2
Reconhecimento de Voz

Para introduzir o tema de reconhecimento de voz, a seção 2.1 começa com o sistema mais simples: o de palavras isoladas. Ou seja, o locutor pronuncia apenas uma palavra e o reconhecedor tenta identificá-la com base numa lista de palavras conhecidas. Nas seções seguintes, é mostrado como esse sistema pode ser expandido para frases de vocabulário amplo.

2.1
Reconhecimento de Palavras Isoladas

A capacidade de entender o que foi dito por alguém é adquirida pelo ser humano ao longo dos seus primeiros anos de idade. Replicar essa capacidade para uma máquina não é tão simples. O problema principal é que existem infinitas variações de pronúncia; um "A" pode ser dito rapidamente ou mais lentamente, e por vozes graves ou agudas. Isso afeta diretamente suas formas de onda, tornando mais complicado diferenciá-la das outras vogais. A Figura 2 ilustra bem o caso.

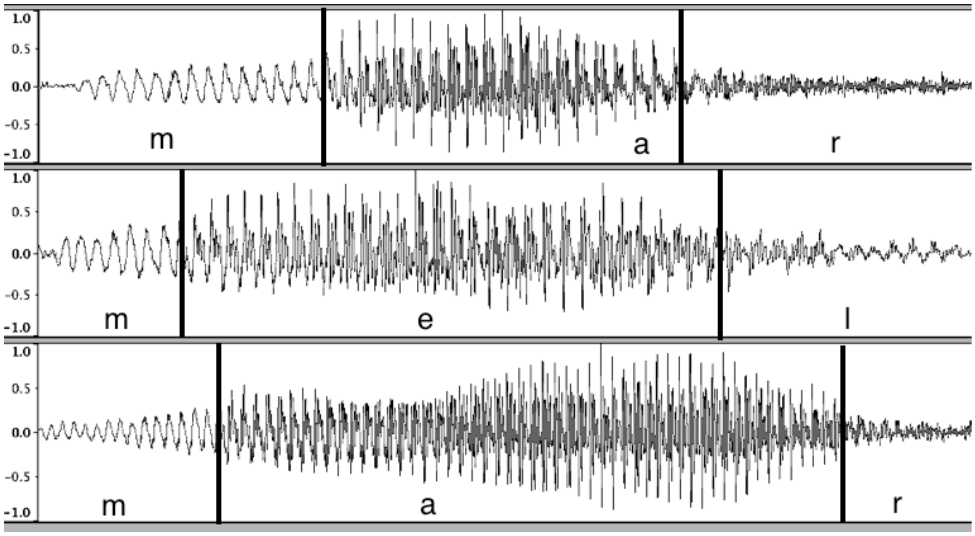

Figura 2: Forma de onda das palavras "mar" (voz masculina), "mel" (voz masculina) e "mar" (voz feminina).

A melhor solução encontrada para esse problema foi treinar modelos que representem elementos do idioma (digamos, palavras), e depois tentar classificar o

sinal num desses modelos. Por exemplo, suponha que o objetivo fosse reconhecer comandos do tipo sim/não. Seria necessário criar dois modelos: um que agrupasse as características sonoras de um "sim" e outro com as características de um "não". Após o treinamento, basta comparar as características do sinal de voz com a dos dois modelos e escolher o mais similar, conforme mostra a Figura 3.

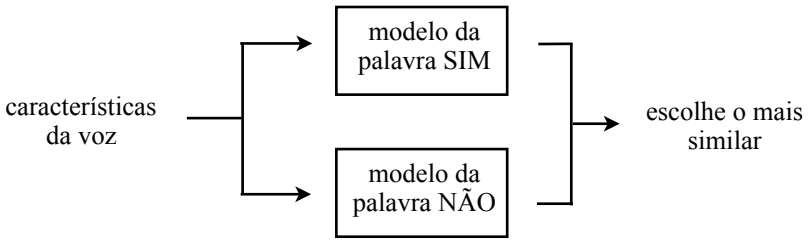

Figura 3: Esquema geral de um reconhecedor de voz para comandos do tipo SIM/NÃO.

Mas como ensinar o computador a agrupar todas as formas de se dizer uma palavra? E como diferenciá-la das demais? Dentre os algoritmos criados para RVC, o mais bem sucedido é o Modelo de Markov Escondido (em inglês, Hidden Markov Model, ou HMM) [7][8].

A ideia por trás do HMM é representar variações de pronúncia e de duração de cada som numa "máquina de estados". Como exemplo, considera-se a representação da palavra "mar" na Figura 4, formada por três fones (o modo de pronunciar o "m", "a" e "r"). Cada fone é associado a um estado e, à medida que se fala, ocorrem transições de estados. As transições podem ocorrer de diversos modos: uma pessoa poderia pronunciar mais demoradamente no "a" (maaaar), enquanto outra poderia se alongar mais no "r" (marrrr). Esses casos representariam as sequências de estados {1,2,2,2,2,3} e {1,2,3,3,3,3}, respectivamente. Como não conhecemos o modo de falar da pessoa em questão, temos que preparar o modelo para suportar todas as possibilidades.

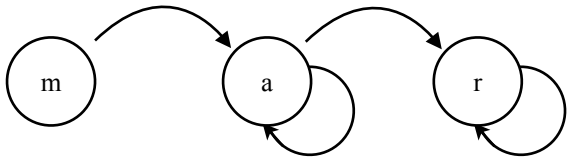

Figura 4: Possível representação da palavra "mar" no Modelo de Markov Escondido e suas transições de estado.

O número *N* de estados pode variar, assim como podem haver transições para estados à direita ou entre estados não-vizinhos. No entanto, os modelos mais comuns costumam ter poucos estados e um sentido único de transições.

Além da duração, existem variações de pronúncia de acordo com o sexo, idade e regionalidade do locutor. Por isso, cada estado guarda informações que representem essas diferenças, como mostra a Figura 5.

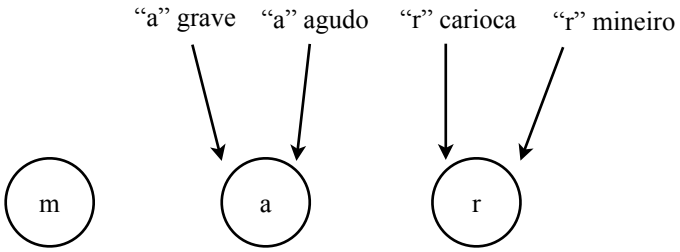

Figura 5: Possível representação da palavra “mar” no Modelo de Markov Escondido e algumas características importantes para cada estado.

Por fim, às vezes podemos suprir alguns sons iniciais da palavra. Não é um caso comum para a palavra “mar”; “está” seria um bom exemplo, onde muitos podem suprir o “e” inicial (ssstá) ou até mesmo o “es” (tá). Logo, existe a chance de se começar por algum estado do meio.

Para abordar esses três conceitos matematicamente, o Modelo de Markov Escondido usa o conceito de probabilidade: existe uma probabilidade de mudança de estado, uma probabilidade de observar características do som e uma probabilidade de começar de um estado inicial. Para isso, o modelo é exemplificado na Figura 6.

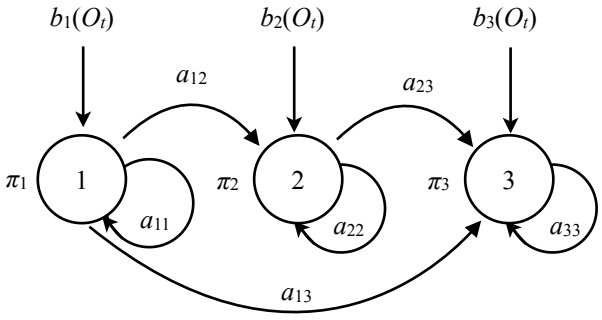

Figura 6: Representação matemática do Modelo de Markov Escondido.

O percurso começa no instante $t = 1$, num estado aleatório s_1. π_i é a probabilidade do estado s_1 ser o i-ésimo estado, resolvendo o problema do sons iniciais serem omitidos.

Em seguida, o sistema passa para o instante $t = 2$ e faz uma transição aleatória de estado. As transições continuam até o instante final T. a_{ij} é a probabilidade de se mudar do estado i para o estado j (se i for igual a j, a_{ij} é a probabilidade de repetir o estado corrente). Desse modo, representam-se as

diferentes durações de cada fonema (“maaaar” versus “marrrr”). Para simplificar os cálculos computacionais, parte-se do pressuposto que a transição atual não depende das anteriores. Por exemplo, a probabilidade de ir do estado 2 para o 3 não é influenciada pelo fato de antes ter ocorrido uma transição do 1 para o 2. Matematicamente falando, a probabilidade de chegada ao estado $s_t = j$ dada a ocorrência anterior de estados $s_{t-1} = i$, $s_{t-2} = h$, etc é igual a

$$P\left(s_t = j \middle| s_{t-1} = i, s_{t-2} = h, s_{t-3} = g, ...\right) = P\left(s_t = j \middle| s_{t-1} = i\right) = a_{ij}, \quad \forall t \tag{1}$$

Após cada transição, o modelo chega a um estado e produz uma observação de um vetor aleatório O_t. $b_i(O_t)$ é a probabilidade de ocorrência de O_t quando s_t é igual a i.

$$b_i\left(O_t\right) = P\left(O_t \middle| s_t = i\right) \tag{2}$$

O_t representa numericamente as características do som (que, por exemplo, diferenciem o “a” grave do “a” agudo) no instante t. O conceito de probabilidade é usado também dentro de cada estado porque não se conhece a priori essas características. Sendo bem ajustado, $b_i(O_t)$ será capaz de representar todas as variações de pronúncia – em 2.4.1, $b_i(O_t)$ se transforma numa função densidade de probabilidade.

Com a representação matemática, pode-se detalhar o esquema anterior da Figura 2 para a Figura 7 a seguir. Cada palavra possível de ser dita é representada por um Modelo de Markov. Para cada modelo, é passado um conjunto de T vetores de características do sinal de voz. Então calcula-se a probabilidade de o modelo observar aquela sequência de vetores – isto é, a probabilidade de observar o primeiro vetor no instante 1, o segundo vetor no instante 2, etc, até o T-ésimo instante. O modelo com a maior das probabilidades será o mais similar ao sinal de voz e, assim, representará a palavra mais provável.

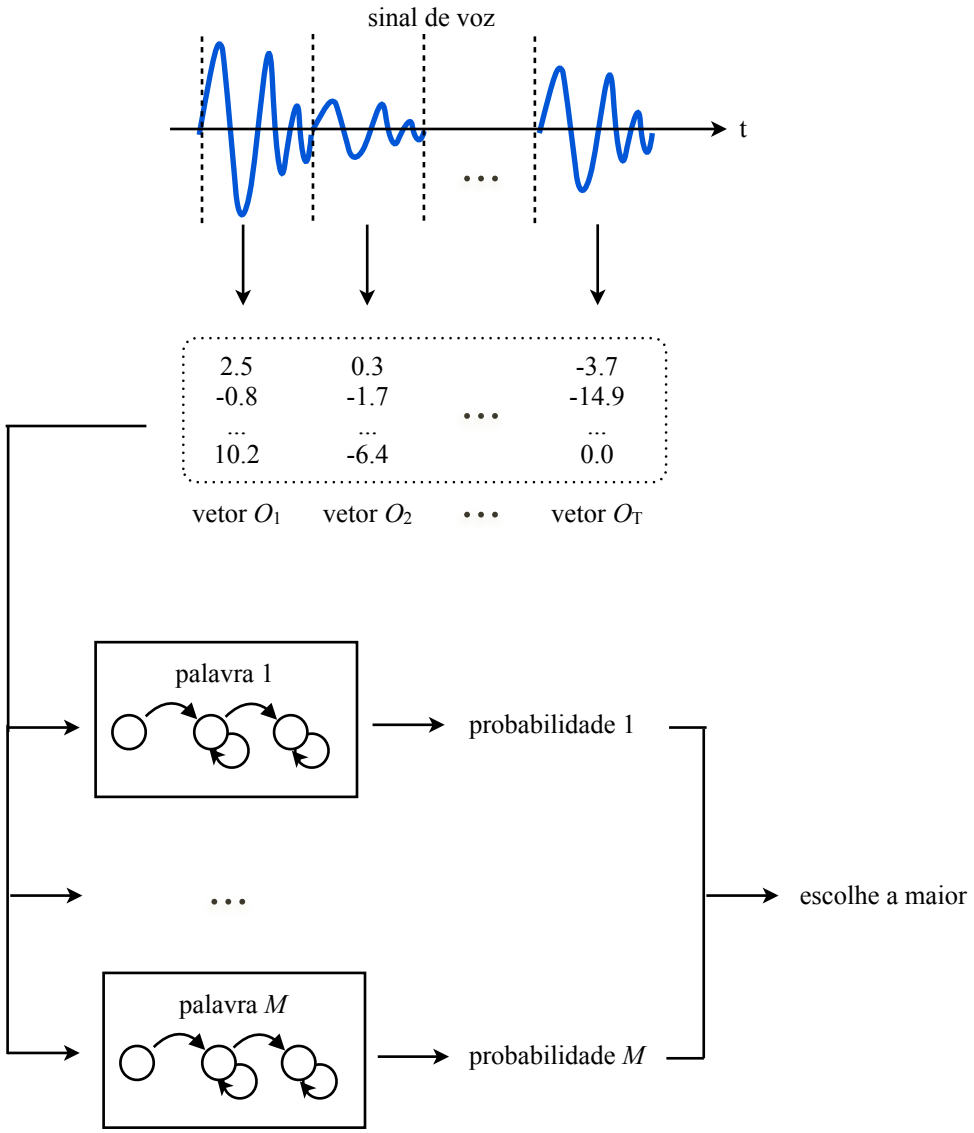

Figura 7: Esquema mais detalhado de um reconhecedor de voz de palavras isoladas, para M possíveis palavras.

Portanto, seja $O = \{O_1, O_2, \ldots, O_T\}$ a sequência de vetores extraídos do sinal de voz e $s = \{s_1, s_2, \ldots, s_T\}$ uma sequência de estados. A probabilidade $P(O)$ de ocorrer uma determinada sequência de observações, num determinado modelo, é dada por

$$P(O) = \sum_{s} P(O|s)P(s) = \sum_{s} \pi_{s_1} b_1(O_t) \prod_{t=2}^{T} a_{s_{t-1}s_t} b_{s_t}(O_t) \tag{3}$$

A demonstração de (3) se encontra no apêndice 9.1, como uma consulta opcional.

Embora simples de expressar, o cálculo acima é muito caro computacionalmente. Isso porque o número possível de "s"s (sequências de estados) cresce exponencialmente para N^T. Ou seja, para 2 transições e 3 estados,

existem $3^2 = 9$ sequências: {1,1}, {1,2}, {1,3}, {2,1}, {2,2}, {2,3}, {3,1}, {3,2} e {3,3}. Já para 25 transições e 3 estados, o total sobe para 3^{25} = mais de 800 bilhões!

Felizmente, existe o algoritmo Forward-Backward [8, p. 262], que realiza uma contagem bem mais eficiente e reduz a ordem de complexidade para N^2T. O princípio por traz dele é bem simples: as sequências de transições {1,2,3} e {1,2,2} têm o começo {1,2} em comum. Logo, não é necessário repetir o cálculo inicial para cada uma.

Sua aplicação é feita com a expressão a seguir.

$$\alpha_t(j) = P(O_1, O_2, \ldots, O_t, s_t = j)$$

1) Inicialização: $\alpha_1(j) = b_j(O_1)\pi_j$

2) Indução: $\alpha_t(j) = b_j(O_t)\sum_{i=1}^{N}\alpha_{t-1}(i)a_{ij}$

3) Finalização: $P(O) = P(O_1, O_2, \ldots, O_T) = \sum_{j=1}^{N}\alpha_T(j)$ (4)

A demonstração de (4) se encontra no apêndice 9.2, como uma consulta opcional.

2.2
Reconhecimento com Vocabulário Amplo

A seção 2.1 usou HMMs para representar palavras. Isso não é feito na prática quando o vocabulário é muito grande, já que seria trabalhoso demais gerar centenas ou milhares de modelos. Em vez disso, os modelo serão subunidades de palavra, capazes de serem combinadas entre si para gerar todo o vocabulário. Essas subunidades poderiam ser sílabas (casa = ca + za), pois foi mostrado em [9] que os resultados são satisfatórios para o caso da língua portuguesa. Outra possibilidade são os fones (casa = k + a + z + a), que têm um desempenho melhor na maioria dos idiomas (como o inglês e o francês [10]). Como este trabalho usa uma base de dados de voz em inglês, cada modelo do reconhecedor representa um fone a partir de agora.

Só que a nova abordagem traz um problema: se agora palavras são divididas em fones, como saber quando um deles termina e quando outro começa, a partir

do sinal de voz? A Figura 1 mostrou que a duração de cada fone pode variar de locutor para locutor, ou até mesmo de palavra para palavra. Trata-se da mesma questão mostrada em 2.1, quando o HMM foi apresentado para lidar com diferentes durações. Uma solução natural, portanto, seria conectar os modelos de fones para formar novos HMMs que representem as palavras. A Figura 8 ilustra o processo.

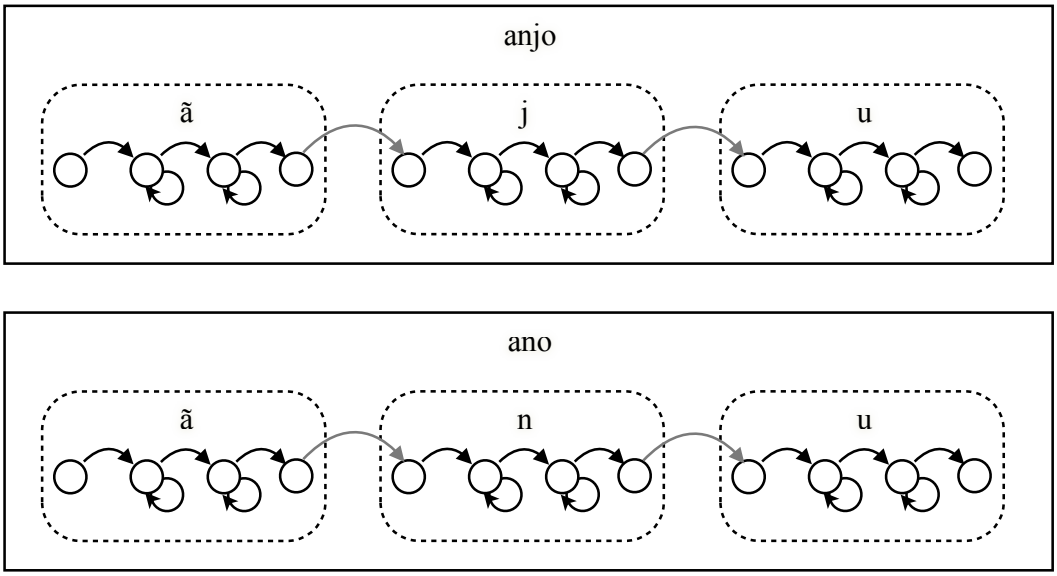

Figura 8: Conexão dos HMMs de fones para a formação de HMMs de palavras.

Note que o modelo do fone "ã" em "anjo" é exatamente o mesmo de "ano", pois o som é igual. Como agora existe um HMM para cada fone, a quantidade de modelos é fixa, não importa o tamanho do vocabulário. Para formar cada palavra, basta saber quais os fones de sua pronúncia e então conectar os modelos adequados.

Na prática, infelizmente, um fone não é completamente independente de seus vizinhos. A identificação do "j" em "aja" é um pouco diferente da identificação em "anjo", já que o primeiro sofre influência do "a" anterior e do "a" posterior, enquanto o segundo do "ã" e do "u". Surge daí o conceito de trifone: um fone que também é caracterizado pelo seu antecessor e pelo sucessor [11, p. 428].

Portanto, em vez de haver apenas um modelo para o "j", o sistema guardará variações baseadas nos fones anteriores e posteriores, como mostra a Figura 9. Por exemplo, o modelo "ã–j+u" representa o fone "j" com influência de um "ã" anterior e um "u" posterior. "a+j" é o fone "a" com influência apenas do "j" posterior, pois trata-se do começo da palavra. Conforme citado na seção 6.3, este

trabalho escolheu trabalhar com trifones intra-palavras, isto é, o primeiro fone de uma palavra não é combinado com o último fone da palavra anterior. Logo, são gerados também bifones (como "a+j") e monofones (como "a", para o caso de palavras formadas por um único fone).

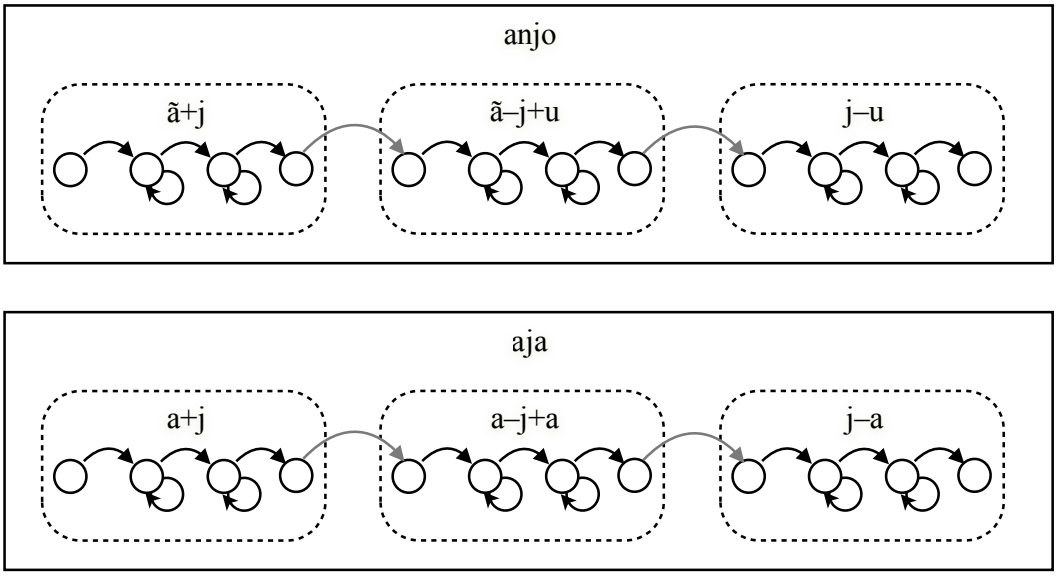

Figura 9: Conexão dos HMMs de bifones e trifones, para a formação de palavras.

Vale notar que os trifones trazem com um aumento considerável do custo computacional, já que o número de modelos para se verificar passa de dezenas para milhares.

2.3 Reconhecimento de Voz Contínua

As duas seções anteriores trataram do reconhecimento de palavras isoladas. O último passo é conectar os modelos de palavras para a formação de frases. O processo é um pouco diferente do que com os trifones; não se pode juntar as palavras em grupos específicos de frases, porque o número de sentenças que existem é infinito. Em vez disso, usa-se um tipo de rede cíclica: o final de cada palavra é conectado ao começo de todas, o que permite formar todas as sequências (frases) possíveis. A Figura 10 a seguir ilustra o conceito para uma rede de duas palavras.

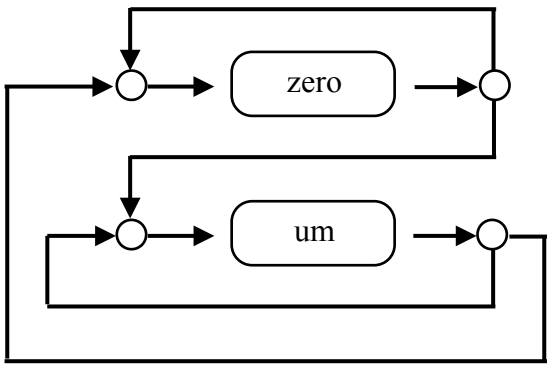

Figura 10:Conexão cíclica das palavras para a formação de frases. O final de cada palavra é ligado ao começo de todas.

Até agora, para encontrar a cadeia de palavras mais provável, o modelo novo utilizou apenas as características da voz. Isso pode gerar frases sem sentido lógico. Por exemplo, "não tenho nada" poderia ser reconhecido como "na um tem um na dá", dado que a pronúncia é extremamente similar. Ou seja, a ligação de HMMs não basta para resultados viáveis. É necessário ter certo conhecimento do idioma do locutor – um modelo de linguagem – para identificar quais as frases fazem mais sentido dados os fones pronunciados.

Embora existam soluções com regras gramaticais, a literatura mostrou que é mais eficiente usar estatísticas de ocorrência de palavras [12]. Isto é, em vez de assumir que adjetivos geralmente aparecem após substantivos, considera-se que a palavra "bonita" ocorra com mais frequência após "casa" do que após "longe", por exemplo.

Logo, seja $w = \{w_1 , w_2 , ... , w_V\}$ uma sequência de V palavras. A probabilidade de observar essa sequência é dada aproximadamente por

$$P(w) \approx P(w_1)P(w_2|w_1)\prod_{i=3}^{V} P(w_i|w_{i-2},w_{i-1}) \tag{5}$$

$$P(w_i|w_{i-2},w_{i-1}) \approx \frac{count(w_{i-2},w_{i-1},w_i)}{count(w_{i-2},w_{i-1})} \tag{6}$$

onde *count*( ) é o número de vezes em que uma sequência de palavras aparece num texto que represente bem o idioma. A demonstração de (5) e a de (6) é feita no apêndice 9.3, como uma consulta opcional.

$P(w)$ é a probabilidade de ocorrência de uma sequência de palavras qualquer. A expressão não considera o que foi dito pelo locutor. O objetivo é calcular $P(w|O)$ – isto é, a probabilidade da sequência de palavras dados os

vetores de característica do sinal de voz. Esse valor é obtido usando o teorema de Bayes [13], conhecido por

$$P(w|O) = \frac{P(w)P(O|w)}{P(O)} \tag{7}$$

Logo, a sequência de palavras mais provável $\hat{w}$ é aquela que maximiza a expressão anterior, ou seja,

$$\hat{w} = \arg\max_{w} P(w|O) = \arg\max_{w} \frac{P(w)P(O|w)}{P(O)} = \arg\max_{w} P(w)P(O|w) \tag{8}$$

Como o sinal de voz pode ser poluído com o ruído (degradando os fones), é comum dar-se mais peso ao modelo de linguagem. Portanto, o termo $P(w)$ é elevado a um expoente $L > 1$. Deste modo, se a probabilidade da sequência de palavras for baixa (ou seja, fizer pouco sentido), o valor da multiplicação será penalizado.

$$\begin{aligned} \hat{w} &= \arg\max_{w} P(w)^{L} P(O|W) \\ &= \arg\max_{w} \left(\prod_{i=1}^{m} P(w_i | w_{i-2}, w_{i-1}) \right)^{L} P(O|w) \end{aligned} \tag{9}$$

L é chamado de peso do modelo de linguagem, e é ajustado empiricamente para um desempenho melhor.

$P(O|w)$ seria calculado simplesmente unindo os HMMs das palavras de w (vide Figura 11) e utilizando o algoritmo forward-backward. Como existe o termo $P(w)^{L}$ multiplicado, transições entre palavras serão ponderadas pelos termos $P(w_i|w_{i-2}, w_{i-1})$.

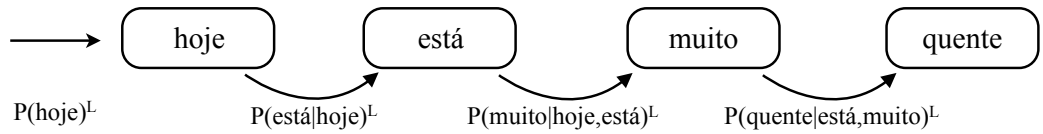

Figura 11:Conexão dos modelos de palavras para a formação da frase w = “Hoje está muito quente”.

Obviamente, $\hat{w}$ não é encontrado avaliando todas as infinitas frases existentes. O que se faz na prática é evoluir a rede da Figura 10 para a da Figura 12, adicionando as probabilidades do modelo de linguagem nas transições entre uma palavra e outra. Como agora é necessário saber quais foram as duas palavras anteriores, “zero” e “um” aparecem duplicadas.

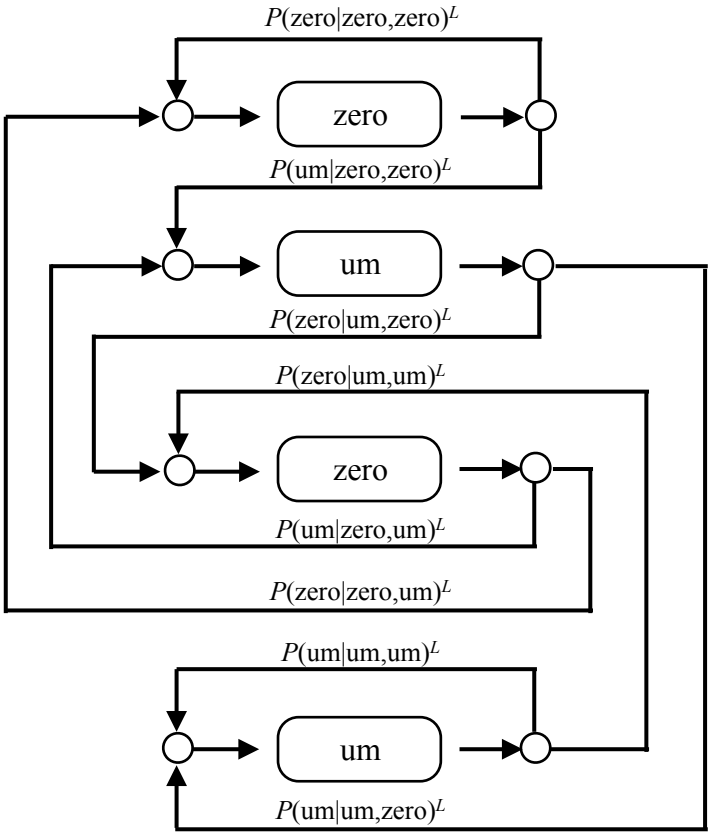

Figura 12:Rede de trigramas, para o caso de apenas duas palavras.

Como cada modelo de palavra foi formado por HMMs conectados, a rede inteira passa a ser um grande HMM. Logo, a busca pela melhor sequência de palavras de $\hat{w}$ passa a ser a busca pela melhor sequência de estados $\hat{s}$.

$$\hat{s} = \arg\max_{s} P(O,s) = \arg\max_{s} P(O|s)P(s) \tag{10}$$

Essa procura é mais simples com a representação da rede na treliça da Figura 13.

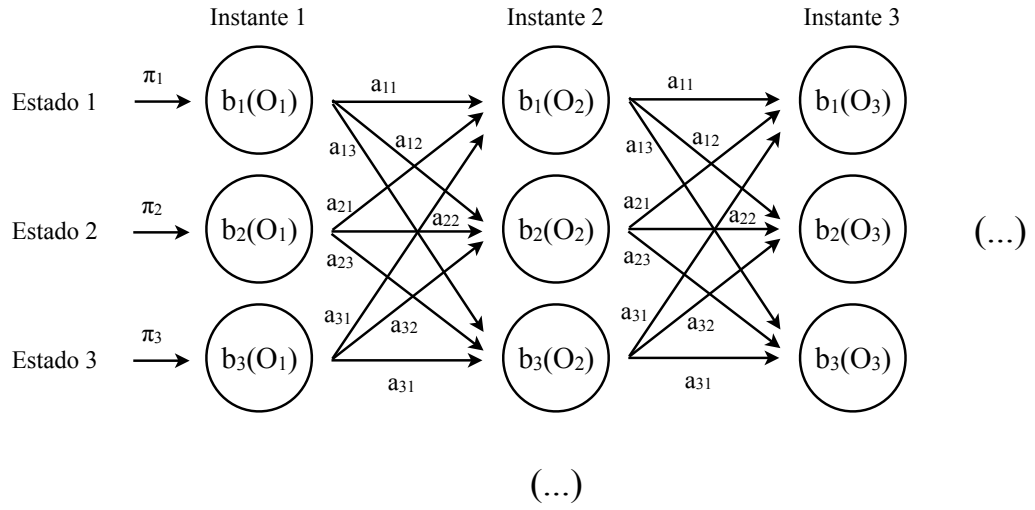

Figura 13:Transições de estado representados numa treliça.

$P(O|s)P(s)$ seria calculado com a multiplicação dos valores "π"s, "a"s e "b"s ao longo do caminho de s. O percurso que tiver a maior probabilidade seria aquele associado a sequência de palavras mais provável.

Porém, em vez de computar todos os possíveis caminhos, será feita uma busca eficiente através do algoritmo de Viterbi [8, p.264][14]. Seu princípio geral consiste em começar do instante 1 e avançar ao longo da treliça guardando apenas os percursos de maior probabilidade. Isto é, se quisermos saber qual o melhor trajeto até o estado 1 no instante 3, basta considerar as maiores probabilidades acumuladas até cada estado do instante 2.

Formalmente falando, seu procedimento é definido por

$$
\begin{aligned}
&\text{1) Inicialização: } \begin{cases} V_1(j) = b_j(O_1)\pi_j \\ B_1(j) = 0 \end{cases} \\
&\text{2) Indução: } \begin{cases} V_t(j) = b_j(O_t)\max\limits_{1\le i\le N} V_{t-1}(i)a_{ij} \\ B_t(j) = \arg\max\limits_{1\le i\le N} V_{t-1}(i)a_{ij} \end{cases} \\
&\text{3) Finalização: } \widehat{s_T} = \arg\max_{1\le i\le N} B_T(i) \\
&\text{4) Backtracing: } \begin{cases} \widehat{s_t} = B_{t+1}\left(\widehat{s_{t+1}}\right), t = T-1, T-2, \ldots, 1 \\ \hat{s} = \left\{\widehat{s_1}, \widehat{s_2}, \ldots, \widehat{s_T}\right\} \end{cases}
\end{aligned}
\tag{11}
$$

A demonstração de (11) se encontra no apêndice 9.4.

2.4
Parâmetros do HMM

Até agora, assumiu-se que a_{ij} e $b_i(O_t)$ estavam corretamente ajustados em cada modelo, para representar as características de cada fonema. Esta seção explica como o ajuste é feito.

2.4.1.
Escolha da Função para $b_i(O_t)$

Enquanto a_{ij} foi definido apenas como uma probabilidade fixa (isto é, um número real entre 0 e 1), $b_i(O_t)$ representava uma função que recebia um vetor e retornava uma probabilidade. Geralmente, cada elemento do vetor pode assumir valores dentro de um intervalo contínuo, o que implica numa probabilidade infinitesimal para cada um. Dado que computadores não processam infinitesimais, $b_i(O_t)$ passará a ser uma função densidade de probabilidade (fdp), cujos valores são proporcionais à probabilidade de ocorrência.

Mas qual função deve ser usada? Segundo o teorema do limite central [15], a combinação de diversos fatores aleatórios (comum em muitos fenômenos naturais) resulta numa fdp Gaussiana (ou normal), dada por

$$\phi(X) = \frac{1}{\left(2\pi|\Sigma|\right)^{\frac{k}{2}}} \exp\left(-\frac{1}{2}(X-\mu)^T \Lambda^{-1}(X-\mu)\right) \tag{12}$$

onde Λ é a matrix de covariância de $\phi(X)$, μ é o vetor de média e k é a dimensão do vetor X.

A distribuição normal representa muito bem variáveis aleatórias que tendem a se concentrar ao redor de um valor específico. No entanto, o comportamento da voz humana pode ser um pouco mais complexo. Por exemplo, suponha que um dos parâmetros extraídos do sinal sonoro fosse sua frequência fundamental. É razoável assumir que a voz masculina tenha valores mais concentrados numa frequência baixa, enquanto as mulheres numa mais alta. Uma função gaussiana poderia ser bem estimada para cada sexo, conforme o gráfico da Figura 14.

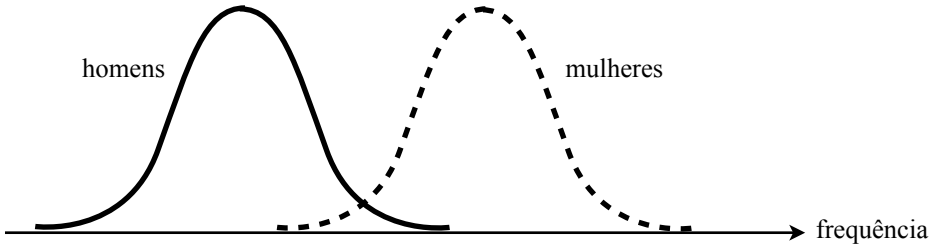

Figura 14:Esboço das densidades de probabilidade gaussianas da frequência fundamental para homens e para mulheres.

Por outro lado, uma única distribuição normal não representa bem o comportamento de ambos os sexos; a frequência geral tende a gerar dois picos, e não um somente. A melhor abordagem seria combinar as duas gaussianas numa nova distribuição como a da Figura 15.

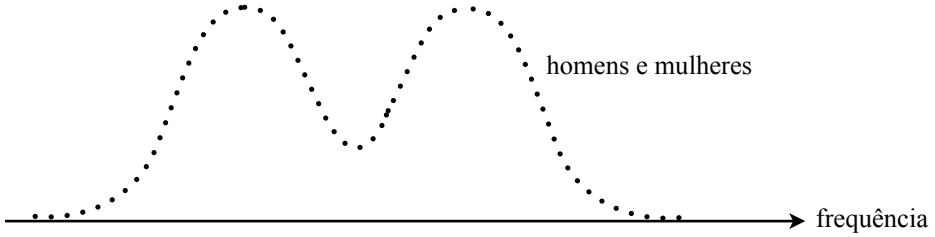

Figura 15:Esboço da densidade de probabilidade unificada da frequência fundamental para homens e mulheres.

$b_i(O_t)$ pode então ser mais bem modelado com esse princípio, através da combinação linear de M distribuições gaussianas com pesos c_{mi}.

$$\begin{aligned} b_i(O_t) &= \sum_{m=1}^{M} c_{mi}\phi(O_t, \mu_{mi}, \Lambda_{mi}) \\ &= \sum_{m=1}^{M} c_{mi} \frac{1}{\left(2\pi|\Lambda_{mi}|\right)^{\frac{k}{2}}} \exp\left(-\frac{1}{2}(O_t-\mu_{mi})^T \Lambda_{mi}^{-1}(O_t-\mu_{mi})\right) \\ b_i(O_t) > 0 &\Rightarrow c_{mi} > 0 \\ \int_{-\infty}^{\infty} b_i(O_t)\,dO_t = 1 &\Rightarrow \sum_{m=1}^{M} c_{mi} = 1 \end{aligned} \tag{13}$$

Assim, cada HMM passa a ser caracterizado pelo conjunto de parâmetros $\theta = \{a_{ij}, \pi_i, c_{mi}, \mu_{mi}$ e $\Lambda_{mi}\}$, com $1 \leq (i, j) \leq N$ e $1 \leq m \leq M$. A estimação desses parâmetros é feita com um conjunto de sinais de voz de treinamento. Nesses sinais, a localização e duração dos fones são conhecidas, permitindo que os vetores O_t extraídos sejam usados para ajustar os modelos apropriados, como se vê na Figura 16.

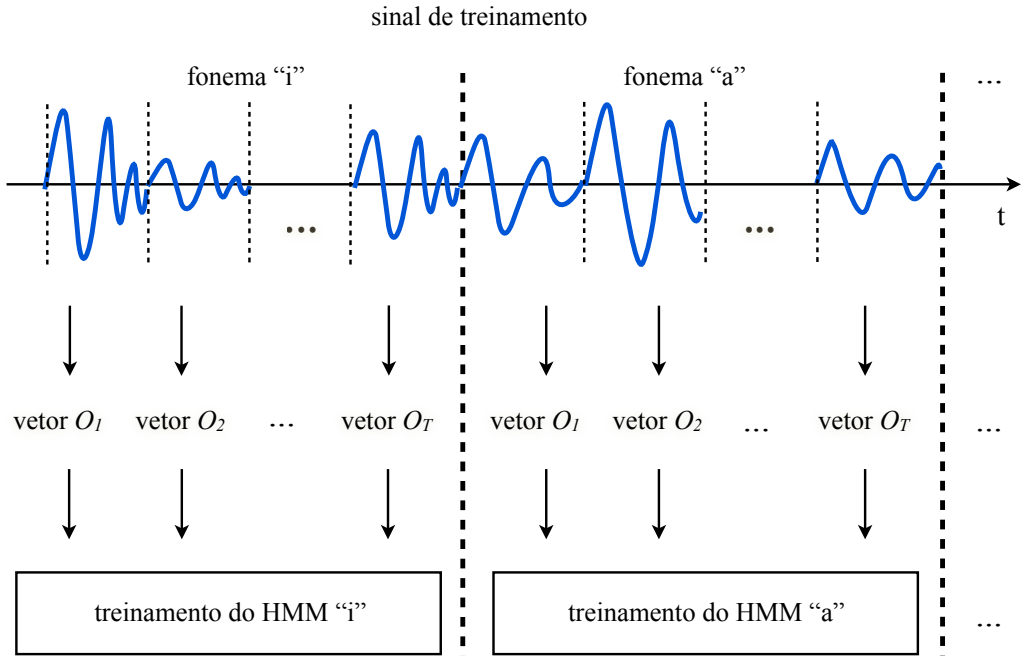

Figura 16:Extração dos vetores Ot para o treinamento de cada fonema.

2.4.2. Otimização dos Parâmetros

Durante o treinamento, busca-se o conjunto de parâmetros θ que maximizem a probabilidade de observar o conjunto de vetores $O = \{O_1, O_2, \ldots, O_T\}$, isto é,

$$\hat{\theta} = \arg\max_{\theta} P(O|\theta) \tag{14}$$

Infelizmente, não se sabe no treinamento quais as sequências de estado que geram os vetores de O. Seria necessário aplicar o teorema da probabilidade total para avaliar todos os possíveis percursos, através de

$$\hat{\theta} = \arg\max_{\theta} P(O|\theta) = \arg\max_{\theta} \sum_{\text{todo s}} P(O|s,\theta)P(s|\theta) \tag{15}$$

A equação (15) possui o mesmo termo encontrado em (3), o que permitiria resolvê-la com o algoritmo forward-backward. Mas o cálculo teria que ser repetido para todas as possíveis combinações de parâmetros de θ, o que seria inviável. Por isso, a técnica usada aqui é o algoritmo Baum-Welch [8, p. 264][16], descrito no apêndice 9.5. Após n iterações, obtém-se um conjunto de parâmetros θ otimizado (localmente, e não globalmente).

2.4.3.
Descasamento entre Treino e Teste

Após a otimização, os HMMs estarão prontos para reconhecer características semelhantes às amostras de treinamento. Contudo, essas características podem ser diferentes quando um locutor utilizar o sistema. Por causa do ruído do ambiente em que ele se encontra, novas informações são adicionadas ao sinal de voz, como mostra a Figura 17.

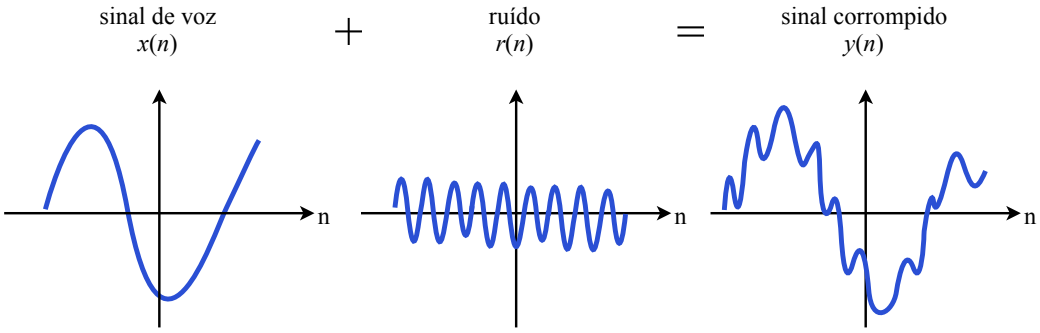

Figura 17:Formação de um sinal corrompido através da adição de ruído ao sinal de voz.

Um grande problema surge então: durante os testes, os modelos passam a receber um sinal $y(n)$ bem diferente do que era esperado com $x(n)$. Por exemplo, o fonema "a" ruidoso pode trazer características esperadas em um "e" limpo. Esse descasamento entre treino e teste provoca vários erros no reconhecimento, prejudicando o desempenho do sistema.

Alguns trabalhos abordam a questão alterando o funcionamento do HMM, na tentativa de prepará-lo para vários tipos de ruído [17]. Já outros autores propuseram novos métodos para extrair dados da fala, de modo que os valores não se alterem tanto com a adição de ruído [3][4]. O autor deste trabalho considerou o segundo caminho muito mais simples e eficiente, e por isso ele é analisado no próximo capítulo.