento em pelo menos ε e reduzir o valor da entropia cruzada no texto de checagem em pelo menos δ. Caso contrário, marcamos essa folha como não podendo nunca mais ser dividida. Repetir esse procedimento até que não existam mais folhas que possam ser divididas.

5.4 Método de Chou

Ainda falta verificar como escolher as perguntas a serem feitas. Vamos apresentar o método de Chou, que é semelhante ao algoritmo de quantização vetorial k-means. O método de Chou, como o k-means, é um método de otimização local.

Desejamos fazer uma pergunta e assim dividimos o nó da árvore em duas classificações da história $\mathcal{A}$ e $\overline{\mathcal{A}}$. Seja β_v uma partição atômica da história, que é o conjunto de histórias em que a palavra na posição m da história h é v. β_v é definida por:

$$\beta_v = \{h = w_{-1}, w_{-2}, \cdots, w_{-m} = v\}$$

Fixado um m qualquer, seja $\mathcal{A}$ a união de algumas histórias β_v e $\overline{\mathcal{A}}$ o complemento de $\mathcal{A}$.

Vamos usar a notação

$$f(w|\beta_i) = f(w|h \in \beta_i)$$
$$f(w|\mathcal{A}) = f(w|h \in \mathcal{A})$$

Essas fórmulas indicam a probabilidade empírica de uma palavra w condicionada à história β_i ou à história $\mathcal{A}$, respectivamente.

A entropia condicional empírica da pergunta $\mathcal{A}$ é definida por:

$$H_{\mathcal{A}} = f(\mathcal{A})H(w|\mathcal{A}) + f(\overline{\mathcal{A}})H(w|\overline{\mathcal{A}})$$

onde $H(w|\mathcal{A}) = \sum_w f(w|\mathcal{A}) \log f(w|\mathcal{A})$

A idéia básica do algoritmo de Chou é a partir de um $\mathcal{A}$ inicial, escolher um $\mathcal{A}^*$ de forma que $H_{\mathcal{A}^*} < H_{\mathcal{A}}$. A construção de $\mathcal{A}^*$ é feita a partir do cálculo da distância (divergência) entre $f(w|\beta_i)$ e $f(w|\mathcal{A})$.

Se $f(w|\beta_i)$ é mais próximo de $f(w|\mathcal{A})$ do que de $f(w|\overline{\mathcal{A}})$, isto é, se

$$D(f(w|\beta_i), f(w|\mathcal{A})) \leqslant D(f(w|\beta_i), f(w|\overline{\mathcal{A}}))$$

colocamos β_i em $\mathcal{A}^*$. caso contrário, colocamos β_i em $\overline{\mathcal{A}}^*$

Em [1] temos a prova que $H_{\mathcal{A}^*} < H_{\mathcal{A}}$.

Abaixo temos o algoritmo de Chou:

1. Escolher partições atômicas β_v, com m fixado.
2. Escolher um $\mathcal{A}$ inicial (e seu complementar $\overline{\mathcal{A}}$), que é a união de alguns dos β_v.
3. Calcular $f(w|\mathcal{A}) = \left(\sum_{\beta \in \mathcal{A}} f(w, \beta)\right) / \left(\sum_{\beta \in \mathcal{A}} f(\beta)\right)$ e $f(w|\overline{\mathcal{A}}) = \left(\sum_{\beta \in \overline{\mathcal{A}}} f(w, \beta)\right) / \left(\sum_{\beta \in \overline{\mathcal{A}}} f(\beta)\right)$

4. Para todo β, se

$$D(f(w|\beta), f(w|\mathcal{A})) \leqslant D(f(w|\beta), f(w|\overline{\mathcal{A}}))$$

colocamos β no novo conjunto $\mathcal{A}^*$. caso contrário, colocamos β em seu complementar $\overline{\mathcal{A}}^*$.

5. Se $\mathcal{A}^* = \mathcal{A}$ paramos. Caso contrário, fazemos $\mathcal{A} = \mathcal{A}^*$ e $\overline{\mathcal{A}} = \overline{\mathcal{A}}^*$ e voltamos ao passo 3.

5.5 Princípio da Máxima Entropia

O Princípio da Máxima Entropia, como visto no capítulo de entropia, nos diz que a mais provável dentre todas as possíveis distribuições de probabilidades é aquela que satisfaz as restrições impostas e que tem a maior entropia. O princípio da máxima entropia (ou princípio da mínima divergência) também pode ser posto como a procura da distribuição de probabilidades mais próxima (com divergência mínima) de uma outra dada a priori.

O Princípio da Máxima Entropia pode ser posto do seguinte modo: Determinar a distribuição de probabilidades $P(\mathbf{x})$ mais próxima de $Q(\mathbf{x})$ que satisfaz as m restrições:

$$\sum_{\mathbf{x}} P(\mathbf{x})k(\mathbf{x}|i) = d(i)$$

onde $k(\mathbf{x}|i)$ é em geral (mas não necessariamente) uma função indicadora da i-ésima restrição, isto é, $k(\mathbf{x}|i) = 1$ se $\mathbf{x}$ pertence à i-ésima restrição e $k(\mathbf{x}|i) = 0$, caso contrário.

Para garantir que $P(\mathbf{x})$ seja uma distribuição de probabilidades é necessário adicionar a função indicadora da restrição 0:

$$k(\mathbf{x}|0) = 1, \text{ para todo } \mathbf{x}$$

A solução desse problema é a função $P(\mathbf{x})$ dada por:

$$P(\mathbf{x}) = Q(\mathbf{x}) \exp(\lambda_0) \exp\left\{\sum_{i=1}^{m} \lambda_i k(\mathbf{x}|i)\right\}$$

onde os λ_i são os multiplicadores de Lagrange que devem ser escolhidos de forma que satisfaçam as restrições:

$$\exp(\lambda_0) \sum_{\mathbf{x}} Q(\mathbf{x})k(\mathbf{x}|j) \exp\left\{\sum_{i=1}^{m} \lambda_i k(\mathbf{x}|i)\right\} = d(j), \text{ para } j = 1, ..., m$$

Podemos aplicar o Princípio da Máxima Entropia para obter o valor de $P(W)$. Para isso é necessário: i) obter restrições ao modelo de linguagem que sejam relevantes e ii) obter os parâmetros λ_i.

Vamos apresentar a seguir o método iterative scaling para a obtenção dos parâmetros λ_i. Esse é um método simples e não muito eficiente, no entanto há versões eficientes desse método quando as funções de restrição $k(\mathbf{x}|i)$ só assunem os valores 0 e 1, como é no nosso caso.

Algoritmo Iterative Scaling

1. Escolher valores iniciais de λ_i
2. Para $j = 0$ até m, fazer:
   - mantendo $\lambda_i, i \neq j$ fixado, achar λ_j^* satisfazendo à j-ésima restrição.
   - fixar $\lambda_j = \lambda_j^*$
3. Se todas as restrições estão suficientemente satisfeitas, terminar. Senão voltar ao passo 2.

Vamos agora exemplificar a aplicação do Princípio da Máxima Entropia em modelamento da linguagem criando um modelo de linguagem de bigramas, isto é, a probabilidade $P(x, y)$, a partir das seguintes restrições:

$$\begin{aligned} P(x,y) &= f(x,y) \qquad \text{se } C(x,y) \geq K \\ P(x) &= f(x) \qquad \text{se } C(x) \geq L \\ P(y) &= f(y) \qquad \text{se } C(y) \geq L \\ \sum_{x,y} P(x,y) &= 1 \end{aligned}$$

onde K e L são constantes, $f(.)$ indica a frequência relativa e $C(x)$ indica quantas vezes x ocorre no texto de treinamento.

Nesse caso, a expressão completa de $P(x, y)$ é:

$$P(x,y) = g_0 g(x,y)^{k(x,y)} g_1(x)^{k_1(x)} g_2(y)^{k_2(y)}$$

onde

$$\begin{cases} g(x,y) = \exp(\lambda_{x,y}) \\ k(x,y) = 1, \text{ se } C(x,y) \geq K \text{ e } k(x,y) = 0, \text{ caso contrário.} \\ k_1(x) = 1, \text{ se } C(x) \geq L \text{ e } k_1(x) = 0, \text{ caso contrário.} \\ k_2(y) = 1, \text{ se } C(y) \geq L \text{ e } k_2(y) = 0, \text{ caso contrário.} \\ g_0 \text{ é uma constante normalizadora para assegurar que } \sum_{x,y} P(x,y). \end{cases}$$

Modelos de linguagem de trigramas, assim como outros tipos de modelos de linguagem podem ser construídos do mesmo modo, escolhendo as restrições adequadas.

Nas próximas três seções vamos apresentar aplicações do Princípio da Máxima Entropia no modelamento da linguagem.

5.6 Adaptação do Modelo de Linguagem a um Novo Domínio

A adaptação do modelo de linguagem é necessária quando os dados de treinamento são insuficientes para o cálculo do modelo de linguagem completo e se o domínio desse modelo é parecido com um outro já existente para o qual existe um modelo completo. Isto é, temos um bom modelo de linguagem Q derivado de um relativamente rico domínio #1 (texto de treinamento

#1 suficientemente grande) e desejamos obter um modelo de linguagem P para um domínio #2 para o qual não há dados suficientes para uma estimação confiável. Se os domínios são parecidos, podemos calcular P com o Princípio da Máxima Entropia. P pode ser estimado como a distribuição de probabilidades mais próxima (mínima divergência) de Q e que satisfaz certas restrições impostas.

Vamos apresentar um exemplo simples de adaptação de modelo de linguagem. Vamos supor que temos um bom modelo de bigramas $Q(x, y)$ para o domínio #1 e que desejamos estimar $P(x, y)$ para o domíno #2. No entanto, o domínio #2 não é suficientemente grande e é adequado somente para o cálculo de unigramas e não de bigramas.

Nesse caso podemos considerar que $P(x, y)$ é a distribuição mais próxima de $Q(x, y)$ que satisfaz as restrições:

$$\begin{aligned} P(w_1) &= f_2(w_1) \\ P(w_2) &= f_2(w_2) \end{aligned} \tag{5.1}$$

onde $f_2(.)$ é a frequência relativa calculada no domínio #2.

A distribuição $P(x, y)$ é da forma:

$$P(w_1, w_2) = Q(w_1, w_2) g_1(w_1) g_2(w_2)$$

onde as funções g_1 e g_2 são determinadas de forma que P satisfaça às restrições 5.1.

Adaptações de modelos de linguagem mais complexas podem ser vistas em [1].

5.7 Modelo de Linguagem com Gatilho

Um ajuste dinâmico de probabilidades é necessário quando em alguma parte de um texto, um certo sub-vocabulário que antes era incomum se torna muito usado. Isso acontece, por exemplo, se um usuário dita um texto de medicina (o que é feito raramente). Palavras específicas de medicina se tornam comuns no texto ditado. Quando o usuário dita textos que não sejam de medicina, essas palavras (de medicina) voltam a ter uma probabilidade muito pequena de ocorrer. Para resolver esse problema devemos identificar certas palavras v como palavras gatilhos, cujo aparecimento no texto afeta consideravelmente a distribuição de probabilidades de palavras futuras.

Vamos denotar as palavras gatilho como $t_1, t_2, \cdots, t_M$ e definir as funções $b_i(h)$, que indicam se a palavra gatilho t_i pertence à história h.

$$b_i(h) = \begin{cases} 1, & \text{se } t_i \in h \\ 0, & \text{caso contrário} \end{cases}$$

Se o modelo de linguagem é baseado em trigramas, definimos a história equivalente como:

$$h^* = w_{-1}, w_{-2}, b_1(h), b_2(h), \ldots, b_M(h)$$

Podemos usar o princípio de máxima entropia para estimar $P(w_0|h)$. Para isso devemos impor restrições baseadas na frequência relativa de aparecimento das palavras gatilho na história e das palavras alvo, que tem suas probabilidades influenciadas pelas palavras gatilhos.

Antes de aplicar o princípio de máxima entropia para a estimação de $P(w_0|h)$ devemos obter, do texto de treinamento, uma lista de palavras gatilho $t_1, t_2, \cdots, t_M$ e das suas respectivas palavras alvo $v_1(t_i), v_2(t_i), \ldots, v_{N_i}(t_i)$, para $i = 1, 2, \ldots, M$.

A primeira idéia para se determinar as palavras gatilho e suas palavras alvo é achar pares com grande informação mútua. No entanto isso não mede o quanto a ausência do gatilho na história influencia o aparecimento das palavras alvo. Para resolver esse problema definimos as novas funções indicadoras:

$$b_w(v) = \begin{cases} 1, \text{ se } w = v \\ 0, \text{ caso contrário} \end{cases}$$

$$b_h(t) = \begin{cases} 1, \text{ se } t \in h \\ 0, \text{ caso contrário} \end{cases}$$

A partir dessas funções auxiliares definimos a informação mútua empírica média:

$$I(b_w(v), b_h(t)) = \begin{cases} \sum\limits_{b_w, b_h} f(b_w(v), b_h(t)) \log \frac{f(b_w(v), b_h(t))}{f(b_w(v)) f(b_h(t))}, \text{ se } C(b_w(v) = 1, b_h(t) = 1) \geq K \\ 0, \text{ caso contrário} \end{cases}$$

onde a soma é sobre todos os quatro valores possíveis de $b_w(v), b_h(t)$.

Selecionamos os pares gatilho-alvo para os quais a informação mútua empírica média está acima de um limiar escolhido.

Uma vez que as palavras gatilho estão determinadas, a probabilidade desejada $P(w_0, h^*)$ é a distribuição de máxima entropia que satisfaz as seguintes restrições:

$$\begin{cases} P(h^*) = f(h^*) \\ P(w_0, w_{-1}, w_{-2}) = f(w_0, w_{-1}, w_{-2}), \text{ se } C(w_0, w_{-1}, w_{-2}) \geq K \\ P(w_0, w_{-1}) = f(w_0, w_{-1}), \text{ se } C(w_0, w_{-1}) \geq K \\ P(w_0) = f(w_0), \text{ se } C(w_0) \geq I \\ P(w_0 = v_j(t_i), b_i(h)) = f(w_0 = v_j(t_i), b_i(h)), \text{ se } C(w_0 = v_j(t_i), b_i(h)) \geq L \\ \qquad\qquad \text{e para } i = 1, \ldots, M; \qquad j = 1, \ldots, N_i; \qquad b_i(h) = 0, 1 \end{cases}$$

onde a frequência relativa f e o contador C são obtidos dos dados de treinamento.

5.8 Modelo de Linguagem com Cache

O modelo de linguagem com cache é um tipo de modelo de linguagem dinâmica que é usado no caso de palavras ainda não observadas em um texto. Esse modelo de linguagem é usado quando ditamos um texto. Construímos um modelo de linguagem desse tipo com os seguintes passos:

- A partir dos n-gramas ditados no texto atual construímos o modelo de linguagem de trigramas $P_c(w_0|w_{-1}, w_{-2})$. Devemos achar os valores de interpolação γ_i adequados para definir o modelo $P_c(w_0|w_{-1}, w_{-2})$ como:

  $$P_c(w_0|w_{-1}, w_{-2}) = \gamma_3 f_c(w_0|w_{-1}, w_{-2}) + \gamma_2 f_c(w_0|w_{-1}) + \gamma_1 f_c(w_0)$$

  onde f_c é a frequência da presença de palavras no texto atual.

- Interpolamos o modelo acima com o modelo estático $P_s(w_0|h)$ pré-computado no texto de treinamento obtendo o modelo de linguagem completo:

$$P(w_0|h) = (1 - \gamma_c)P_s(w_0|h) + \lambda_c P_c(w_0|w_{-1}, w_{-2})$$

  onde λ_c varia com o tamanho do cache.

Novas palavras são descobertas com a correção do texto ditado. Quando uma palavra é usada pela primeira vez, o reconhecedor de voz irá reconhecê-la erroneamente. O usuário deve corrigir essa palavra, e então o reconhecedor de voz irá incluí-la em seu dicionário (expandido). As frequências relativas f_c são computadas sobre esse crescente vocabulário dinâmico e são ajustadas continuamente a medida em que são feitos o reconhecimento e as eventuais correções.

Podemos fazer um refinamento nesse modelo de linguagem levando em conta a probabilidade $P(w_0 \in h)$ de que a palavra w_0 tenha sido observada no cache em tempo de execução. Este refinamento é desejável pois na prática observamos que os usuários têm a tendência de repetir palavras relevantes num dado contexto. A fórmula refinada é:

$$P_c(w_0|w_{-1}, w_{-2}) = P(w_0 \notin h)P_s(w_0|h) + P(w_0 \in h)\left[\gamma_3 f_c(w_0|w_{-1}, w_{-2}) + \gamma_2 f_c(w_0|w_{-1}) + \gamma_1 f_c(w_0)\right]$$

Um melhoramento adicional é possível com o princípio da máxima entropia. Seguindo a idéia e a notação da seção 5.6, tomamos o dado de treinamento original como o domínio #1 e os dados do cache (texto atual) como o domínio #2. Daí estimamos o modelo de linguagem de bigramas $P_c^*(w_0|w_{-1})$ como sendo a distribuição com máxima entropia que satisfaz as seguintes restrições:

$$P_c^*(w_0) = \begin{cases} P(w_0 \in h)f_c(w_0), \text{ para } w_0 \in h \\ P(w_0 \notin h)f_s(w_0)/\left(\sum_{v \notin h} f_s(v)\right), \text{ caso contrário} \end{cases}$$

$$P_c^*(w_{-1}) = f_c(w_{-1})$$

Referências Bibliográficas

[1] Frederick Jelinek, *Statistical Methods for Speech Recognition.* MIT Press, 1997.

[2] Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition,* Prentice-Hall, 1993.

[3] Claudio Becchetti and Lucio Prima Ricotti, *Speech Recognition,* John Wiley&Sons, 1999.

[4] David Munford, *"Pattern Theory: the Mathematics of Perception"* in Proceedings of the International Congress of Mathematicians, Beijing, 2002, vol. 1, Higher Educ. Press, Bejing, 2002. ICM, 2002.

[5] R.M. Warren, *"Restoration of missing speech sounds",* Science, 167, 1970.

[6] M. C. Nechyba, *Spring 2000 Lecture Notes - EEL6935: Machine Learning in Robotics II*, `http://www.mil.ufl.edu/~nechyba/readings_s00.html`

[7] Claude E. Shannon, *"A Mathematical Theory of Communication"*, The Bell System Technical Journal, vol. 27, pp. 379-423 and 623-56, July, October, 1948. `http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html`

[8] J. N. Kapur and H. K. Kesavan, *Entropy Optimization Principles with Applications.* Academic Press, 1992.

[9] Simon Haykin, *Neural Networks: a comprehensive foundation.* Prentice Hall, 1999.

[10] Howard L. Resnikoff, *The Illusion of Reality.* Springer-Verlag, 1989.

[11] F. W. Sears e G. L. Salinger, *Termodinâmica, Teoria Cinética e Termodinâmica Estatística.* Guanabara Dois, 1979.

[12] Sir Arthur Stanley Eddington, *The Nature of the Physical World.* Maxmillan, New York, 1948, p. 74. Frase obtida em `http://www.dam.brown.edu/people/yiannis/friends.html`

[13] Albert Einstein, quoted in M.J. Klein, *Thermodynamics in Einstein's Universe*, in Science, 157 (1967), p. 509. Frase obtida em `http://www.dam.brown.edu/people/yiannis/friends.html`

[14] David Applebaum, *Probability and Information*, Cambridge University Press, 1996.