Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Comunicações

IMPLEMENTAÇÃO EM TEMPO REAL DE UM SISTEMA DE RECONHECIMENTO DE DÍGITOS CONECTADOS

Autor: Rodrigo Varejão Andreão

Orientador: Prof. Dr. Luis Geraldo Pedroso Meloni

Banca Examinadora: Prof. Dr. Luis Geraldo Pedroso Meloni (FEEC/UNICAMP)
Prof. Dr. Carlos Alberto Ynoguti (INATEL)
Prof. Dr. Plínio Almeida Barbosa (IEL/UNICAMP)
Prof. Dr. Akebo Yamakami (FEEC/UNICAMP)

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica.

Campinas, Janeiro de 2001

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

An25i

Andreão, Rodrigo Varejão
Implementação em tempo real de um sistema de reconhecimento de dígitos conectados / Rodrigo Varejão Andreão.--Campinas, SP: [s.n.], 2001.

Orientador: Luis Geraldo Pedroso Meloni.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Reconhecimento automático da voz. 2. Sistemas de processamento da fala. 3. Markov, Processos de . I. Meloni, Luis Geraldo Pedroso. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

RESUMO

O trabalho envolve a concepção de um sistema de reconhecimento de dígitos conectados, na área de processamento de fala, e sua implementação em tempo real. O projeto do sistema é dividido em três partes: análise espectral, quantização vetorial e modelagem estatística. A modelagem estatística, convencionalmente chamada de decodificação, é a parte principal do sistema, onde cada palavra de um vocabulário de dígitos é representada por um modelo oculto de Markov discreto. O sistema de reconhecimento de fala é analisado inicialmente num ambiente de simulação, e um procedimento de treinamento dos modelos, que utiliza a segmentação automática, é implementado. A avaliação do sistema é feita através de bases de sinais, e altas taxas de acerto são obtidas. O sistema é, então, adaptado ao processamento em tempo real. Algumas condições de operação em tempo real são abordadas. Utiliza-se um detector de *endpoint* em conjunto com uma máquina de estados, proposta no trabalho, para segmentação das elocuções pronunciadas pelo locutor.

ABSTRACT

The work involves the project of a connected-digit recognition system, in the field of speech processing with its implementation for application in real time. The project is divided in three parts: spectral analysis, vector quantization and statistical modeling. The statistical modeling, also called decodification, is the main part of the system, in which each word of the vocabulary is represented by a hidden Markov model. The speech recognition system is first analyzed into a simulation environment, and a training procedure using automatic segmentation is implemented. The evaluation of the system uses two Data Bases, and high score are obtained. The system is then adapted for real time processing. Some real time operating conditions are discussed. The method uses an endpoint detector with a proposed state machine for segmentation of the speaker's utterances.

AGRADECIMENTOS

Agradeço a colaboração do Prof. Dr. Carlos Alberto Ynoguti, que sempre se mostrou disponível para os esclarecimentos necessários, por ter cedido sua base de dígitos conectados e alguns de seus trabalhos em reconhecimento de fala. As suas experiências em reconhecimento de fala contribuíram muito no meu trabalho de pesquisa.

Agradeço ao suporte dado pelo Prof. Dr. Plínio Almeida Barbosa, pesquisador do Laboratório de Fonética/IEL/UNICAMP, e sua contribuição em minha formação na área de reconhecimento de fala.

Gostaria de demonstrar minha gratidão ao Prof. Dr. Luis Geraldo Pedroso Meloni por ter me acolhido nesta faculdade, me envolvido no trabalho de pesquisa e pela troca de experiências tão proveitosa.

Os colegas desta faculdade sempre mostraram como é imprescindível um ambiente de união, onde a troca de conhecimentos e a ajuda mútua fazem parte do dia a dia, fazendo-nos sentir participantes do crescimento de cada um. Gostaria de agradecer em especial aos colegas do Departamento de Comunicações.

Os longos anos de pesquisa foram divididos por incontáveis momentos de amizade, acolhimento, partilha e aprendizado com aqueles que chamamos de amigos. Aos amigos de república Marcelo, Joaquim e Tarciano, ao amigo de laboratório Helder, aos amigos professores Evandro, Antonio Frasson e Marcelo Segatto e ao amigo de toda hora Romis, a minha eterna gratidão.

Por tudo que conquistei, eu dedico carinhosamente aos meus pais Leandro e Angela e aos meus irmãos Leandro, Beatriz e Bruno. Eles são o meu grande amor e instrumento de Deus na minha vida.

ÍNDICE

Capítulo 1
INTRODUÇÃO

As pesquisas em reconhecimento de fala vêm sendo desenvolvidas intensamente pela comunidade científica há mais de 30 anos, e os frutos desse investimento científico começam a ser colhidos.

No início das pesquisas, os sistemas de reconhecimento eram testados através de simulações, onde as condições eram sempre as mais favoráveis, longe do que se encontra em um sistema prático de reconhecimento de fala. Passados anos de pesquisa, somados aos avanços tecnológicos, uma nova realidade se abre ao público em geral com o aparecimento de sistemas de reconhecimento de fala incorporados aos serviços voltados aos usuários.

Os sistemas práticos de reconhecimento de fala contínua são conhecidos por exigir grandes vocabulários, alguns chegando a 250 mil palavras. O desempenho de tais sistemas depende muito da aplicação e do usuário que vai utilizar o serviço. Algumas dessas aplicações requerem o treinamento do sistema por parte do usuário para que sejam alcançados bons desempenhos.

Por outro lado, pode-se conseguir elevado desempenho dos sistemas de reconhecimento de fala, reduzindo-se a complexidade da aplicação. Estudos recentes indicam que os sistemas de reconhecimento atuais, voltados para aplicações restritas independentes de locutor e que utilizam vocabulários reduzidos, podem atingir taxas de acerto de palavras de até 98% [4].

Os requisitos de desempenho de tais sistemas são ainda mais severos quando se pretende operar em tempo real e num ambiente ruidoso. Neste trabalho, o enfoque é dado à implementação em tempo real.

1.1 O Reconhecimento de Fala e suas Aplicações

Várias aplicações estão sendo concebidas incorporando reconhecimento de fala: discagem através da fala em telefones celulares, serviços de centrais de atendimentos aos clientes de bancos, consultas de cartões de crédito e de lista telefônica, reservas de vôos em companhias aéreas, entre outros.

O serviço de discagem falada, por exemplo, possui um elevado desempenho, pois se consegue reduzir drasticamente a complexidade do sistema de reconhecimento de fala. Além disso, o vocabulário utilizado nesses sistemas é bastante restrito, podendo compreender alguns comandos e os dígitos do teclado. Em contrapartida, sistemas que utilizam grandes vocabulários, caso em que se incluem os sistemas de reservas de vôos em companhias aéreas via reconhecimento de fala, tendem a apresentar taxas de erro mais elevadas. Entretanto, devido à incorporação de um sistema de inteligência artificial operando em conjunto com o sistema de reconhecimento de fala, tais sistemas passam a se tornar mais eficientes. A inteligência artificial faz a interação com o usuário, apresentando perguntas associadas às respostas ou até a algumas palavras pronunciadas pelo usuário. Assim, o sistema vai dirigindo o diálogo até culminar na reserva do vôo.

1.2 Visão Geral do Trabalho

Dentro desse contexto, buscou-se o desenvolvimento de um sistema de reconhecimento de fala consistente, com alta taxa de acerto, voltado para uma aplicação prática. A aplicação de reconhecimento de dígitos conectados é a opção ideal para se iniciar estudos dessa natureza, pois permite desenvolver o conhecimento necessário na busca de aplicações mais complexas, como o reconhecimento de fala contínua.

O projeto de um sistema de reconhecimento de fala inclui o estudo e aplicação de inúmeras técnicas de processamento de sinais de fala, como também a utilização de uma modelagem estatística através de modelos ocultos de Markov.

No capítulo 2 são descritas as principais características de um sistema de reconhecimento de fala, enfatizando os módulos do sistema que foram desenvolvidos nesse trabalho.

O capítulo 3 apresenta os algoritmos utilizados no processamento dos modelos ocultos de Markov. Algumas peculiaridades em relação à implementação dos algoritmos e ao custo computacional destes são também discutidas.

No capítulo 4 é descrito o sistema proposto neste trabalho. O desempenho do sistema é testado com duas bases de sinais: uma desenvolvida por pesquisadores do Laboratório de Processamento Digital de Fala [6][21] e a TI *Digits* [10]. Os testes, apesar de serem realizados em ambiente de laboratório, atestam a consistência do sistema implementado em tempo real.

No capítulo 5 é realizada a implementação em tempo real do sistema. São discutidas e apresentadas as adaptações necessárias do sistema de reconhecimento para torná-lo capaz de interagir com o usuário.

O capítulo 6 contém as conclusões do trabalho e as propostas de trabalhos futuros.

O desenvolvimento deste trabalho contou com o suporte de outros trabalhos já realizados por Meloni e Ynoguti [12][21]. As principais ferramentas de processamentos de sinais e os algoritmos *forward*, *backward* e Baum-Welch foram desenvolvidos em [12] e o quantizador vetorial é o mesmo utilizado em [21]. A tarefa de programação foi a que demandou mais tempo, e foi iniciada de posse de dois programas computacionais em linguagem Visual C++ quase acabados: um que realiza a extração de parâmetros do sinal de fala e outro que realiza a quantização vetorial. Esses programas foram ampliados, passando a processar as derivadas primeira e segunda dos parâmetros extraídos do sinal de fala (mais detalhes no capítulo 2). Um programa adicional foi criado, englobando procedimentos de treinamento dos modelos ocultos de Markov discretos para os casos de palavras isoladas e palavras conectadas, como também algoritmos de decodificação voltados para o reconhecimento de palavras isoladas e conectadas [18][19].

Uma das contribuições deste trabalho foi explorar uma metodologia de treinamento, já proposta na literatura [18], utilizando modelos ocultos de Markov discretos. Porém, sua contribuição mais importante está no desenvolvimento de uma aplicação de reconhecimento de fala em tempo real, abordando as particularidades de sua implementação e operação.

Capítulo 2
ALGUMAS TÉCNICAS DE RECONHECIMENTO DE FALA

2.1 Introdução

Pode-se descrever um sistema reconhecimento de fala, fazendo-se um paralelo com o sistema auditivo humano.

Considere um sinal de fala pronunciado por um locutor qualquer na entrada do sistema, como mostrado na Figura 2.1.

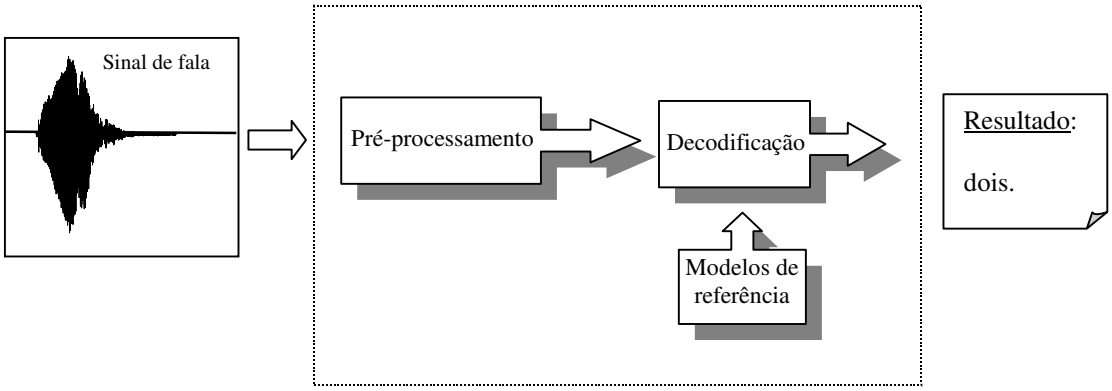

Figura 2.1 Sistema de reconhecimento de fala

O sinal é tratado através de ferramentas de processamento de sinais, que muitas vezes se assemelham às características do aparelho auditivo humano. No estado da arte em reconhecimento de fala, o processo complexo de decodificação do sinal realizado pela mente

humana é feito por modelos estatísticos simples. Para que uma decisão seja tomada, é necessário um conhecimento a priori, representado na Figura 2.1 por modelos de referência que são obtidos a partir de bases de sinais. Os modelos de referência representam o léxico ou vocabulário. Portanto, são selecionados vários exemplos de frases pronunciadas por inúmeros locutores formando uma base de dados. O conteúdo das frases depende do contexto em que se está trabalhando. A abordagem desse trabalho é voltada para uma aplicação de dígitos conectados e, assim, o vocabulário é restrito aos dígitos zero, um, dois, três, quatro, cinco, seis, sete, oito, nove e meia.

2.2 Características do Sistema de Reconhecimento de Fala

A Figura 2.1 divide o sistema de reconhecimento de fala em duas partes: pré-processamento e decodificação. O pré-processamento tem como principal característica converter o sinal de fala em parâmetros. Esses parâmetros podem ou não ser quantizados. Os parâmetros são, então, submetidos a técnicas de reconhecimento de padrões (decodificação), tais como redes neurais, modelos ocultos de Markov ou modelos híbridos. Este último inclui as duas primeiras técnicas.

Os parâmetros resultantes do pré-processamento influenciarão nas características do sistema, no que diz respeito às etapas de processamento posteriores à extração de parâmetros. Os sistemas contínuos são modelados por modelos ocultos de Markov contínuos, através de funções de densidade de probabilidade. A função densidade de probabilidade gaussiana é freqüentemente adotada para os HMM's contínuos. Melhores resultados são obtidos quando se trabalha com uma combinação de várias funções gaussianas, também conhecidas como misturas [17]. Os sistemas discretos são conhecidos por possuir uma etapa adicional conhecida como quantização vetorial, que é aplicada depois da etapa de extração de parâmetros. Os parâmetros quantizados são modelados por modelos ocultos de Markov discretos.

Neste trabalho, desenvolveu-se um sistema de reconhecimento de fala com as seguintes etapas de processamento do sinal de fala: análise espectral, quantização vetorial e modelagem estatística via modelos ocultos de Markov discretos. A descrição completa de cada uma dessas etapas é realizada a seguir.

2.2.1 Análise Espectral

Várias ferramentas de processamento de sinais foram propostas para a aplicação de reconhecimento de fala. Em [11] são comparadas algumas dessas ferramentas em função do desempenho do sistema de reconhecimento. Neste trabalho, optou-se por utilizar as configurações testadas em [11] que ofereceram melhor desempenho ao sistema de reconhecimento de fala.

A análise espectral empregada envolve os seguintes processamentos sobre o sinal de fala: filtragem passa-banda, filtragem de pré-ênfase, aplicação de janelas de *Hamming* superpostas e extração de parâmetros mel cepstrais, como mostrado na Figura 2.2.

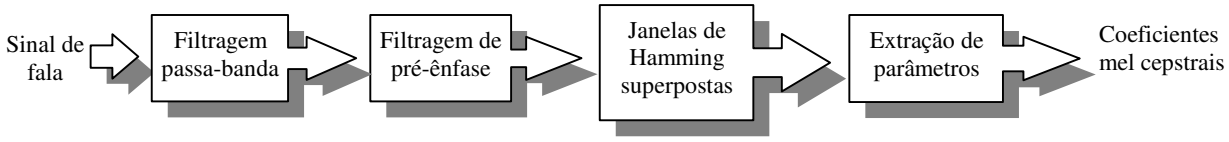

Figura 2.2 Análise Espectral

O filtro passa-banda possui freqüências de corte em 300 Hz e 3400 Hz. Através dele, simula-se as características em freqüência dos filtros empregados nos conversores analógico-digital em telefonia, que seguem o padrão da ITU (*International Telecommunications Union*). A função de transferência do filtro passa-banda é dada pela equação (2.1), os coeficientes dos filtros são apresentados na Tabela 2.1, e a resposta em freqüência do filtro é mostrada na Figura 2.3.

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_7 z^{-7}}{1 + a_1 z^{-1} + \ldots + a_7 z^{-7}} \tag{2.1}$$

Tabela 2.1 Coeficientes do filtro passa-banda

Coeficientes do filtro passa-banda			
a_0	1.00000000	b_0	0.00480775
a_1	-5.55209501	b_1	-0.00144185
a_2	13.64182831	b_2	-0.01105735
a_3	-19.23476132	b_3	0.00769144
a_4	16.79035417	b_4	0.00769144
a_5	-9.06253725	b_5	-0.01105735
a_6	2.79793502	b_6	-0.00144185
a_7	-0.38069353	b_7	0.00480775

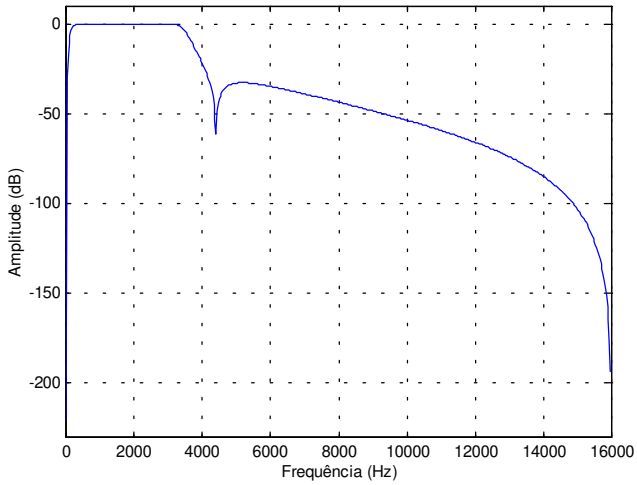

Figura 2.3 Resposta em frequência do filtro passa-banda.

A filtragem de pré-ênfase tem como função compensar a atenuação nas altas freqüências do sinal de fala pelo processo de produção da fala (espectro glotal), tornando o seu espectro de freqüência mais plano [15]. A resposta em freqüência de um filtro de pré-ênfase é mostrada na Figura 2.4.

A função de transferência do filtro de pré-ênfase é dada pela equação abaixo:

$$H(z) = 1 - \mu z^{-1}, \tag{2.2}$$

onde neste trabalho μ foi feito igual a 0.95

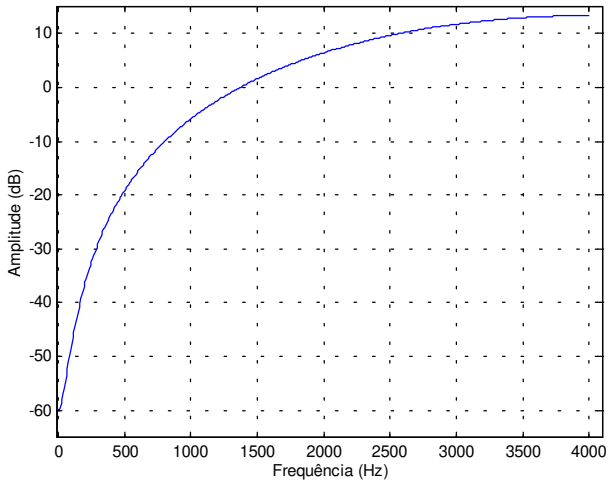

Figura 2.4 Resposta em frequência do filtro de pré-ênfase para $\mu = 0.95$.

O sinal de fala é dividido em quadros, cobrindo períodos típicos de 10 ms. Dentro deste intervalo, pode-se considerar o sinal de fala quase-estacionário. A divisão em quadros é feita através das janelas de *Hamming*, que são apresentadas na Figura 2.5. Essas janelas possuem períodos um pouco superiores ao do quadro, realçando a sua parte central. A janela de *Hamming* é definida pela seguinte equação:

$$h(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \le n \le N-1 \tag{2.3}$$

onde n é o índice da amostra, e N é o número total de amostras da janela de *Hamming*.

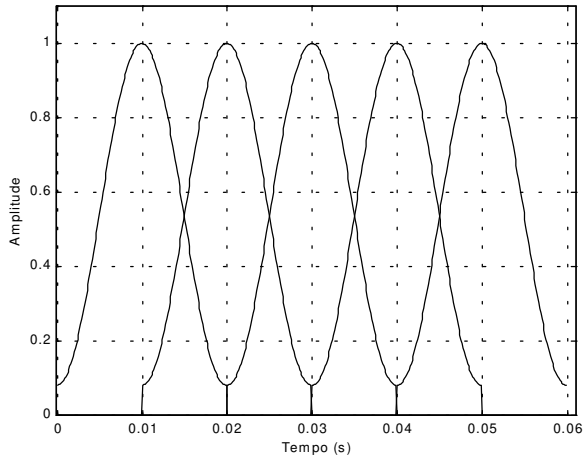

Figura 2.5 Janelas de *Hamming* de 20 ms com superposição de 50 %.

Os coeficientes mel cepstrais são obtidos a partir de cada janela do sinal, depois de realizados os seguintes processamentos:

• Aplicação do banco de filtros triangulares em escala mel e cálculo do logaritmo da energia de saída de cada filtro. A aplicação do logaritmo é necessária para a obtenção do cepstro. São utilizados geralmente 20 filtros de formato triangular, como mostrado na Figura 2.6. O espaçamento e a largura de faixa dos filtros usados são os tabelados em [11];

• Cálculo da transformada discreta inversa do co-seno (*DCT*) do vetor do logaritmo da energia de saída do banco de filtros através da equação

$$c(n) = \sum_{k=1}^{M} \left(\log_{10} X(k)\right) \cos\left(\frac{n(k-0.5)\pi}{M}\right), \qquad 1 \le n \le N \tag{2.4}$$

onde n – índice dos coeficientes mel cepstrais;

N – número total de coeficientes mel cepstrais;

k - índice do filtro

M – número total de filtros

$X(k)$ – energia de saída do filtro k.

O coeficiente $c(0)$, que é função da soma das energias de todos os filtros, não é utilizado [15].

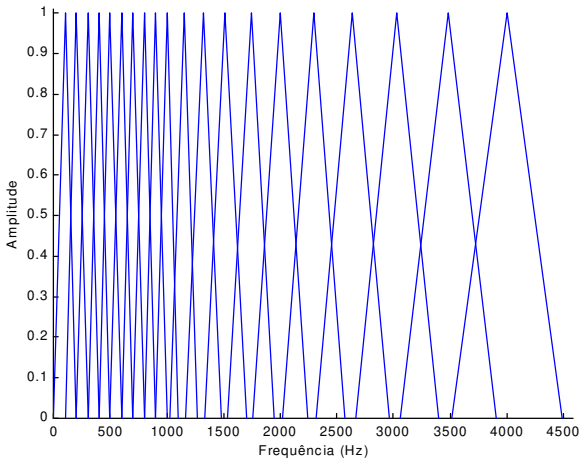

Figura 2.6 Banco de 20 filtros triangulares na escala mel.

Quanto mais detalhada for a extração das características do sinal de fala, melhor será o resultado do sistema de reconhecimento. Resultados publicados na literatura mostram que o emprego de energia e das derivadas primeira e segunda desses parâmetros (mel cepstrais e energia) melhoram sobremaneira a taxa de acerto em reconhecimento de fala [11][14]. A primeira e segunda derivadas dos coeficientes cepstrais são obtidas pelas seguintes equações:

$$Dc_i^1(n) = \sum_{k=-K}^{K} \frac{k\, c_{i-k}(n)}{2K+1} \tag{2.5}$$

$$Dc_i^2(n) = \sum_{k=-K}^{K} \frac{k\, Dc_{i-k}^1(n)}{2K+1} \tag{2.6}$$

onde i – índice do quadro do sinal;

n – índice do coeficiente mel cepstral;

K – número de quadros utilizados no cálculo das derivadas;

O cálculo da primeira e segunda derivada da energia é feito pelas próprias equações (2.5) e (2.6), substituindo o vetor de coeficientes $c(n)$ pela energia do quadro.

2.2.2 Quantização Vetorial

Sistemas discretos requerem a representação no domínio discreto do conjunto de coeficientes cepstrais. Uma forma eficiente de se discretizar o conjunto de parâmetros é através da quantização vetorial. A partir da base de dados de treinamento, define-se um dicionário de códigos ou *codebook* segundo certo critério de otimização. O *codebook* é acessado a cada tarefa de quantização de um conjunto de parâmetros. A determinação do vetor mais adequado é resultado de uma busca exaustiva da menor distância entre o vetor de parâmetros e o vetor do *codebook*. Várias medidas de distorção podem ser utilizadas, porém, a mais comum é a medida de distorção euclidiana [19].

A geração dos *codebooks* depende de um processo de otimização, o qual faz o levantamento dos vetores que melhor representam todo um conjunto de parâmetros. O algoritmo utilizado no treinamento é o LBG [7]. Utiliza-se um *codebook* para cada tipo de parâmetro extraído do sinal.

Aplicando-se a quantização vetorial sobre um conjunto de parâmetros composto pelos coeficientes cepstrais e energia, além das suas primeira e segunda derivadas, o resultado será um vetor de 6 símbolos. Cada símbolo representará o índice do vetor resultante da quantização. A Figura 2.7 ilustra melhor o processo de quantização.

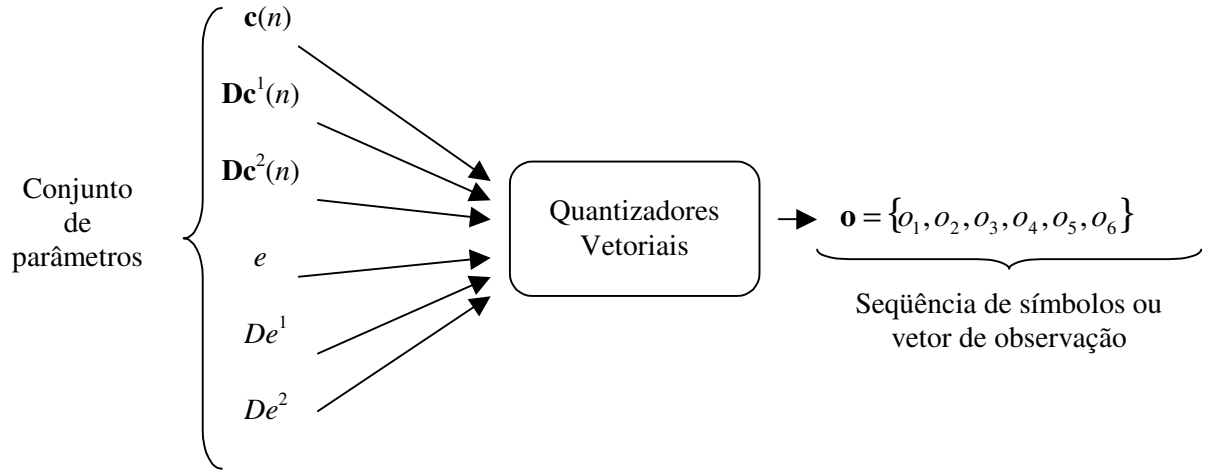

Figura 2.7 Quantização vetorial.

Apesar da energia ser um simples escalar, utilizou-se o procedimento de quantização vetorial. Neste caso, os vetores do *codebook* possuem uma dimensão apenas.

2.2.3 Modelos Ocultos de Markov – HMM (do inglês, Hidden Markov Models)

Os modelos ocultos de Markov são uma teoria matemática desenvolvida no final dos anos 60 [3] e utilizada com muita propriedade no reconhecimento de fala em meados dos anos 70 [2]. Desde então, essa teoria vem sendo utilizada em várias aplicações, inclusive no reconhecimento de fala. Isso justifica o destaque dado a ela neste capítulo.

2.2.3.1 Elementos que compõem o HMM

Suponha uma máquina de estados com as características apresentadas na Figura 2.8.

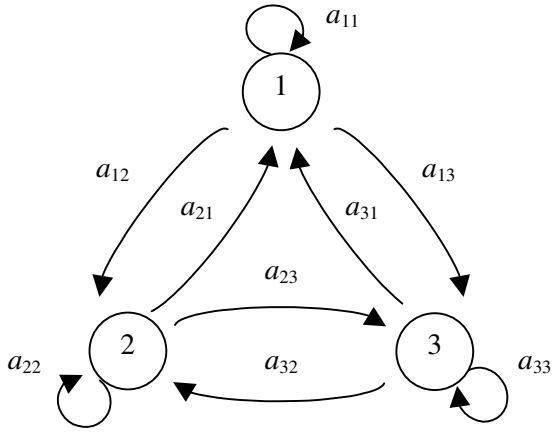

Figura 2.8 Máquina de estados.

Os estados da estrutura da Figura 2.8 podem representar condições climáticas hipotéticas [17] . Por exemplo, o estado 1 indica *dia ensolarado*, o estado 2 indica *dia nublado* e o estado 3 indica *dia chuvoso*. Os coeficientes a_{ij} representam a probabilidade de transição do estado *i* para o estado *j*. Estas probabilidades definem uma matriz de transição de estados **A**.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.4 & 0.4 \\ 0.1 & 0.5 & 0.4 \\ 0.4 & 0.3 & 0.3 \end{bmatrix} \tag{2.7}$$

Na equação (2.7) são apresentados alguns valores numéricos fictícios.

Modelando-se as probabilidades de transição a_{ij} por cadeias de Markov de primeira ordem, essas passam a ser dependentes apenas do estado anterior, isto é,

$$a_{ij} = P[q_{t+1} = j \mid q_t = i], \qquad 1 \le i \le 3 \text{ e } 1 \le j \le 3 \tag{2.8}$$

onde q_t é o estado no instante t.

Outra característica presente na máquina de estados é a probabilidade do estado inicial, que é representada pelo vetor $\boldsymbol{\pi}$ da equação (2.9). Os coeficientes π_i são as probabilidades de se iniciar no estado *i*, onde $1 \le i \le 3$.

$$\boldsymbol{\pi} = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0 \\ 0 \end{bmatrix} \tag{2.9}$$

A máquina de estados pode ser utilizada para modelar inúmeros problemas, e cada estado corresponde a um determinado evento. O exemplo da Figura 2.8 representa a modelagem das

possíveis condições climáticas do tempo num determinada região do planeta. Neste caso, os valores determinísticos das probabilidades da máquina de estado são obtidos a partir de informações de condições climáticas anteriores.

Várias informações podem ser obtidas da máquina de estados. Supondo, por exemplo, o clima no dia d_0 = *ensolarado*, a probabilidade de o clima ser **o** = {*ensolarado*, *chuvoso*, *chuvoso*} nos três dias seguintes é

$$P(\mathbf{o} \mid Modelo) = \pi_1 a_{11} a_{13} a_{33} = 0.024 \tag{2.10}$$

Aumentando a complexidade da máquina de estado, pode-se considerar desconhecidos os eventos associados a cada estado. Isto corresponde a dizer que o *estado* 1 não mais representa apenas a condição climática *ensolarado*, o mesmo ocorrendo para os demais estados. Nesta situação, deseja-se criar uma nova relação entre os possíveis eventos e os estados, para que seja possível determinar a seqüência de estados que melhor represente as condições climáticas do exemplo anterior. Isto é possível através da inserção de uma informação adicional que representa a probabilidade de emissão.

A Figura 2.9 apresenta a mesma máquina de estado da Figura 2.8 acrescida da probabilidade de emissão. Esta extensão das cadeias de Markov é que denominamos modelos ocultos de Markov (HMM).

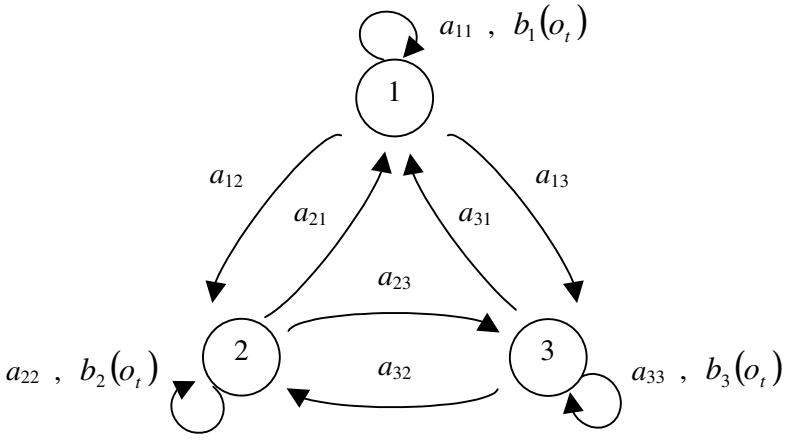

Figura 2.9 Máquina de estados acrescida da matriz probabilidade de emissão **B**.

Os coeficientes $b_j(o_t)$ representam a probabilidade de ocorrência do evento o_t no estado j, tal que $o_t \in$ { e = ensolarado, n = nublado, c = chuvoso }. A probabilidade de emissão dependerá dos eventos que serão emitidos a cada instante de tempo. A equação abaixo ilustra os valores dos coeficientes da matriz probabilidade de emissão **B**.

$$\mathbf{B} = \begin{bmatrix} b_1(e) & b_2(e) & b_3(e) \\ b_1(n) & b_2(n) & b_3(n) \\ b_1(c) & b_2(c) & b_3(c) \end{bmatrix} = \begin{bmatrix} 0.5 & 0.2 & 0.2 \\ 0.1 & 0.5 & 0.4 \\ 0.4 & 0.3 & 0.4 \end{bmatrix} \tag{2.11}$$

Refazendo-se o exemplo anterior com a inserção da probabilidade de emissão e tomando-se a seqüência de estados que maximiza o resultado, vem

$$\max_{q_1,q_2,q_3} P(q_1 q_2 q_3, \mathbf{o} \mid Modelo) = \pi_1 \, a_{11} \, b_1(o_1) a_{13} \, b_3(o_2) a_{33} \, b_3(o_3) = 0.00192\,, \tag{2.12}$$

onde (q_1 q_2 q_3) representam uma seqüência de estados qualquer e **o** = {e, c, c} é o vetor de eventos ocorridos.

É importante notar na equação (2.12) que as probabilidades de transição e de emissão de cada símbolo são consideradas independentes, e, por esse motivo, elas são simplesmente multiplicadas. Essa consideração é umas das limitações dos HMM's [17].

A mudança realizada entre a modelagem da Figura 2.8 e a Figura 2.9 provocou um aumento no grau de liberdade da máquina de estado, que está associado às matrizes de probabilidade. Assim, a capacidade de se modelar uma seqüência de eventos ficou maior. Isso é útil quando tratamos de problemas que apresentam uma grande quantidade de eventos possíveis.

Até esse ponto, foram definidos dois conceitos muito importantes: as matrizes de probabilidade de transição **A** e probabilidade de emissão **B**. Esses matrizes são os principais elementos que compõem os HMM's. A notação mais comum utilizada na representação dos HMM's é a seguinte:

$$\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}), \tag{2.13}$$

onde λ é o HMM e **A**, **B** e $\boldsymbol{\pi}$ são respectivamente as matrizes de probabilidades de transição, emissão e o vetor de estado inicial.

A probabilidade de emissão está sendo apresentada na sua forma discreta, pois os eventos são discretos. Entretanto, em reconhecimento de fala, os parâmetros extraídos do sinal de fala são contínuos. Neste caso, existem duas alternativas de modelar eventos de natureza contínua. A primeira utiliza a probabilidade de emissão discreta mostrada até aqui, mas é necessária a quantização vetorial do conjunto de parâmetros. A segunda possibilidade consiste em se utilizar uma modelagem por probabilidades de emissão contínuas. Geralmente emprega-se uma ou mais pdf's gaussianas podem modelar cada parâmetro. Portanto, cada pdf é identificada por um valor de média e variância [17]. Os elementos de cada vetor de parâmetros são considerados independentes entre si e, assim, a probabilidade de emissão em cada estado *j* é dada por um vetor de médias $\boldsymbol{\mu}_j$ e uma matriz de covariâncias diagonal $\mathbf{U}_j$, conforme a equação abaixo.

$$b_j(\mathbf{o}_t) = \frac{1}{(2\pi)^{\frac{D}{2}} \left|\mathbf{U}_j\right|^{\frac{1}{2}}} \exp\left(-\frac{(\mathbf{o}_t - \boldsymbol{\mu}_j)\mathbf{U}_j^{-1}(\mathbf{o}_t - \boldsymbol{\mu}_j)^T}{2} \right), \tag{2.14}$$

onde *D* é a dimensão do vetor $\mathbf{o}_t$, que é a ordem do vetor de parâmetros gerado a cada quadro *t* do sinal de fala.

Neste trabalho, é utilizada probabilidade de emissão discreta e, conseqüentemente, os modelos ocultos de Markov são discretos.

2.2.3.2 Problemas envolvendo os modelos ocultos de Markov

Até esse ponto, foram apresentadas as principais características dos modelos ocultos de Markov. Entretanto, para que sua utilização seja possível, é necessário que se discutam alguns de seus problemas.

Podem-se destacar três problemas em se tratando de modelos ocultos de Markov [17]. Um deles, já identificado no exemplo da previsão do tempo, consiste na determinação da seqüência de estados associada ao conjunto de parâmetros obtidos do pré-processamento de uma palavra qualquer. Os parâmetros são identificados como símbolos e cada símbolo terá seu estado correspondente. Considerando conhecidas a priori as probabilidades de emissão e de transição, basta calcular todas as combinações das seqüências de estados possíveis, que a seqüência escolhida será a que possuir maior probabilidade.

A solução para esse problema é uma tarefa que requer um grande esforço computacional. Entretanto, algoritmos otimizados executam tal tarefa levando em consideração as características do modelo. O algoritmo que se destaca na realização dessa tarefa é o algoritmo de Viterbi [17].

Outro problema a ser solucionado consiste em se determinar a probabilidade de uma seqüência de observação para um dado modelo $P(\mathbf{o} \mid \lambda)$. Para essa tarefa, utiliza-se o algoritmo *forward* [17]. Em situações em que são usados vários modelos, utiliza-se essa probabilidade para determinar o modelo mais provável. No exemplo da previsão do tempo, isso ajudaria a determinar a região ou o estado que possui características climáticas mais próximas da apresentada pela seqüência de observação e, no caso de reconhecimento de fala, a seqüência de fones mais provável. O algoritmo de Viterbi também pode ser utilizado para determinar o modelo mais provável. Entretanto, a probabilidade obtida pelo algoritmo de Viterbi está associada apenas à seqüência de estados ótima.

O terceiro e último problema consiste em ajustar os parâmetros do modelo, até aqui identificados por suas probabilidades de transição e emissão, para a nova seqüência de observação. Na verdade o ajuste dos parâmetros visa maximizar a probabilidade do modelo para a seqüência que ainda não fazia parte da informação a *priori* do sistema. Esse problema pode ser identificado como a etapa de treinamento dos HMM's. No caso do reconhecimento de fala, os HMM's são otimizados a partir da maximização da probabilidade das seqüências de observação. A tarefa de treinamento é a que demanda maior custo computacional, pois depende de algoritmos de otimização. O treinamento normalmente é demorado devido ao conjunto de treinamento ser muito grande. Mais detalhes de treinamento serão apresentados no capítulo de simulação do sistema de reconhecimento.

Dois algoritmos são utilizados na solução do terceiro problema: Baum-Welch e Viterbi [17][19]. Ambos realizam uma tarefa equivalente a uma contagem de ocorrências das observações em cada estado do modelo. O algoritmo de Viterbi representa uma alternativa de baixo custo computacional para solução de problema, pois o algoritmo determina uma seqüência de observação ótima e a partir dela efetua a contagem das ocorrências das observações. Já o algoritmo de Baum-Welch incorpora os algoritmos *forward* e *backward*. O algoritmo *backward* possui as mesmas características do *forward*, distinguindo-se somente do ponto de partida do algoritmo, que é o último estado do modelo.

Os modelos ocultos de Markov podem representar palavras ou subunidades fonéticas, e essa representação é resultado do treinamento de um conjunto representativo de palavras ou subunidades fonéticas obtidas a partir de uma base de dados. A escolha entre modelos de palavras e modelos de subunidades fonéticas dependerá do número de palavras presentes no vocabulário do sistema.

Aplicações que utilizam vocabulários com um número de palavras superior a uma centena requerem modelos HMM's de subunidades fonéticas, tais como fonemas, sílabas, etc. Isso reduzirá o número de HMM's necessários para o reconhecimento das palavras do vocabulário. Vale comentar que a combinação desses fones irá gerar todas as palavras do vocabulário. Entretanto, para sistemas de reconhecimento de dígitos conectados, cujo vocabulário é da ordem de uma dezena de palavras, a alternativa de se trabalhar com modelos de palavras é mais atraente.

A concepção de um sistema de reconhecimento de fala e, portanto, a escolha dos diversos algoritmos depende da aplicação: palavras isoladas, palavras conectadas e fala contínua. Este é o grande desafio do pesquisador da área de reconhecimento de fala: qual é a melhor escolha para o sistema almejado entre as diversas possibilidades a sua mão.

No que diz respeito ao reconhecimento de palavras conectadas, que é o objeto desta pesquisa, dois algoritmos se destacam: *Level Building* e *One Step* [19][9]. Detalhes referentes a todos os algoritmos mencionados até aqui são apresentados no próximo capítulo.

2.2.3.3 Estrutura *left-right*

Uma estrutura para os modelos ocultos de Markov, aplicada no reconhecimento de fala, é a estrutura *left-right* da Figura 2.10.

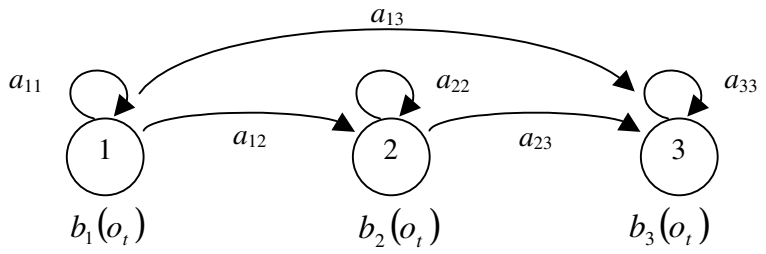

Figura 2.10 Estrutura *left-right*.

A estrutura *left-right* é mais adequada para aplicações de reconhecimento de fala por possuir somente transições da esquerda para direita e, dessa forma, modela as propriedades

progressivas dos sinais de fala. Para essa estrutura, a matriz de probabilidade de transição de estados **A** terá alguns coeficientes nulos e, portanto, poderá ser simplificada conforme exemplificação da equação (2.15).

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix} \tag{2.15}$$

2.2.4 Modelos de Duração

Os modelos ocultos de Markov não incorporam a informação de duração de forma consistente. A duração expressa em quadros pode ser levada em consideração tanto quanto a permanência em cada um dos estados do modelo como também pela soma de todas as transições entre estados. A primeira é conhecida como duração de estados e a segunda como duração da unidade fonética.

A duração de estados é muito discutida, pois sua utilização visa corrigir uma inconsistência inerente ao modelo. Essa inconsistência é a característica exponencial da probabilidade de duração de estado, que é dada por [17]

$$p_i(d) = (a_{ii})^{d-1} a_{ij}\,, \tag{2.16}$$

onde *d* é a duração em número de quadros.

A probabilidade de duração de estado em muitos sinais físicos, inclusive nos sinais de fala, não é modelada por uma exponencial. A alternativa consiste em explicitar a probabilidade de duração pela construção de histogramas no procedimento de treinamento [18]. O histograma conterá a probabilidade de duração em cada estado em função da permanência nele. Vários procedimentos de geração dos histogramas são propostos na literatura [18], porém nenhum deles mostrou melhora nos resultados obtidos em nossos experimentos.

A duração de palavras é outra alternativa de melhorar o desempenho dos modelos ocultos de Markov. A modelagem é feita através da parametrização da duração por uma função densidade de probabilidade gaussiana caracterizada pela equação abaixo.

$$p_v(d) = \frac{1}{\sqrt{2\pi\sigma_v^2}} \exp\left(-\frac{(d-\mu_v)^2}{2\sigma_v^2} \right), \tag{2.17}$$

onde μ_v é a duração média da palavra *v* e σ_v^2 é a variância.

O modelo de duração de palavras penaliza as probabilidades obtidas pelos algoritmos de reconhecimento de fala.

Capítulo 3
ALGORITMOS DE RECONHECIMENTO DE FALA UTILIZADOS

3.1 Introdução

No capítulo 2 foram apresentados os modelos ocultos de Markov bem como mencionados alguns algoritmos que realizam o seu processamento.

Os algoritmos mencionados, Viterbi, *forward*, *backward*, Baum-Welch, *Level Building* e *One Step*, possuem peculiaridades, e sua utilização depende da tarefa a qual se deseja realizar.

Assim, uma descrição mais detalhada desses algoritmos, enfatizando os problemas que advêm de sua implementação, torna-se necessária. Esse capítulo é dedicado à apresentação dos algoritmos.

3.2 Algoritmos aplicados no processamento dos HMM's

Os HMM's apresentam problemas de interesse, cuja formulação pode ser resumida em [17]:

1) Como calcular a probabilidade de uma seqüência de observação **o** dado o HMM λ, ou $P(\mathbf{o} \mid \lambda)$?

2) Qual é a seqüência de estados ótima, se conhecidos a seqüência de observação e o HMM?

3) Como re-estimar os parâmetros $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$, maximizando-se $P(\mathbf{o} \mid \lambda)$?

A seguir são revisados os algoritmos que eficientemente resolvem os problemas citados.

3.2.1 Algoritmo *Forward*

O algoritmo *forward*[1] permite a solução do primeiro problema dos HMM's. Ele possui uma complexidade superior, se comparado com o algoritmo de Viterbi. Ao invés de realizar o cálculo de probabilidade em função apenas da seqüência de estados ótima, o algoritmo *forward* realiza o cálculo de maneira direta, levando em consideração todas as seqüências de estados possíveis. Computacionalmente essa operação é custosa, justificando a utilização de um procedimento cujo processamento é eficiente.

O algoritmo emprega a seguinte variável:

$$\alpha_t(i) = P(o_1 o_2 \ldots o_t, q_t = s_i \mid \lambda) \tag{3.1}$$

onde α_t é a probabilidade parcial da seqüência de observação $\{o_1\ o_2\ \ldots\ o_t\}$ até o instante t e do estado s_i no instante t, dado o modelo λ.

O cálculo eficiente de α_t é realizado pelo algoritmo *forward*, apresentado abaixo:

1. *Iniciação*

$$\alpha_1(i) = \pi_i b_i(o_1), \qquad 1 \le i \le N \tag{3.2}$$

2. *Recursão*

Para $1 \le t \le T-1$ e $1 \le j \le N$, **faça**:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(o_{t+1}) \tag{3.3}$$

3. *Finalização*

$$P(\mathbf{o} \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{3.4}$$

O resultado do algoritmo *forward* consiste na probabilidade da seqüência de observação dado o modelo HMM λ; ou seja $P(\mathbf{o} \mid \lambda)$.

[1] As variáveis definidas seguem a mesma notação de [17]. Isso vale também para os algoritmos *backward*, Viterbi e Baum-Welch.

Considerando a estrutura *left-right*, que é um tipo de topologia de HMM utilizada no reconhecimento de fala, pode-se fazer algumas modificações no algoritmo, o que o tornará mais simplificado e reduzirá o custo computacional:

1. A probabilidade de estado inicial passa a ser dada por:

$$\pi_i = \begin{cases} 1 & i = 1 \\ 0 & i \neq 1 \end{cases} \tag{3.5}$$

2. A probabilidade de transição apresenta a seguinte propriedade:

$$a_{ij} = 0 \text{ para } i > j \text{ e } j > \Delta + i \tag{3.6}$$

onde Δ é o salto máximo entre estados (índice do estado final menos o índice do estado inicial).

3. Quando se trabalha com HMM discreto, a probabilidade de emissão *b* é representada por uma matriz de probabilidades, assim como a probabilidade de transição *a*. Portanto, calculando-se o logaritmo dos valores de probabilidade previamente, o algoritmo apresentará somente operações de soma e comparação.

Implementando as simplificações acima, o algoritmo *forward* fica:

1. *Iniciação*

$$\alpha_1(1) = \log(b_1(o_1)) \tag{3.7}$$

2. *Recursão*

Para $1 \leq t \leq T-1$ e $1 \leq j \leq N$, **faça**:

$$\alpha_{t+1}(j) = \sum_{i=j}^{(j+\Delta)\leq N} [\alpha_t(i) + \log(a_{ij})] + \log(b_j(o_{t+1})) \tag{3.8}$$

3. *Finalização*

$$P(\mathbf{o} \mid \lambda) = \sum_{i=1}^{N} \exp(\alpha_T(i)) \tag{3.9}$$

Vale ressaltar que tais modificações dependem da estrutura utilizada ser de topologia *left-right*. Além disso, o cálculo do logaritmo das probabilidades de transição e emissão deve ser efetuado antes do processamento do algoritmo, para que este opere corretamente.

3.2.2 Algoritmo *Backward*

Definindo a variável β_t como sendo

$$\beta_t(i) = P(o_{t+1}o_{t+2}\ldots o_T \mid q_t = s_i, \lambda) \tag{3.10}$$

onde β_t é a probabilidade parcial do final da seqüência de observação $\{o_{t+1}o_{t+2}\ldots o_T\}$, dado o estado s_i no instante t e o modelo λ.

O cálculo eficiente de β_t pelo algoritmo *backward* [17] fica:

1. *Iniciação*

$$\beta_T(i) = 1, \qquad 1 \le i \le N \tag{3.11}$$

2. *Recursão*

Para $T-1 \ge t \ge 1$ e $1 \le j \le N$, **faça**:

$$\beta_t(j) = \sum_{i=1}^{N} a_{ij} b_i(o_{t+1}) \beta_{t+1}(i) \tag{3.12}$$

3. *Finalização*

$$P(\mathbf{o} \mid \lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i) \tag{3.13}$$

As características deste algoritmo são semelhantes às do algoritmo *forward*, diferenciando-se apenas na recursão que opera no sentido inverso do índice temporal. Esse algoritmo se faz útil para a implementação do algoritmo Baum-Welch.

Efetuando as mesmas simplificações vistas para o algoritmo forward, tem-se

1. *Iniciação*

$$\beta_T(i) = 0, \qquad 1 \le i \le N \tag{3.14}$$

2. *Recursão*

Para $T-1 \ge t \ge 1$ e $1 \le j \le N$, **faça**:

$$\beta_t(j) = \sum_{i=j}^{(j+\Delta)\le N} \left[\log(a_{ij}) + \log(b_i(o_{t+1})) + \log(\beta_{t+1}(i))\right] \tag{3.15}$$

3. *Finalização*

$$P(\mathbf{o} \mid \lambda) = \sum_{i=1}^{N} \exp\left[\log(\pi_i) + \log(b_i(o_1)) + \beta_1(i)\right] \tag{3.16}$$

3.2.3 Algoritmo de Viterbi

O algoritmo emprega a seguinte grandeza

$$\delta_t(i) = \max_{q_1 q_2 \ldots q_{t-1}} P(q_1 q_2 \ldots q_{t-1}, q_t = s_i, o_1 o_2 \ldots o_t \mid \lambda) \tag{3.17}$$

onde i – índice do estado, podendo variar de 1 a N.

t – instante de tempo, variando de 1 a T.

q_t – representa o estado e $\{q_1\ q_2\ \ldots\ q_t\}$ é definido como uma seqüência de estados qualquer obtida até o instante t.

o_t – representa a observação dada no instante t e $\{o_1\ o_2\ \ldots\ o_t\}$ é o conjunto de observações até o instante t.

λ - modelo oculto de Markov

δ_t – maior probabilidade resultante de uma seqüência de estados associada ao conjunto de t observações, onde a última observação ocorre no estado s_t.

O algoritmo de Viterbi é apresentado abaixo:

1. *Iniciação*

$$\delta_1(i) = \pi_i b_i(o_1), \qquad 1 \le i \le N \tag{3.18}$$

$$\psi_1(i) = 0, \qquad 1 \le i \le N \tag{3.19}$$

onde π_i é a probabilidade do estado inicial

2. *Recursão*

Para $2 \le t \le T$ e $1 \le j \le N$, **faça**:

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(o_t) \tag{3.20}$$

$$\psi_t(j) = \arg \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] \tag{3.21}$$

3. *Finalização*

$$P^* = \max_{1 \le i \le N} \delta_T(i) \tag{3.22}$$

$$q_T^* = \arg \max_{1 \le i \le N} \delta_T(i) \tag{3.23}$$

4. *Busca pela seqüência de estados ótima – backtracking*

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots, 1 \tag{3.24}$$

A variável ψ_t é um vetor de estados, que armazena a seqüência de estados ótima obtida até o instante *t*. O valor de probabilidade da seqüência ótima é dado por P^* e o último estado da seqüência ótima é armazenado em q_T^*. O procedimento de *backtracking* inicia no estado q_T^* e termina no estado q_1^*, que é o primeiro estado da seqüência ótima.

As considerações feitas para o algoritmo de *forward*, quando aplicadas ao HMM de estrutura *left-right*, também servem para o algoritmo de Viterbi. Efetuando-se as alterações, o algoritmo de Viterbi modificado fica:

1. *Iniciação*

$$\delta_1(1) = \log(b_1(o_1)) \tag{3.25}$$

$$\psi_1(i) = 0, \qquad 1 \le i \le N \tag{3.26}$$

$$\delta_1(i) = -\inf, \quad 2 \le i \le N \tag{3.27}$$

onde -inf é o limite inferior da representação numérica que está sendo utilizada.

2. *Recursão*

Para $2 \le t \le T$ e $1 \le j \le N$, **faça**:

$$\delta_t(j) = \underset{\substack{i,t.q. \\ j \le i \\ i \le N \\ i < j-\Delta}}{máx}[\delta_{t-1}(i) + \log(a_{ij})] + \log(b_j(o_t)) \tag{3.28}$$

$$\psi_t(j) = \arg \underset{\substack{i,t.q. \\ j \le i \\ i \le N \\ i < j-\Delta}}{máx}[\delta_{t-1}(i) + \log(a_{ij})] \tag{3.29}$$

3. *Finalização*

$$P^* = \underset{1 \le i \le N}{máx}\, \delta_T(i) \tag{3.30}$$

$$q_T^* = \arg \underset{1 \le i \le N}{máx}\, \delta_T(i) \tag{3.31}$$

4. *Busca pela seqüência de estados ótima – backtracking*

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots, 1 \tag{3.32}$$

O custo computacional é bastante reduzido, com instruções de soma e comparação apenas.

O algoritmo acima se aplica a todos problemas dos HMM's levantados na seção 3.2. A determinação da seqüência de estados ótima e da medida de probabilidade de uma seqüência de observações e desta seqüência de estados dado o modelo são resolvidas diretamente. O problema de re-estimação dos parâmetros do HMM é solucionado a partir das informações de transição de estados obtidas da seqüência de estados ótima. Assim, associa-se a probabilidade de emissão $b_j(o_t)$ com o número de vezes que ocorreu a observação o_t no estado j dividido pelo número de

observações no estado *j*. Para a probabilidade de transição a_{ij}, será necessário observar as situações abaixo:

a_{ii} = número de vezes que ocorreu a transição do estado *i* para o estado *i* dividido pelo número de transições a partir do estado *i*;

a_{ij} = número de vezes que ocorreu a transição do estado *i* para o estado $j = i + \Delta$ dividido pelo número de transições no estado *i*, onde o inteiro $\Delta > 0$.

3.2.4 Algoritmo Baum-Welch

O algoritmo Baum-Welch é o mais indicado para a estimação dos parâmetros do HMM [17], que consiste em escolher um modelo λ que maximize a $P(\mathbf{o} \mid \lambda)$ (terceiro problema dos HMM's mencionado na seção 3.2).

As equações deste algoritmo podem ser apresentadas e melhor visualizadas em termos das variáveis α_t e β_t dos algoritmos *forward* e *backward*.

Para o HMM discreto, a re-estimação das matrizes de transição e emissão é dada por [17]:

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)} \tag{3.33}$$

$$\bar{b}_i(k) = \frac{\sum_{t=1}^{T} \alpha_t(i) \beta_t(i)}{\sum_{t=1}^{T} \alpha_t(i) \beta_t(i)} \tag{3.34}$$

onde $\bar{a}_{ij}$ - re-estimação da probabilidade de transição do estado *i* ao *j*;

$\bar{b}_i(k)$ - re-estimação da probabilidade de emissão do símbolo de índice *k* no estado *i*;

k – índice do símbolo, podendo variar de 1 a *K;*

V_k - símbolo cuja probabilidade de emissão está sendo re-estimada;

Devido às características da topologia *left-right*, o vetor de probabilidade do estado inicial π não é re-estimado.

Problemas de *underflow* e, conseqüentemente, divisão por zero podem ocorrer à medida que o instante de tempo *t* cresce, pois α_t aproxima-se de zero. Por isso, é razoável que se utilize o escalonamento das variáveis [17].

Aplicando o escalonamento no algoritmo *forward* [13], as equações ficam:

1. *Iniciação*

$$c_1 = \frac{1}{\sum_{i=1}^{N} \alpha_1(i)} \tag{3.35}$$

$$\tilde{\alpha}_1 = c_1 \alpha_1(i), \qquad 1 \le i \le N \tag{3.36}$$

2. *Recursão*

2.1 *Variáveis Forward*

Para $1 \le t \le T-1$, **faça**:

$$\hat{\alpha}_{t+1}(j) = \left[\sum_{i=1}^{N} \tilde{\alpha}_t(i) a_{ij} \right] b_j(o_{t+1}), \qquad 1 \le j \le N \tag{3.37}$$

$$c_{t+1} = \frac{1}{\sum_{i=1}^{N} \hat{\alpha}_{t+1}(i)} \tag{3.38}$$

$$\tilde{\alpha}_{t+1} = c_{t+1} \hat{\alpha}_{t+1}(i), \qquad 1 \le i \le N \tag{3.39}$$

onde c_t é o coeficiente de escalonamento.

2.2 *Variáveis Backward*

$$\tilde{\beta}_T(i) = c_T, \qquad 1 \le i \le N \tag{3.40}$$

Para $T-1 \ge t \ge 1$ e $1 \le j \le N$, **faça**:

$$\hat{\beta}_t(j) = \sum_{i=1}^{N} a_{ij} b_i(o_{t+1}) \tilde{\beta}_{t+1}(i) \tag{3.41}$$

$$\tilde{\beta}_t(j) = \hat{\beta}_t(j) c_t \tag{3.42}$$

3. *Re-estimação*

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \tilde{\alpha}_t(i) a_{ij} b_j(o_{t+1}) \tilde{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1} \tilde{\alpha}_t(i) \tilde{\beta}_t(i) / c_t} \tag{3.43}$$

$$\bar{b}_i(k) = \frac{\sum_{t=1}^{T} \tilde{\alpha}_t(i) \tilde{\beta}_t(i) / c_t}{\sum_{t=1}^{T} \tilde{\alpha}_t(i) \tilde{\beta}_t(i) / c_t} \tag{3.44}$$

Com a utilização do algoritmo *forward* escalonado, o cálculo da $P(\mathbf{o} \mid \lambda)$ tendo em vista as equações (3.9) e (3.38) será obtido a partir da equação:

$$\log(P(\mathrm{o} \mid \lambda)) = -\sum_{t=1}^{T} \log c_t \tag{3.45}$$

As variáveis α_t e β_t escalonadas passam a ser representadas por $\tilde{\alpha}_t$ e $\tilde{\beta}_t$ respectivamente, e $\hat{\alpha}_t$ e $\hat{\beta}_t$ são as variáveis atualizadas a partir de valores escalonados.

Em reconhecimento de fala, as probabilidades de transição e emissão são re-estimadas a partir de um conjunto de elocuções de treinamento, ou seja, utilizam-se várias seqüências de observação cada uma com comprimento T_r, onde r é o índice da elocução. Além disso, observa-se que as equações de re-estimação (3.43) e (3.44) consideram uma única elocução (vetor de observações) de comprimento T e, portanto, uma adaptação nessas equações se faz necessária para operar com uma base de sinais (observações). As equações de re-estimação do algoritmo Baum-Welch ficam:

$$\bar{a}_{ij} = \frac{\sum_{r=1}^{R} \sum_{t=1}^{T_r-1} \tilde{\alpha}_t^r(i) a_{ij} b_j(o_{t+1}^r) \tilde{\beta}_{t+1}^r(j)}{\sum_{r=1}^{R} \sum_{t=1}^{T_r-1} \tilde{\alpha}_t^r(i) \tilde{\beta}_t^r(i) / c_t^r} \tag{3.46}$$

$$\bar{b}_i(k) = \frac{\sum_{r=1}^{R} \sum_{\substack{t=1 \\ t.q.\ o_t=V_k}}^{T_r-1} \tilde{\alpha}_t^r(i) \tilde{\beta}_t^r(j) / c_t^r}{\sum_{r=1}^{R} \sum_{t=1}^{T_r-1} \tilde{\alpha}_t^r(i) \tilde{\beta}_t^r(i) / c_t^r} \tag{3.47}$$

onde R o número total de elocuções. Cada elocução tem um total de T_r quadros.

3.3 Algoritmos de Decodificação

Até aqui foram apresentados algoritmos que resolvem os principais problemas dos HMM's. Alguns deles, *forward* e Viterbi, são utilizados na tarefa de associação entre o conjunto de observações e o modelo, e, conseqüentemente, decodifica o sinal de fala numa palavra presente no vocabulário. Entretanto existem limitações, pois a sua aplicação é possível apenas quando se trabalha com palavras isoladas.

O problema de se associar uma seqüência de observações à uma seqüência de palavras é ainda mais complexo. Inicialmente a dificuldade era determinar o melhor HMM para uma dada seqüência de observações, agora é necessário determinar a melhor combinação de HMM's para uma dada seqüência de observações.

O reconhecimento de palavras conectadas e fala contínua exigem algoritmos eficientes e que requeiram mais uma vez um baixo custo computacional. Os algoritmos de decodificação *Level Building* [17][19] e *One Step* [19][9] são os mais indicados na realização dessa tarefa.

3.3.1 Decodificador *Level Building*

Suponha que uma elocução, pronunciada por um locutor qualquer, contenha dois dígitos concatenados (dígitos de 0 a 9). Conhecendo os HMM's de cada dígito, como proceder para a determinação da seqüência de dois dígitos correta?

Uma alternativa seria detectar a fronteira entre os dois dígitos, e analisar o problema como sendo de dígitos isolados. Apesar de simples, esta alternativa não teria eficácia. Alguns dígitos possuem pequenos intervalos de oclusão, que pode ser facilmente verificado nos dígitos 4 e 8, por causa da ocorrência da oclusiva [t].

Outra alternativa seria criar novos HMM's, como resultado da concatenação de dois HMM's quaisquer. Esse novo conjunto de modelos somaria um total de 10^2. A tarefa de reconhecimento ou decodificação poderia ser resolvida utilizando-se os algoritmos *forward* e de Viterbi na determinação do modelo mais provável, porém não é a mais adequada devido ao alto custo computacional.

Uma solução eficiente está numa formulação que leve em consideração as duas alternativas anteriores, e que apresente um baixo custo computacional. Esse método é o algoritmo *Level Building*.

Considere a elocução de dois dígitos, Figura 3.1, previamente processada pelas etapas de extração de parâmetros e quantização vetorial. Assim, dispõe-se de uma seqüência de observações que representam de forma compacta o conteúdo do sinal.

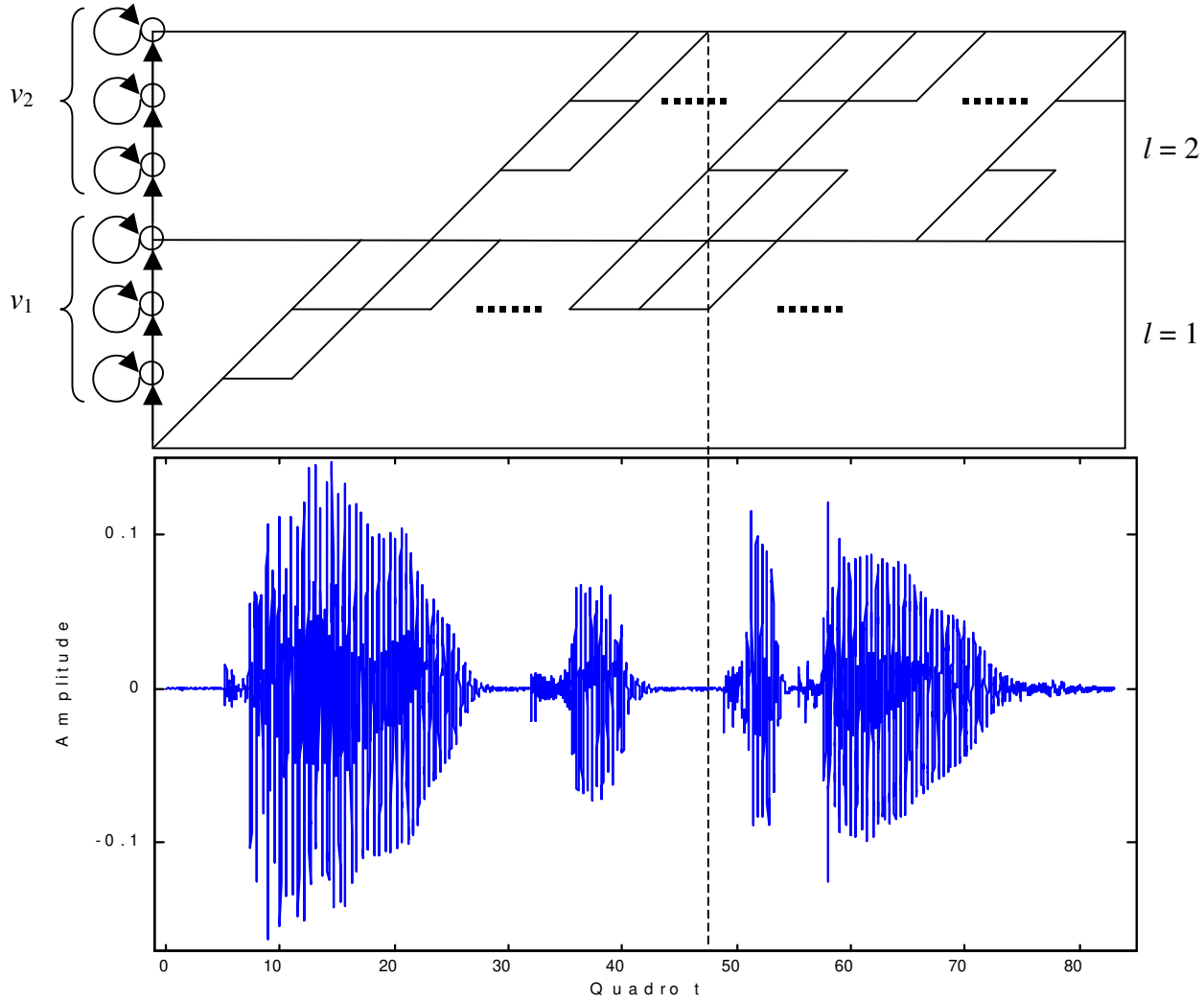

Figura 3.1 Algoritmo *Level Building* aplicado sobre a seqüência de dígitos “oito três”.

O algoritmo *Level Building* não sabe em qual quadro está a fronteira entre os dígitos (linha tracejada da Figura 3.1), logo supõe todos os quadros como sendo prováveis fronteiras. A *provável fronteira* é a região localizada entre os níveis l=1 e l=2, onde cada nível representa a posição do dígito na seqüência.

Aplica-se o algoritmo de Viterbi sobre a seqüência de observações de comprimento T quadros. O algoritmo de Viterbi determina os caminhos de maior probabilidade, os quais começam no quadro t=1 e correspondem aos traçados contínuos da Figura 3.1. Para cada quadro $t \leq T$, armazenam-se os valores da probabilidade acumulada $P_l^v(t)$ no último estado do HMM. Esse procedimento é realizado para cada palavra v_l do vocabulário de dígitos ($0 \leq v_l \leq 9$). Em seguida, as probabilidades acumuladas são comparadas e o HMM vencedor $W_l^G(t)$ e o valor da probabilidade do HMM vencedor $P_l^G(t)$ são armazenados (G é o identificador da probabilidade ou do modelo vencedor).

O segundo nível, que representa o segundo dígito, pode iniciar a partir de qualquer um dos quadros do intervalo $1 \leq t \leq T$. Assim, o algoritmo de Viterbi testa não só a transição entre estados, como também a transição entre níveis. Para isso, utiliza-se a probabilidade acumulada do nível anterior $P_{l-1}^G(t-1)$ como probabilidade inicial. O algoritmo de Viterbi é novamente aplicado para cada HMM. Em seguida, as probabilidades acumuladas são comparadas e o HMM vencedor $W_l^G(t)$ e o valor da probabilidade do HMM vencedor $P_l^G(t)$ são armazenados. A seqüência vencedora é a que apresenta a maior probabilidade acumulada no quadro T. As palavras que compõem a seqüência vencedora são obtidas pelo procedimento de *backtracking*, usando-se a informação do quadro inicial armazenado na variável $F_l^G(t)$.

Pode-se observar que o algoritmo de Viterbi foi aplicado somente 2×10 vezes ($2 \times$ número de HMM's), ao invés de 10^2 vezes. Essa diferença é ainda maior quando se trabalha com vocabulários grandes.

O pseudocódigo do algoritmo *Level Building* é apresentado abaixo.

1. *Decodificação*

Para $1 \le l \le L$, **faça**: // iteração sobre o nível

Para $1 \le v \le V$, **faça**: // iteração sobre as palavras

Para $1 \le t \le T$, **faça**: // iteração no tempo

Calcula $P_l^v(t)$.

Armazena em $F_l^v(t)$ o quadro inicial da palavra *v*.

Para $1 \le t \le T$, **faça**: // atualiza variáveis ótimas

Armazena em $W_l^G(t)$ o HMM vencedor do quadro *t* e nível *l*.

Armazena em $P_l^G(t)$ a probabilidade do HMM vencedor do quadro *t* e nível *l*.

Armazena em $F_l^G(t)$ o quadro inicial da palavra vencedora do quadro t e nível *l*.

Se $l < L$, **faça**:

Para $1 \le v \le V$, **faça**:

$P_{l+1}^v(t) = P_l^G(t)$. // iniciação das probabilidades para o próximo nível

2. *Procedimento de Backtracking*

2.1. *Iniciação*

$$L^* = \arg\max_{1 \le l \le L} \left[W_l^G(T) \right] \tag{3.48}$$

$$t = \mathrm{T} \tag{3.49}$$

$$W(L^*) = W_{L^*}^G(t) \tag{3.50}$$

2.2. *Recursão*

Para $L^*\text{-}1 \ge l \ge 1$, **faça**:

$$W(l) = W_l^G\left(F_{l+1}^G(t) - 1\right) \tag{3.51}$$

$$t = F_{l+1}^G(t) - 1, \tag{3.52}$$

onde *t* é o quadro final da palavra do nível precedente.

Algumas adaptações podem ser feitas para tornar o algoritmo ainda mais eficiente. São elas:

- Considerar a duração de um nível no mínimo igual ao número de estados do HMM.
- Desconsiderar as fronteiras entre níveis que apresentarem probabilidades acumuladas inferior a um certo limiar. A escolha deste limiar pode seguir vários critérios, os quais são discutidos na referência [19].

O processamento do decodificador *Level Building* requer o armazenamento em memória de toda a elocução e, portanto, não processa em sincronismo com o tempo. Assim, uma nova metodologia gerada a partir deste algoritmo foi apresentada na literatura para processar uma elocução em sincronismo com tempo. Ela é apresentada em seguida.

3.3.2 Decodificador *One Step*

A literatura apresenta outros nomes para esse algoritmo como *One Pass* [19] ou *Frame Synchronous* [9].

Apesar da lógica deste algoritmo ser mais complexa que a do algoritmo *Level Building*, a sua implementação computacional é mais simplificada. Isso é tarefa da programação dinâmica implícita no algoritmo.

A programação dinâmica requer somente a informação do instante anterior para a tomada de decisão do próximo instante. Por isso a necessidade de memória e processamento é minimizada.

O processamento síncrono com o tempo faz com que o algoritmo indique a seqüência de palavras vencedora a cada instante de tempo. Levando-se em consideração o desconhecimento do número de palavras presentes na seqüência que está sendo processada, o algoritmo determinará a melhor seqüência para cada tamanho de seqüência possível e, por último, a seqüência vencedora.

Exemplificando o que foi dito, para uma frase com sete dígitos pronunciados por um locutor qualquer, à medida que o sinal vai sendo processado, o algoritmo determina as seqüências vencedoras de tamanhos 1 a L a cada quadro, onde $L = 7$ níveis. No processamento do último quadro da frase, o algoritmo decide novamente pelas seqüências mais prováveis e, entre essas, a vencedora. Pode ocorrer do algoritmo escolher uma seqüência de seis ou menos dígitos como

sendo a vencedora, e, assim, pode-se dizer que ocorreu a remoção de uma palavra. O contrário disso seria obter uma seqüência vencedora com mais de sete níveis, e neste caso, uma palavra adicional foi inserida.

Para o melhor entendimento deste algoritmo, é necessário enumerar as diferentes situações que envolvem o problema. Essa é a grande complexidade deste algoritmo, pois no algoritmo *Level Building* as decisões são tomadas ao final de cada nível, depois de processado todos os quadros; ao passo que o *One Step* realiza toda a lógica de decisão para todos os níveis a cada quadro do sinal de fala.

O pseudocódigo do algoritmo *One Step* fica:

Para $1 \le t \le T$, **faça**:

Para $1 \le l \le L$, **faça**:

Para $1 \le v \le V$, **faça**:

Algoritmo de Viterbi.

Armazena em $W_l^G(t)$ o HMM vencedor do quadro *t* e nível *l*.

Armazena em $P_l^G(t)$ a probabilidade do HMM vencedor do quadro *t* e nível *l*.

Armazena em $D_l^G(t)$ a duração do HMM vencedor do quadro *t* e nível *l*.

Backtracking

O algoritmo de Viterbi, que está presente no laço principal do algoritmo *One Step*, sofreu alguns ajustes devido às particularidades do problema no qual foi inserido. Assim, considerando um quadro *t* qualquer, um nível *l* e um modelo *v* o algoritmo de Viterbi fica:

1. *Iniciação*

$$P_l^v(0) = P_{l-1}^G(t) \tag{3.53}$$

$$D_l^v(0) = 0 \tag{3.54}$$

2. *Recursão*

Para $1 \le j \le N_v$, **faça**:

$$\delta(j) = \max_{0 \le i \le N}[P_l^v(i)\, a_{ij}] \tag{3.55}$$

$$\psi(j) = D_l^v\left(\arg \max_{0 \le i \le N}[P_l^v(i)a_{ij}]\right) + 1 \quad (3.56)$$

// incrementa de 1 a duração da palavra *v* no nível *l*

3. *Finalização*

Para $1 \le i \le N_v$, **faça**:

$$P_l^v(i) = \delta(i)b_i(\mathbf{o}_t) \quad (3.57)$$

$$D_l^v(i) = \psi(i) \qquad \text{// atualiza a matriz de duração} \quad (3.58)$$

O estado *i*=0 é fictício. Ele representa somente a fronteira entre dois níveis. Por isso, a recursão do algoritmo de Viterbi inclui, além das transições entre estados, a transição entre níveis. A variável temporária $\psi(j)$ armazena a duração do modelo *v* no nível *l* e estado *j*, e a variável temporária $\delta(j)$ armazena a probabilidade acumulada desconsiderando a probabilidade de emissão, que é acrescentada na finalização do algoritmo. Isso reduz algumas operações desnecessárias.

A variável $D_l^G(t)$ é utilizada no procedimento de busca pela seqüência de palavras vencedora, também conhecido como *backtracking*. O *backtracking* retorna a seqüência de palavras reconhecidas de comprimento L^* através de *W*(*l*). O algoritmo de *backtracking* é descrito a seguir.

1. *Iniciação*

$$L^* = \arg \max_{1 \le l \le L}[W_l^G(T)] \quad (3.59)$$

$$t = \mathrm{T} \quad (3.60)$$

2. *Recursão*

Para $L^* \ge l \ge 1$, **faça**:

$$W(l) = W_l^G(t) \quad (3.61)$$

$$t = t - D_l^G(t) \quad (3.62)$$

As adaptações que tornam o algoritmo ainda mais eficiente são semelhantes às utilizadas no algoritmo *Level Building*.

Os resultados obtidos pelos algoritmos *One Step* e *Level Building* são iguais. Além disso, a simplicidade de implementação, uso reduzido de memória e baixo custo computacional tornam o algoritmo *One Step* a melhor opção.

Por outro lado, o algoritmo *One Step* retorna somente a seqüência vencedora e descarta as outras candidatas, enquanto que o algoritmo *Level Building* também pode fornecer todas seqüências candidatas. Conforme apresentado em [9], pequenas adaptações podem ser feitas no algoritmo *One Step*, as quais possibilitam armazenar duas melhores seqüências candidatas à seqüência vencedora.

Capítulo 4
TREINAMENTO E RESULTADOS DE SIMULAÇÃO DO SISTEMA DE RECONHECIMENTO DE DÍGITOS CONECTADOS

4.1 Introdução

Os capítulos anteriores descrevem as principais características de um sistema de reconhecimento de fala. No entanto, algumas questões relacionadas à implementação deste sistema ainda não foram discutidas.

A primeira questão diz respeito ao treinamento dos modelos ocultos de Markov (HMM's). O treinamento apresenta algumas particularidades e, neste trabalho, é utilizada uma metodologia de treinamento por segmentação automática da seqüência de treinamento [18].

A outra questão está relacionada com a base de sinais. Todo sistema de reconhecimento de fala utiliza, no treinamento dos HMM's, um conjunto de frases, que compõem uma base de sinais. Algumas vezes, a qualidade da base de sinais influencia sobremaneira o desempenho do sistema.

4.2 Base de Sinais

Foram usadas duas bases de sinais para avaliação do desempenho do sistema. Uma do inglês americano chamada de TI *Digits* e outra do Laboratório de Processamento Digital de Fala que foi denominada neste trabalho de LPDF Dígitos.

4.2.1 TI Digits

A TI *Digits* é uma base de dígitos do inglês americano. Foi gravada na freqüência de amostragem de 20 kHz. Esta base recebeu filtragem passa-banda e foi dizimada para a freqüência de 8 kHz. Foram tomados cuidados especiais para gravação desta base no que diz respeito ao ruído ambiente e ao equipamento de gravação. Além disso, essa base incorporou os principais dialetos dos Estados Unidos, através da divisão do país em 21 regiões dialetais [10]. Cada região dialetal contribuiu com pelo menos cinco locutores femininos e cinco locutores masculinos. Os negros foram representados por um dialeto adicional.

O vocabulário desta base compreende os dígitos *zero*, *oh*, *one*, *two*, *three*, *four*, *five*, *six*, *seven*, *eight* e *nine*.

A TI *Digits* é completa, pois possui seqüências de dígitos de vários tamanhos, desde 1 até 7 dígitos conectados (não há seqüências de 6 dígitos). Essa base é dividida em porções adulta e infantil, e em cada uma das porções há um conjunto de treinamento e de teste.

Neste trabalho foi utilizada somente a porção adulta da base. A Tabela 4.2 mostra o número de locutores para cada conjunto.

Tabela 4.2 Porção adulta da base de sinais TI *Digits*

Conjunto	Locutores	
	Femininos	Masculinos
Treinamento	57	55
Teste	57	56

Cada locutor pronunciou 77 frases geradas aleatoriamente. Essas frases foram divididas nas seguintes categorias:

- 22 seqüências de dígitos isolados, sendo duas repetições de cada uma das pronúncias dos 11 dígitos do vocabulário;
- 11 seqüências de dois dígitos;
- 11 seqüências de três dígitos;
- 11 seqüências de quatro dígitos;

- 11 seqüências de cinco dígitos;
- 11 seqüências de sete dígitos.

A Tabela 4.3 apresenta a divisão da porção adulta da base em termos do número total de frases pronunciadas nos conjuntos de treinamento e teste. Conforme descrito na documentação da base TI *Digits*, quatro frases foram removidas da base devido a erros cometidos por alguns locutores. Uma dessas frases erradas foi reaproveitada e inserida no conjunto de seqüências de 5 dígitos.

Tabela 4.3 Número de elocuções da porção adulta da base de sinais TI *Digits*

Número de dígitos na seqüência	Número de frases	
	Conjunto de treinamento	Conjunto de teste
1	2464	2486
2	1231	1243
3	1232	1243
4	1232	1242
5	1232	1244
7	1231	1242
total	8622	8700

4.2.2 LPDF Dígitos

A LPDF Dígitos [6] é uma pequena base de dígitos conectados em Português gravada na freqüência de amostragem de 11,025 kHz. O ambiente de gravação é o de escritório e feito através de placas de som de computadores pessoais.

Ela foi dizimada para a freqüência de 8 kHz. Entretanto, antes da dizimação, foi realizada uma filtragem passa-banda. A função de transferência desse filtro é apresentada no capítulo 2.

O vocabulário desta base compreende dígitos zero, um, dois, três, quatro, cinco, seis, sete, oito, nove e meia.

A base é dividida num conjunto de treinamento e de teste como mostrado na Tabela 4.1. O conjunto de treinamento possui frases com seqüências de oito dígitos e estas são diferentes das

frases usadas no conjunto de teste. Os locutores também foram separados para possibilitar a condição de independência de locutor.

Tabela 4.1 Base de sinais LPDF Dígitos

Conjunto	Locutores	
	Femininos	Masculinos
Treinamento	13	18
Teste	4	5

Cada locutor pronunciou 11 frases escolhidas aleatoriamente de um conjunto de 55 frases diferentes. Não são permitidas repetições entre as frases para um mesmo locutor. Assim, o conjunto de treinamento totaliza 341 frases e o conjunto de teste totaliza 99 frases.

4.3 Treinamento

Na etapa de treinamento são estimados os HMM's referentes a cada palavra do vocabulário. A estimação dos HMM's requer um conjunto representativo de elocuções que permita uma boa estimativa de cada modelo. Além disso, as estimativas serão ainda melhores se cada elocução for originária de frases equivalentes às utilizadas no reconhecimento. O treinamento voltado para o reconhecimento de dígitos conectados, que é apresentado na seqüência, necessita de modelos iniciais de palavras isoladas.

4.3.1 Palavras Isoladas

O treinamento de palavras isoladas é o caso mais simples de treinamento dos HMM's. Esse procedimento é baseado na seleção de conjuntos de treinamento para cada palavra do vocabulário e a estimativa do HMM correspondente. A Figura 4.1 apresenta o diagrama de blocos do procedimento de treinamento.

O treinamento requer o conhecimento a priori dos HMM's, ou seja, é necessário iniciá-los. A iniciação dos HMM's consiste em estimar a matriz de probabilidade de transição **A** e a matriz probabilidade de emissão **B**. A literatura apresenta alguns procedimentos de iniciação dos HMM´s [19][8]. As matrizes **A** e **B** não necessitam do mesmo procedimento de iniciação devido

às diferentes influências que elas oferecem no processo de treinamento. A matriz **A** é iniciada com distribuição uniforme de probabilidade. Por outro lado, a matriz **B** requer uma boa iniciação, pois esta influenciará na convergência do algoritmo. Assim, utiliza-se a segmentação uniforme da seqüência de observação descrita a seguir, para iniciação da matriz **B**.

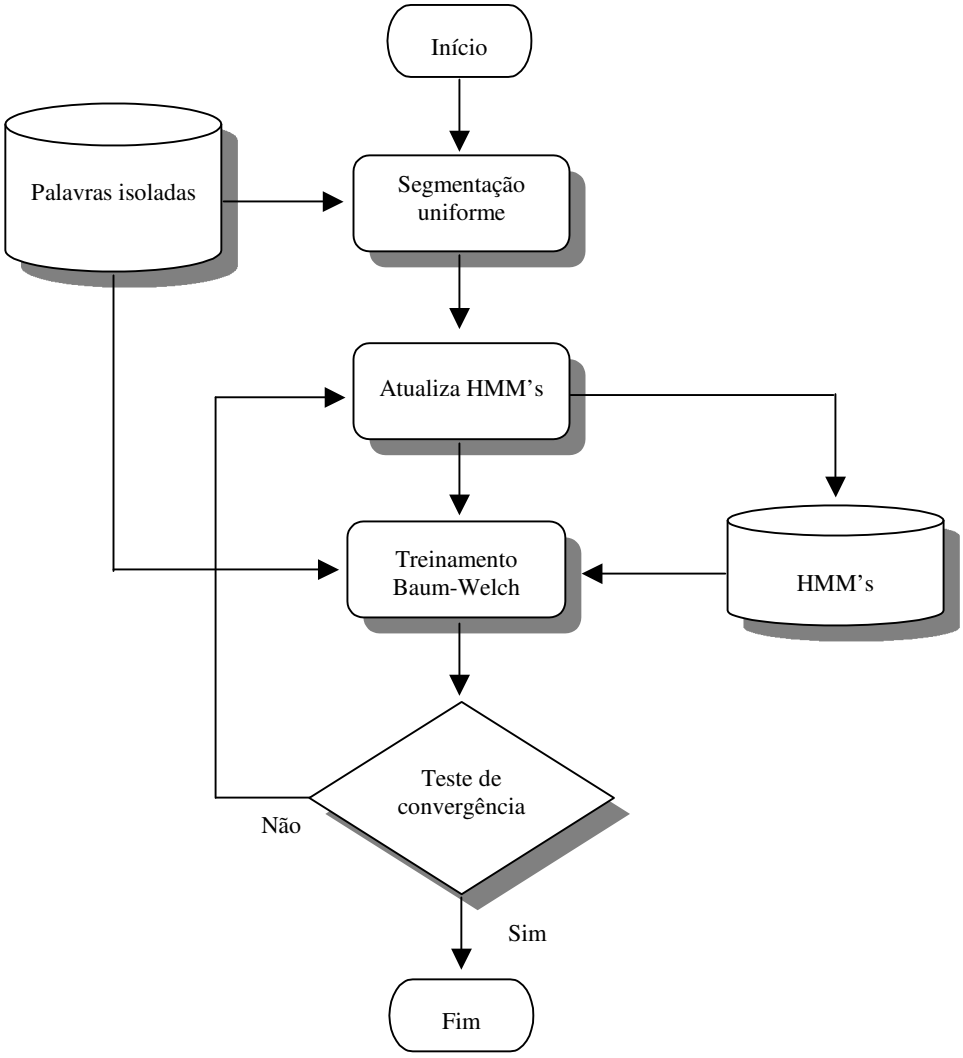

Figura 4.1 Diagrama de blocos do algoritmo de treinamento de palavras isoladas

Para descrever o processo de segmentação uniforme da seqüência de observação, relembra-se que cada elocução é representada por uma seqüência de observações, onde cada observação é resultado da quantização vetorial dos parâmetros extraídos de um quadro do sinal. A seqüência de observações é dividida pelo número de estados do modelo, de forma a tornar o

número de observações por estado uniforme. Assim, cada estado emitirá um conjunto de observações obtidas dessa segmentação uniforme. Conhecidas as observações em cada estado do HMM, a estimação das probabilidades de emissão será resultado dum processo de contagem das ocorrências das observações em cada estado do modelo.

Terminada a iniciação das matrizes **A** e **B**, o próximo passo consiste no treinamento via algoritmo Baum-Welch. A convergência ocorre quando a distância entre os modelos antes e depois de uma época de treinamento for menor que 0.1% [19]. A medida de distância é dada pela normalização da verossimilhança média de todas as elocuções de treinamento de um determinado modelo [21].

4.3.2 Palavras Conectadas

Os sistemas de reconhecimento de palavras conectadas, mais especificamente dígitos conectados, requerem um treinamento que leve em consideração as características das palavras quando pronunciadas em seqüência. Nesta condição, as palavras são pronunciadas com maior rapidez e suas durações são menores que no caso das palavras serem pronunciadas isoladamente. Além disso, algumas palavras podem ser interligadas durante a pronúncia da seqüência, suprimindo com freqüência pausas entre palavras. Por isso, há necessidade de se treinar os HMM's de cada palavra a partir de uma base de sinais também composta por seqüências de palavras, semelhantes às utilizadas no reconhecimento.

Considerando agora um conjunto de treinamento composto por seqüências de palavras conectadas, a tarefa de segmentação das seqüências de dígitos deve ser bem executada, pois influenciará diretamente no desempenho do sistema.

A segmentação automática apresentada em [18] requer uma boa iniciação dos HMM's. Essa exigência é ainda maior quando se trabalha com modelos discretos [19]. Por isso, os modelos são iniciados por um pequeno conjunto de treinamento de dígitos isolados. Neste caso, o treinamento faz uso do algoritmo Baum-Welch para o treinamento de dígitos isolados, conforme descrito na seção referente ao treinamento de palavras isoladas. Depois de treinados separadamente, os HMM's são colocados no recipiente *HMM de cada palavra do vocabulário*, conforme mostrado na Figura 4.2.

Iniciando o procedimento de treinamento, os *Arquivos de Treinamento* compostos por dígitos conectados são segmentados em dígitos isolados pelo algoritmo de *Segmentação Automática*. A *Segmentação Automática* é feita pelo algoritmo *Level Building*, que determina a fronteira entre cada dígito. A particularidade do treinamento é o conhecimento a priori da seqüência correta de dígitos.

As *Elocuções* geradas pela *Segmentação Automática* de cada arquivo são armazenadas em recipientes associados ao HMM correspondente. Depois que todos os *Arquivos de Treinamento* são segmentados, é executado o *Treinamento* propriamente dito.

O *Treinamento* de cada HMM é feito através do algoritmo Baum-Welch. Nesta fase, são obtidas as informações para o modelo de duração de palavra.

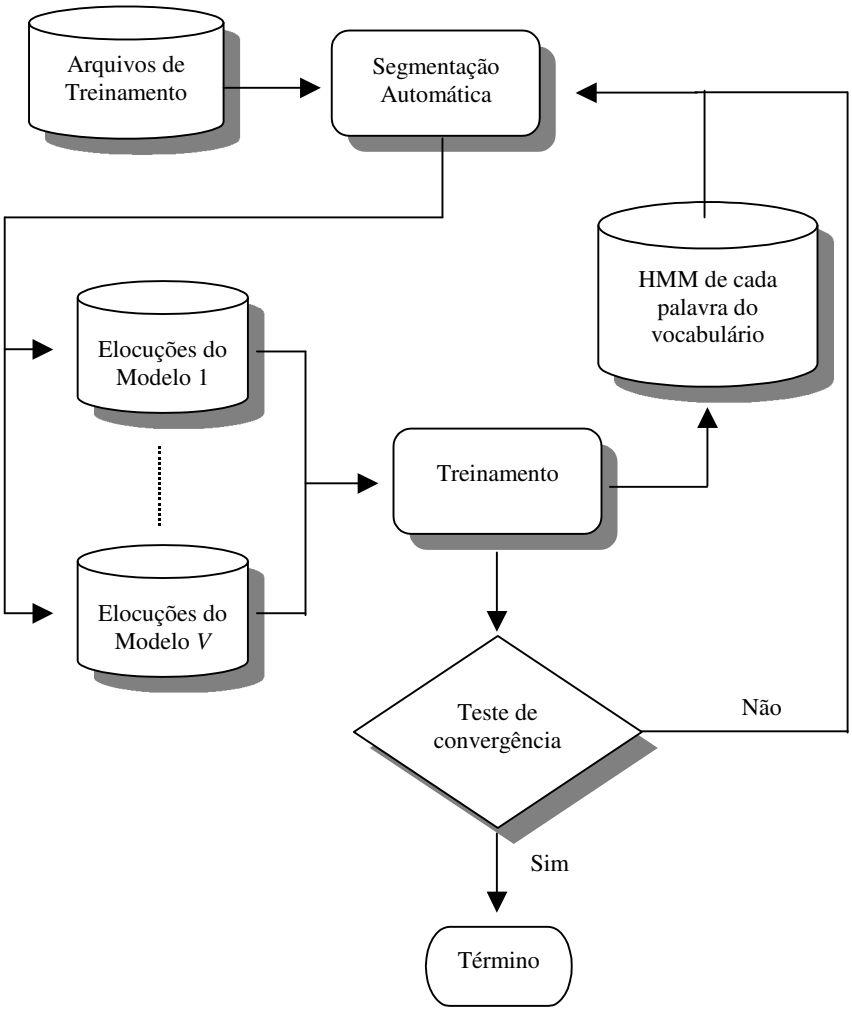

Figura 4.2 Diagrama de blocos do sistema de treinamento

Terminado o *Treinamento* de todos os modelos, o teste de convergência é executado. A convergência ocorre quando a distância entre os modelos antes e depois de uma época de treinamento for menor que 0.1%. O procedimento utilizado é igual ao que foi descrito no treinamento de palavras isoladas.

4.4 Simulações

Alguns testes foram realizados com ambos os algoritmos de reconhecimento, *Level Building* e *One Step*, e confirmou-se a igualdade nos resultados. Entretanto, optou-se em trabalhar somente com o algoritmo *One Step*, visando a implementação em tempo real. Além disso, acrescenta-se o modelo de duração de palavra como pós-processamento. Não foi feita nenhuma ponderação no modelo de duração de palavra.

Foram realizados experimentos com as duas bases de sinais descritas neste trabalho. Nestes experimentos são avaliadas as taxas de acerto de palavras e frases, tanto para o conjunto de treinamento quanto para o conjunto de teste. Os resultados obtidos pela base de treinamento são muito úteis, pois permitem avaliar a convergência de algoritmo de treinamento.

4.4.1 Resultados – TI *Digits*

A TI *Digits* é uma base amplamente utilizada pela comunidade científica. Algumas configurações propostas na literatura foram aproveitadas nas simulações [18][14].

Foram realizados dois experimentos com a base TI *Digits*. As principais diferenças entre os experimentos realizados são a inclusão do modelo de silêncio e o procedimento de iniciação dos HMM's. As etapas de *Análise Espectral* e *Quantização Vetorial* são iguais, e as configurações são as seguintes:

1) Análise Espectral

- Filtragem de pré-ênfase com fator 0.95;
- Aplicação de janelas de Hamming de 20 ms e superposição entre janelas de 50%;
- Extração de 12 coeficientes mel cepstrais e energia, e mais suas derivadas primeira e segunda, resultando num vetor de parâmetros de tamanho 39.

2) Quantização Vetorial

- 6 *codebooks* independentes, um para cada parâmetro;

• Foram utilizados *codebooks* de 256 vetores para os parâmetros mel cepstrais e de 32 vetores para as energias.

4.4.1.1 Experimento I

Neste experimento não foi utilizado modelo de silêncio, conforme apresentado por [18]. Além disso, no procedimento de iniciação dos HMM's, selecionou-se a porção da base TI *Digits* composta por frases de apenas um dígito, e realizou-se treinamento de palavras isoladas a partir dela. Utilizou-se HMM de 15 estados e 3 transições por estado.

Em seguida, iniciou-se o procedimento de treinamento de dígitos conectados com toda a base TI *Digits*, incluindo a porção da base utilizada na iniciação. O treinamento convergiu em 9 iterações, e os resultados gerados são apresentados na Tabela 4.4.

Tabela 4.4 Resultados com a base TI *Digits* sem o modelo de silêncio.

Conjunto	Acerto de frases (%)	
	TC	TD
Treinamento	98.49	95.23
Teste	97.55	93.51

Os resultados obtidos com a base TI *Digits* consideraram o número de dígitos na frase conhecido TC e desconhecido TD, objetivando explorar melhor as características da base.

Apesar dos resultados na condição TC serem satisfatórios, verifica-se uma queda muito grande no desempenho na condição TD. Essa diferença não foi verificada em [18].

Analisando as características da base TI *Digits*, verifica-se a necessidade de se utilizar modelo de silêncio. Os períodos de silêncio no início e no fim de cada frase são muito grandes, e em várias frases foi constatada a presença de longas pausas entre os dígitos. O experimento II visa levar em consideração esse fato.

4.4.1.2 Experimento II

Neste experimento, o propósito é iniciar os HMM's a partir da porção composta por frases de apenas um dígito do conjunto de treinamento da base TI *Digits* e acrescentar um modelo para

o silêncio. O modelo para o silêncio é gerado via treinamento de segmentos de silêncio retirados de frases do conjunto de treinamento da base TI *Digits*. Entretanto, verificou-se melhores resultados quando os HMM's são iniciados pelo seguinte procedimento:

- Segmentação manual de um conjunto pequeno de elocuções (seis) para cada dígito do conjunto de treinamento da base TI *Digits*, igualmente dividido entre locutores masculino e feminino.
- Iniciação dos HMM's a partir do treinamento de palavras isoladas das elocuções segmentadas.

Todos HMM's possuem 15 estados e 3 transições por estado, com exceção do HMM do silêncio que possui apenas dois estados.

Depois de iniciados os HMM's, passa-se para o procedimento de treinamento de palavras conectadas. Para cada frase, considera-se um dígito adicional no início e no fim correspondente ao silêncio. Esse procedimento é realizado tanto no treinamento quanto no reconhecimento. Após o treinamento, que convergiu em 15 iterações, obteve-se os resultados da Tabela 4.5.

Tabela 4.5 Resultados com a base TI *Digits* usando modelo de silêncio.

Conjunto	Acerto de frases (%)	
	TC	TD
Treinamento	98.74	97.61
Teste	97.93	96.15

Constata-se um aumento significativo do desempenho do sistema, tanto para o conjunto de treinamento quanto para o conjunto de teste. A melhora foi mais acentuada na condição na qual se desconhece o número de dígitos na frase.

Foram utilizados procedimentos distintos no reconhecimento das frases nas condições TC e TD. No caso TC, o formato das seqüências de dígitos é semelhante ao utilizado no treinamento, ou seja, considera-se um dígito adicional no início e no fim da seqüência correspondendo ao silêncio. Entretanto, na condição TD em que se desconhece o número de dígitos da seqüência, adicionou-se o silêncio ao conjunto de palavras do vocabulário, e aumentou-se o número de níveis de busca do algoritmo de reconhecimento para 10. As alternativas propostas para os testes

sob as condições TD e TC, apesar de não serem iguais, foram as que apresentaram os melhores resultados.

Objetivando avaliar a influência do modelo de duração de palavra no desempenho do sistema, repetiram-se os testes sem a inclusão do modelo de duração, e os resultados são mostrados na Tabela 4.6.

Tabela 4.6 Resultados com a base TI *Digits* usando modelo de silêncio, sem a inclusão do modelo de duração de palavra.

Conjunto	Acerto de frases (%)	
	TC	TD
Treinamento	98.55	97.02
Teste	97.49	95.82

Comparando-se as Tabelas 4.5 e 4.6, constata-se uma redução significativa do acerto de frases, quando se suprime o modelo de duração de palavra. A redução do acerto de frases é ainda mais importante na condição TD.

A Tabela 4.7 mostra o desempenho do sistema em função do número de dígitos das frases. O modelo de duração palavra é utilizado. Observa-se que a taxa de acerto do conjunto de treinamento sofre um grande queda para frases com três dígitos ou mais. Essa queda é maior para as frases com sete dígitos. Por outro lado, a taxa de acerto do conjunto de teste começa a ter uma queda acentuada para frases com dois dígitos ou mais. Verifica-se novamente uma queda mais acentuada para frases de sete dígitos. Esse comportamento é verificado principalmente na condição TD.

Tabela 4.7 Resultados com a base TI *Digits* detalhados em número de dígitos.

Número de dígitos na frase	Acerto de frases (%)			
	Treinamento		Teste	
	TC	TD	TC	TD
1	100	99.84	99.72	99.03
2	99.68	99.03	99.12	97.83
3	99.03	97.40	97.91	96.30
4	98.78	97.32	97.02	94.36
5	98.30	96.75	96.46	93.37
7	95.37	93.10	95.57	93.08

Os resultados mostrados nas Tabelas 4.5 e 4.7 estão expressos em termos de acertos de frases. No entanto, isto não caracteriza muito bem o comportamento do sistema. A ocorrência de erros de frases pode ser resultado da substituição (*S*), remoção (*R*) ou inserção (*I*) de alguma palavra da frase. A taxa de palavras erradas (*T*) é dada pela equação (4.1).

$$T(\%) = \frac{S + R + I}{N_P}, \tag{4.1}$$

onde N_P é o número total de palavras.

Refazendo-se a Tabela 4.5 e calculando-se a taxa de acerto de palavras, que a partir da equação (4.1) é igual a (100-*T*)%, chega-se à Tabela. 4.8.

Tabela 4.8 Resultados com a base TI *Digits* considerando apenas o acerto de palavras.

Conjunto	Acerto de palavras (%)	
	TC	TD
Treinamento	99.58	99.18
Teste	99.31	98.64

Escrevendo-se os resultados em função do número de substituições, inserções e remoções de palavras, e estruturando-os de acordo com o número de dígitos de cada frase, obtêm-se as Tabelas 4.9 e 4.10, para os conjuntos de treinamento e de teste respectivamente.

Tabela 4.9 Taxa de erro de palavras para o conjunto de treinamento da base TI *Digits*.

Número de dígitos na frase	N_P	Erro de palavras – Conjunto de treinamento					
		TC		TD			
		S	*T* (%)	*S*	*I*	*R*	*T* (%)
1	2464	0	0	0	4	0	0.16
2	2462	4	0.16	4	6	2	0.49
3	3696	12	0.32	14	7	17	1.03
4	4928	15	0.30	17	6	11	0.69
5	6160	23	0.37	22	8	14	0.71
7	8617	66	0.77	58	14	26	1.14
Resultado global	28327	120	0.42	115	45	70	0.81

Tabela 4.10 Taxa de erro de palavras para o conjunto de teste da base TI *Digits*.

Número de dígitos na frase	N_P	Erro de palavras – Conjunto de teste					
		TC		TD			
		S	*T* (%)	*S*	*I*	*R*	*T* (%)
1	2486	7	0.28	8	18	0	1.05
2	2486	11	0.44	10	18	1	1.17
3	3729	28	0.75	29	16	11	1.50
4	4968	41	0.83	39	27	17	1.67
5	6220	49	0.79	47	29	19	1.53
7	8694	60	0.69	58	27	16	1.16
Resultado global	28583	196	0.69	191	135	64	1.36

As Tabelas 4.9 e 4.10 demonstram claramente o que influencia o aumento da taxa de erro, quando se trabalha na condição TD. Isso se deve às palavras que são inseridas ou removidas na frase, representando aproximadamente 50% do total de erros de palavras. A maioria dos erros cometidos pelo sistema deve-se principalmente a:

- Inserção da palavra "six" no início da frase no lugar do silêncio. Isto é ocorre, pois a palavra "six" é praticamente um ruído colorido e pode ser confundida com o silêncio;
- Inserção da palavra "oh" após a palavra "zero". O final da palavra "zero" se confunde com a palavra "oh", o que é acentuado devido ao efeito de co-articulação;
- Remoção da palavra "eight" e da palavra "oh", pois, para vários locutores, essas palavras possuem duração muito pequena.

4.4.2 Resultados - LPDF Dígitos

A definição das características de cada parte do sistema de reconhecimento é resultado de testes exaustivos. Como o número de variáveis que são ajustadas é muito grande, optou-se por considerar ao menos em parte resultados obtidos por outros trabalhos do Laboratório de Processamento Digital de Fala [11][6].

Mais uma vez, as simulações são divididas em dois experimentos, objetivando analisar a influência do modelo de silêncio no desempenho do sistema. Além disso, no dois experimentos, a *Análise Espectral* e *Quantização Vetorial* são semelhantes, e as configurações são iguais às utilizadas com a base TI *Digits*:

1) Análise Espectral

- Filtragem de pré-ênfase com fator 0.95;
- Aplicação de janelas de Hamming de 20 ms e superposição entre janelas de 50%;
- Extração de 12 coeficientes mel cepstrais e energia, e mais suas derivadas primeira e segunda, resultando num vetor de parâmetros de tamanho 39.

2) Quantização Vetorial

- 6 *codebooks* independentes, um para cada parâmetro;
- Foram utilizados *codebooks* de 256 vetores para os parâmetros mel cepstrais e de 32 vetores para as energias.

4.4.2.1 Experimento I

Neste experimento não é utilizado o modelo de silêncio. Além disso, considerando a característica da base LPDF Dígitos de possuir somente seqüências de oito dígitos, optou-se pelo seguinte procedimento de iniciação e treinamento dos HMM's:

- Segmentação manual de algumas seqüências de treinamento em palavras isoladas, formando um pequeno conjunto de aproximadamente oito elocuções de cada palavra, igualmente dividido entre locutores masculino e feminino.
- Treinamento dos modelos a partir das elocuções segmentadas. Os HMM's gerados a partir desse treinamento são os modelos iniciais do procedimento de treinamento.

Foram treinados HMM's de 15 estados e 3 transições por estado. Após o treinamento, que durou 13 iterações ou épocas, os HMM's gerados apresentaram o desempenho mostrado na Tabela 4.11. A busca feita pelo algoritmo de decodificação simulou frases com número de dígitos variando de um a oito.

Tabela 4.11 Resultados com a base LPDF Dígitos, sem o uso de modelo de silêncio

Conjunto	Acerto de frases (%)	
	TC	TD
Treinamento	99.41	97.65
Teste	90.91	89.90

O conjunto de teste apresenta uma baixa taxa de acerto de frases. A diferença acentuada entre os acertos de frases do conjunto de treinamento e teste não é verificada com base TI *Digits*. A queda no acerto de frases é causada possivelmente pela ocorrência de períodos de silêncio no início e no fim de cada frase. Assim, realiza-se no Experimento II uma modelagem mais precisa, com a adição de modelo de silêncio, objetivando melhorar o desempenho do sistema.

4.4.2.2 Experimento II

Nesse experimento é adicionado um modelo de silêncio. O modelo de silêncio é gerado via treinamento de segmentos de silêncio retirados de frases do conjunto de treinamento da base

LPDF Dígitos. Os HMM's de cada palavra do vocabulário são iniciados a partir do treinamento de palavras isoladas das elocuções segmentadas no Experimento I. Em seguida, realiza-se o procedimento de treinamento de palavras conectadas, que envolve todo o conjunto de treinamento. No treinamento, considera-se para cada frase um dígito adicional no início e no fim correspondendo ao silêncio. Todos HMM's possuem 15 estados e 3 transições por estado, com exceção do HMM do silêncio que possui apenas dois estados.

Após o treinamento, que durou 13 iterações ou épocas, os HMM's gerados apresentaram o desempenho da Tabela 4.12.

Tabela 4.12 Resultados com a base LPDF Dígitos usando modelo de silêncio.

Conjunto	Acerto de palavras (%)		Acerto de frases (%)	
	TC	TD	TC	TD
Treinamento	99.89	99.89	99.12	99.12
Teste	99.37	99.37	94.95	94.95

Apesar das frases que compõem os conjuntos de teste e treinamento possuírem número de dígitos fixo e igual a oito, a busca feita pelo algoritmo de decodificação simulou frases com número de dígitos desconhecido (condição TD), ou seja, seqüências de um a oito dígitos. Esse procedimento é relevante, pois simula uma situação onde se desconhece o número de palavras pronunciadas pelo locutor. Foram utilizados procedimentos iguais no reconhecimento das frases nas condições TC e TD. Além disso, o formato das seqüências de dígitos é semelhante ao utilizado no treinamento, ou seja, considera-se um dígito adicional no início e no fim da seqüência correspondendo ao silêncio.

Constatou-se que os resultados do reconhecimento foram iguais, como mostrado na Tabela 4.12, para os casos de frases com número de dígitos conhecido e desconhecido. Além disso, a incorporação de modelo de duração de palavra, conforme apresentado em [18], não influenciou nos resultados.

A explicação para tais resultados se dá pelas características dos conjuntos de treinamento e teste. Como já enfatizado, os conjuntos são formados por seqüências de oito dígitos, e essa

informação é possivelmente incorporada nos HMM's pelo processo de treinamento. Além disso, como o número de dígitos nas frases é fixo, não ocorre uma variação significativa na duração dos dígitos na frase. Por isso, a modelagem de duração de palavras não influencia nos resultados.

4.4.3 Considerações Finais

Através dos resultados das Tabelas 4.5, 4.8 e 4.12, pode-se observar que a diferença entre a taxa de acerto do conjunto de treinamento e de teste foi maior com a base LPDF Dígitos. Além disso, a taxa de acerto de frases para o conjunto de treinamento da própria LPDF Dígitos foi muito superior ao da base TI *Digits*. Atribui-se este resultado ao tamanho reduzido do conjunto de treinamento da base LPDF Dígitos. Levando-se em consideração o número de dígitos das frases (TI *Digits* até sete dígitos e LPDF oito dígitos), verifica-se uma equivalência probabilística entre a taxa de acerto de palavras e a taxa de acerto de frases.

Observa-se na Tabela 4.7 uma equivalência de resultados entre os conjuntos de treinamento e teste para frases de sete dígitos. Os resultados indicam que o procedimento de treinamento não modelou suficientemente seqüências de sete dígitos, pois se esperam normalmente resultados superiores para o conjunto de treinamento.

A utilização de modelo de duração de palavras melhora o desempenho do sistema, e, em último caso, não influencia nos resultados, caso constatado com a base LPDF Dígitos. Além disso, o custo computacional sofre um acréscimo desprezível. Por estas razões, considera-se esta alternativa excelente na melhora de desempenho do sistema de reconhecimento de fala.

As condições TC e TD são situações em que um sistema de reconhecimento de fala pode ser submetido. Se for dada liberdade ao locutor em pronunciar um número indefinido de dígitos, ter-se-á uma queda no desempenho do sistema, como demonstrado pelas Tabelas 4.5 ou 4.7. Entretanto, sempre que for possível definir o número de dígitos a ser pronunciado pelo locutor, ocorre melhora no desempenho e no grau de confiança do sistema. Essa questão será analisada numa condição de reconhecimento de dígitos conectados em tempo real, a ser apresentada no próximo capítulo.

Capítulo 5
IMPLEMENTAÇÃO DO SISTEMA DE RECONHECIMENTO DE DÍGITOS CONECTADOS EM TEMPO REAL

5.1 Introdução

Muitas vezes, as dificuldades de implementação de um sistema não são fáceis de se prever. As simulações, como as apresentadas no capítulo anterior, dão uma grande segurança ao trabalho, se for considerada a quantidade de testes e situações que se procura analisar. Mesmo assim, ainda há uma grande diferença entre abordar as características de um sistema de reconhecimento de fala a partir da sua simulação e realizar sua implementação em tempo real. Essa experiência prática é difícil de ser adquirida, principalmente pelo fato de não ser o enfoque principal de muitos projetos de pesquisa da comunidade científica. Neste capítulo, abordam-se alguns problemas advindos da implementação do sistema de reconhecimento de fala em tempo real.

5.2 Questões de Implementação em Tempo Real

O capítulo 4 apresenta os resultados de simulação do sistema de reconhecimento de dígitos conectados, realizados a partir de sinais de fala gravados em arquivo. Neste caso, como os sinais já estão gravados, não há preocupação com relação ao tempo de processamento dos mesmos. Na adaptação do sistema de reconhecimento de fala para o processamento em tempo real surge a questão do custo computacional.

O custo computacional depende diretamente da complexidade do sistema. No capítulo 3 são descritos os algoritmos utilizados no sistema de reconhecimento, como também algumas propostas de redução do custo computacional.

Além disso, para uma aplicação de reconhecimento de fala funcionar adequadamente em tempo real, não pode haver interrupção do sinal de fala. Considerando as características do sistema apresentado no capítulo anterior, a cada 10 ms (período de atualização do quadro) o sistema deve extrair parâmetros, realizar a quantização vetorial e executar o algoritmo de decodificação. Naturalmente, para a operação em tempo real, o esforço computacional deve ser inferior à capacidade de processamento da CPU no período de 10 ms.

O conhecimento dos instantes de enunciação é outra questão relacionada à implementação em tempo real. Longos períodos de silêncio podem representar gastos computacionais desnecessários. Além disso, o conhecimento do instante em que o enunciado inicia e termina serve de auxílio ao sistema de reconhecimento, influenciando positivamente no desempenho do mesmo.

Por último, uma aplicação em tempo real depende também da escolha do *hardware* adequado, que atenda às necessidades do sistema. Alguns comentários a respeito de *hardware* são feitos a seguir.

5.3 Escolha do *Hardware*

Os avanços tecnológicos vêm exigindo sistemas ou equipamentos com inúmeras vantagens funcionais, além de baixo custo e consumo de energia. As telecomunicações participam ativamente desse processo desde o início da migração dos sistemas analógicos para os mais recentes sistemas digitais. A criação de um dispositivo capaz de processar digitalmente um sinal analógico está entre os grandes avanços conseguidos. Este dispositivo é conhecido como Processador Digital de Sinais (em inglês, *Digital Signal Processor*, DSP).

Desde a sua concepção, os DSP's apresentam características muito atraentes, as quais os tornam vantajosos frente às arquiteturas dos microcontroladores e microprocessadores: baixo custo e baixo consumo de energia. Computadores pessoais, telefones celulares e outros equipamentos utilizados em telecomunicações, que dependem de bateria para funcionar ao invés de ficarem permanentemente ligados na rede de energia elétrica, requerem um processador com

as características de um DSP. Além destas características, o elevado desempenho em processamento de sinais é a grande atração desses processadores.

Já há um bom tempo, os trabalhos na área de reconhecimento de fala utilizaram os DSP's na realização da tarefa como um co-processamento. A possibilidade de se implementar um sistema de reconhecimento de fala completo em um DSP começou logo em seguida, a medida que os próprios processadores se tornaram mais rápidos e com maior capacidade de armazenamento.

Os sistemas de reconhecimento de fala comerciais, que são processados em DSP's, estão presentes principalmente na telefonia celular. Entretanto, aplicações mais complexas, que fazem uso de grandes vocabulários em sua base de dados, utilizam poderosos microprocessadores operando em paralelo, e até mesmo DSP's realizando tarefas de co-processamento.

Inicialmente, o trabalho de implementação em tempo real voltou-se para a programação dos algoritmos em um DSP (21062) da Analog Devices. Entretanto, devido às limitações de memória disponível no cartão do processador e ao seu desempenho não superior ao de um microcomputador Pentium III, a 500 MHz, optou-se por implementar o sistema em um computador pessoal.

A tarefa de implementação num microcomputador é mais simplificada, e a integração com um ambiente visual é imediata.

5.4 Adaptação do Sistema de Reconhecimento ao Processamento em Tempo Real

O código utilizado no capítulo 4 para a simulação do sistema de reconhecimento realiza separadamente as tarefas de processamento de toda a elocução. Assim, há uma dependência no tamanho da elocução a ser processada.

Na situação de processamento em tempo real, as tarefas de processamento precisam ser intercaladas, pois não se conhece o tamanho das elocuções. Entretanto, algumas variáveis, como aquelas dedicadas à tarefa de *backtracking*, dependem diretamente do tamanho da elocução.

Visando reduzir o tamanho das elocuções que serão processadas e, por conseguinte, o espaço de memória ocupado por algumas variáveis e o custo computacional do procedimento de *backtracking*, acrescenta-se um sistema de detecção de fala e silêncio. Quando aplicada à

detecção dos instantes de silêncio das elocuções utilizadas em simulação, ocorre uma pequena melhora na taxa de acerto do reconhecedor [11]. Isto conta como mais um fator positivo para a sua aplicação em tempo real.

Refazendo a lógica do sistema de reconhecimento do capítulo 4, obtém-se a estrutura apresentada na Figura 5.1.

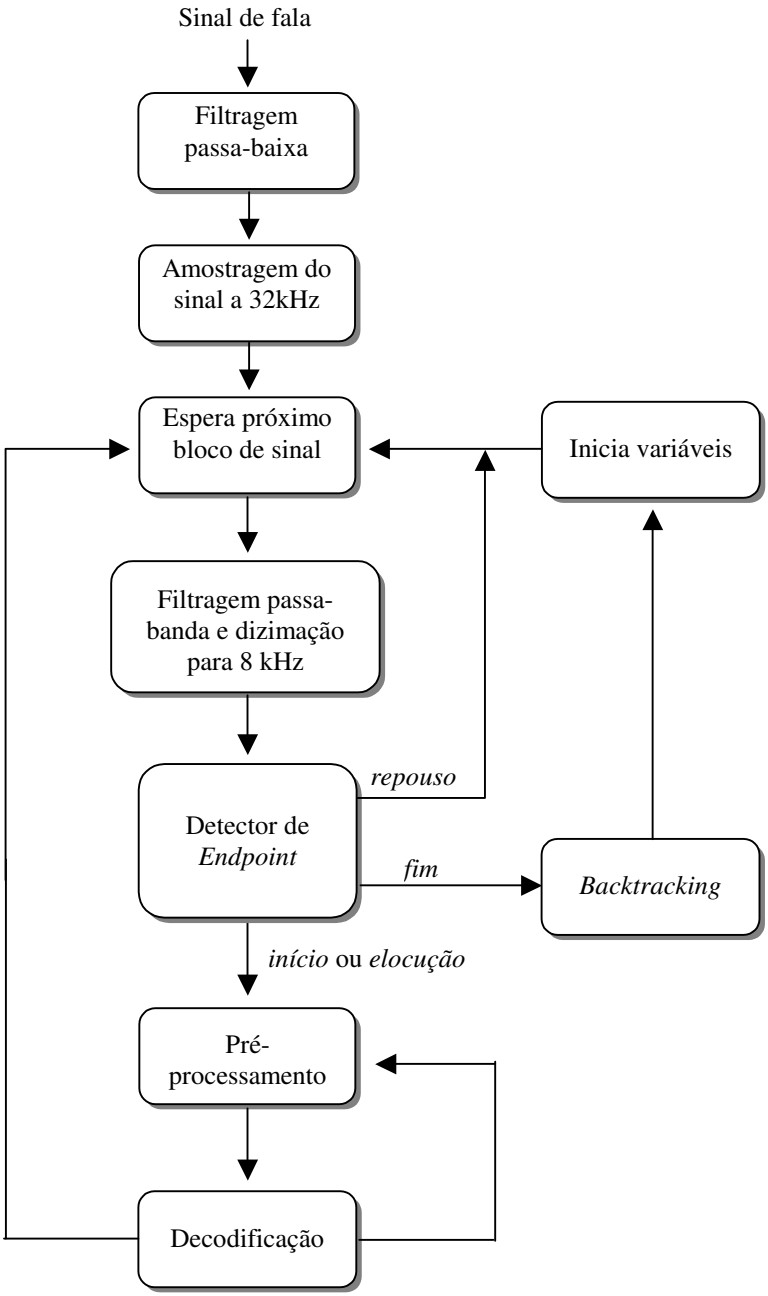

Figura 5.1 Diagrama de blocos do sistema de reconhecimento de fala

O sinal de fala é captado por um microfone de eletreto, e inserido na entrada da placa de som do PC. A filtragem passa-baixa a 16 kHz é o primeiro processamento realizado sobre o sinal, objetivando retirar o conteúdo em freqüência superior à freqüência de Nyquist. Após a filtragem passa-baixa, o sinal é amostrado a 32 kHz. As amostras do sinal são representadas por um conjunto limitado de bits. Nesta aplicação, a representação de cada amostra do sinal é em 16 bits. As amostras do sinal são agrupadas em janelas de 20 ms, cuja atualização se dá a cada 10 ms. Deve-se ressaltar que o período de 10 ms entre cada quadro é o intervalo de tempo que o sistema tem para realizar todas os processamentos necessários. Entretanto, realiza-se o armazenamento de 200 ms de sinal, que corresponde a 20 quadros, antes da filtragem passa-banda. Esse armazenamento é justificado pelo uso do detector de *endpoint*.

Depois de completado um bloco de 200 ms do sinal, realiza-se a filtragem passa-banda (300-3400 Hz), e, em seguida, a dizimação para a freqüência de 8 kHz. As características do filtro passa-banda são apresentas no capítulo 2. Processar o filtro passa-banda em 32 kHz ao invés de 8 kHz melhora a qualidade da filtragem, apesar de um pequeno acréscimo no custo computacional.

Terminado os processamentos iniciais, o bloco de 200 ms do sinal é enviado para o detector de *endpoint*. O detector de *endpoint* controla os estados do sistema. Considerando uma situação prática, enquanto o locutor não pronuncia nada, o sistema se encontra no estado de *repouso*. Detectado o início de uma frase, o sistema passa para o estado *início*. Durante o processamento da frase, o sistema permanece no estado *elocução*. Terminada a frase, o sistema entra no estado *fim*.

Os estados do detector de *endpoint* descrevem diferentes caminhos pelo diagrama de blocos da Figura 5.1. O caminho indicado pelos estados *início* e *elocução*, os quais indicam a ocorrência da fala, leva ao bloco de pré-processamento (análise espectral e quantização vetorial) e decodificação.

O pré-processamento não é muito diferente do que já foi apresentado no capítulo 2. Assim, são extraídos 39 coeficientes, sendo 12 coeficientes mel cepstrais, 12 coeficientes delta mel cepstrais e 12 coeficientes delta-delta mel cepstrais, e os 3 restantes representam a energia, a delta energia e a delta-delta energia. Deve-se estar atento ao fato de que a geração dos parâmetros do tipo delta e delta-delta requer um atraso. O atraso está relacionado com o número de janelas de

tempo necessárias para a geração desses parâmetros e se refletirá no término do processamento da elocução. Assim, o sistema indicará o resultado do reconhecimento com um atraso de 10 quadros depois de terminada a elocução. Esse atraso equivale ao número de quadros para o processamento dos parâmetros delta somado ao número de quadros para o processamento dos parâmetros delta-delta. Vale ressaltar que esse pequeno atraso é somado ao atraso de recebimento de um bloco do sinal.

O bloco de quantização vetorial vem logo em seguida. Tomam-se os 39 coeficientes extraídos de cada quadro de 20 ms do sinal de fala, e gera-se um vetor de 6 índices. Esses índices representam os vetores dos *codebooks*, sendo um índice para cada tipo de parâmetro.

Finalmente, o sistema está pronto para iniciar o processo de identificação da seqüência de palavras a partir do vetor de observações resultante da quantização vetorial. Sabendo-se que o tempo de duração da frase que está sendo pronunciada pelo locutor é desconhecido para o sistema, deve-se estar atento ao algoritmo de decodificação utilizado. Além disso, há a necessidade de se decodificar a seqüência de observação em sincronismo com o tempo. Nessas condições, o algoritmo de decodificação *One Step* é o mais adequado.

O sistema permanece no laço mais interno (*pré-processamento* e *decodificação*), enquanto estiverem sendo processados os quadros de um bloco de sinal (200 ms). Terminado o processamento do bloco, um novo bloco de sinal é esperado e os processos se repetem até que seja detectado silêncio. Assim, o detector de *endpoint* passa para o estado *fim*, e o sistema executa o procedimento de *backtracking*, que é a busca pela seqüência mais provável até o quadro atual. Apesar de não constar na Figura 5.1, o procedimento de *backtracking* é executado após um atraso equivalente ao processamento de 10 quadros. Esse atraso, já comentado anteriormente, é necessário para o processamento dos parâmetros delta e delta-delta.

Finalizado o *backtracking*, as variáveis temporárias do sistema são reiniciadas e o sistema retorna ao estado *repouso*.

5.4.1 Detector de *Endpoint*

O detector de *endpoint* desempenha algumas funções fundamentais ao processamento em tempo real. Entre elas podemos destacar:

- Remoção de períodos longos de silêncio;

- Redução no processamento;
- Descartar falsas elocuções.

O processo de detecção é feito a partir de duas medidas: taxa de cruzamento de zeros e energia. Essas informações são extraídas a cada 10 ms de um bloco *k* de sinal de 200 ms. Portanto, para cada bloco de sinal recebido, geram-se dois vetores de comprimento 20 para armazenar a taxa de cruzamento de zeros e a energia. Em seguida, são gerados os limiares de detecção. Comparando-se os limiares com a taxa de cruzamento de zeros e a energia, determinam-se pontos limitantes, que são as marcações do quadro inicial $q_i(k)$ e final $q_f(k)$ dentro do bloco *k* de sinal de 200 ms. O digrama de blocos da Figura 5.2 mostra o processo de detecção.

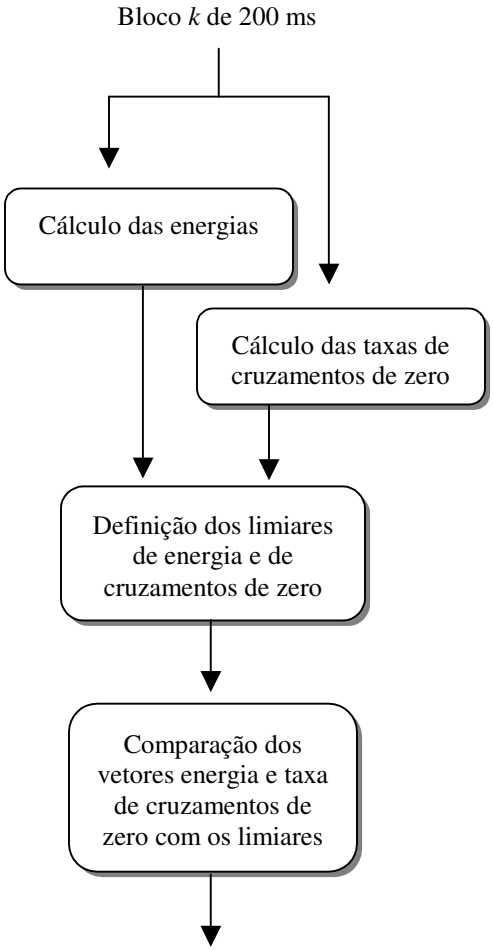

Figura 5.2 Diagrama de blocos do detector de *endpoint*

Há a necessidade de se adaptar os limiares às condições de operação. As condições de operação são o reflexo do equipamento utilizado e o ambiente de trabalho. Por isso, uma vez ativado o sistema de reconhecimento, são extraídos do primeiro bloco de 100 ms de sinal, a taxa de cruzamentos de zero e a energia do ruído. Esses valores são refletidos para um período de 10 ms de sinal, dividindo-se por 10 a taxa de cruzamentos de zero e a energia. O conhecimento dessas variáveis é fundamental para a geração dos limiares do detector.

5.4.2 Limitações do Detector de *Endpoint*

O detector de *endpoint* pode provocar erros de detecção. Naturalmente, sabe-se que um período de silêncio posterior a uma elocução indica o término do processamento e o início do processo de *backtracking*. Não se pode afirmar com certeza que qualquer período de silêncio detectado durante o processamento de uma palavra seja o fim da elocução. A ocorrência de consoantes oclusivas numa palavra gera um pequeno período de oclusão. Isso possibilita uma falsa detecção de silêncio e, por esse motivo, o algoritmo necessita de uma certa inteligência para tomar a decisão correta.

A Figura 5.3 mostra uma situação típica onde uma falsa detecção causará uma possível falha no reconhecimento.

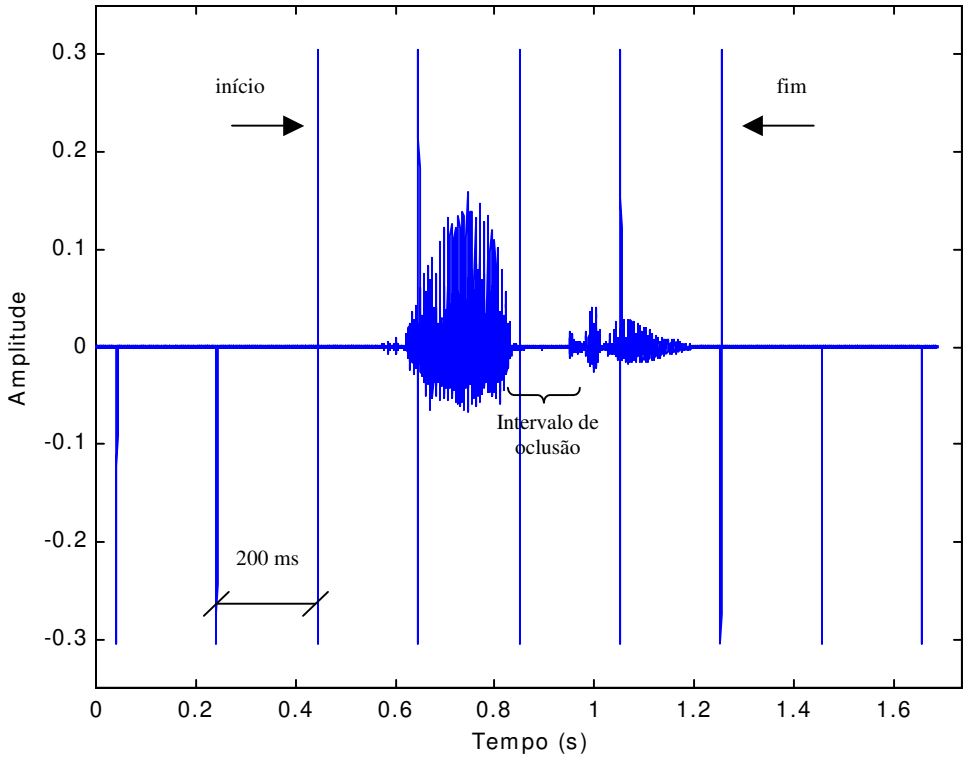

Figura 5.3 Resultado da detecção da palavra “quatro”.

As barras verticais no sinal indicam os intervalos de 200 ms de cada bloco do sinal. As barras indicadas pelas setas foram inseridas pelo detector de *endpoint*. O pequeno período de silêncio é detectado pelo algoritmo de *endpoint*, mas não foi removido para evitar a separação das sílabas da palavra “quatro”. Essa situação se soma a outras pequenas situações, as quais o detector deve prever. Por isso, foi adicionada ao detector de *endpoint* a máquina de estados da Figura 5.4. Os estados são alcançados depois de respeitadas algumas condições, as quais foram ajustadas experimentalmente.

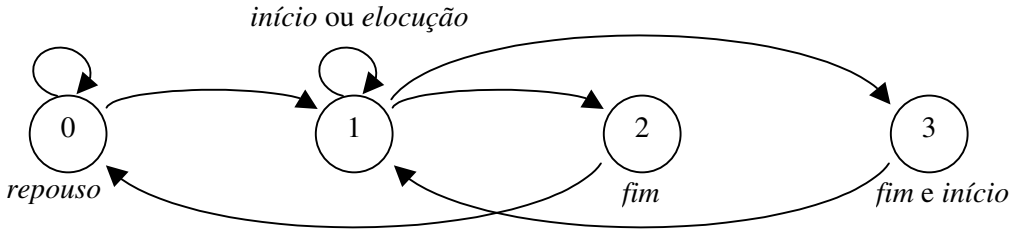

Figura 5.4 Máquina de 4 estados do detector de *endpoint*

No estado 0, o sistema está em repouso. Nenhum processamento é realizado no sinal, pois existe somente silêncio. Enquanto isso, o sistema está esperando uma nova elocução.

O estado 1 é atingido quando o início da elocução for detectado. Entretanto, falsas elocuções podem ser descartadas, caso a diferença entre o quadro inicial $q_i(k)$ e o quadro final $q_f(k)$ seja pequena, ou seja,

$$q_f(k) - q_i(k) \leq 5, \tag{5.1}$$

onde *k* é o índice do bloco atual. Por outro lado, se o início da elocução for confirmada, o sistema inicia o processamento do sinal a partir do quadro $q_i(k)$ até o último quadro do bloco. Nesta condição, o valor de $q_f(k)$ é desconsiderado, pois não é certo que a elocução tenha terminado Considerando a confirmação de uma nova elocução, a permanência neste estado para os blocos seguintes depende de duas condições:

$$20 - q_f(k) < 10, \tag{5.2}$$

$$q_f(k-1) + q_i(k) < 10. \tag{5.3}$$

Se verificadas as condições, o sistema inicia o processamento do sinal a partir do quadro 1 até o último quadro do bloco. Entretanto, se a condição da equação (5.2) não for satisfeita, transita-se para o estado 2. Por outro lado, se a condição da equação (5.3) não for satisfeita, transita-se para o estado 3.

O estado 2 é atingido quando for detectado o fim da elocução. Assim, processa-se somente os $q_f(k)$ primeiros quadros, inicia-se o procedimento de *backtracking* e retorna-se ao estado 0.

O estado 3 é atingido quando detectado o fim da elocução e o início de uma nova em um mesmo bloco. Esse estado representa um caso especial de detecção e ocorre somente nas seguintes situações:

- Pausa detectada no bloco anterior *k*-1 e continuação no bloco atual *k*, superando 100 ms ou 10 quadros.
- Possibilidade de se estar iniciando uma nova elocução a partir do quadro $q_i(k)$.

Assim, processa-se o sinal a partir do quadro $q_i(k)$ até o último quadro do bloco *k*, e retorna-se ao estado 1.

5.4.3 Custo Computacional

Depois da adaptação do algoritmo para processamento em tempo real e da inclusão do detector de silêncio, foram realizados testes para a medição do custo computacional em função do tempo de processamento. Considerando a implementação do sistema num microcomputador com processador Pentium III, a 500 MHz, obteve-se um gasto de 110 ms para o processamento do bloco de 200 ms. Isso garante a operação em tempo real com uma folga de 90 ms.

5.5 Teste do Sistema de Reconhecimento

Selecionaram-se oito locutores adultos do sexo masculino para o teste do sistema. Nenhum desses locutores faz parte da base de sinais. Foi solicitada a pronúncia de seqüências de dígitos com naturalidade. Todas as seqüências contêm oito dígitos e representam números de telefones escolhidos pelo próprio locutor. Cada locutor pronunciou quatro números de telefone distintos.

O ambiente de teste é o de escritório. Não foram tomados cuidados com ruídos de fundo. O microfone é de eletreto com suporte na cabeça.

5.5.1 Configurações do Sistema de Reconhecimento

As configurações do sistema estão relacionadas com as características do bloco de pré-processamento e decodificação:

1) Pré-processamento

1.1)Análise Espectral

- Filtragem de pré-ênfase com fator 0.95;
- Aplicação de janelas de Hamming de 20 ms e superposição entre janelas de 50%;
- Extração de 12 coeficientes mel cepstrais e energia, e mais suas derivadas primeira e segunda, resultando num vetor de parâmetros de tamanho 39.

1.2) Quantização Vetorial

- 6 *codebooks* independentes, um para cada parâmetro;

• Foram utilizados *codebooks* de 256 vetores para os parâmetros mel cepstrais e de 32 vetores para as energias.

2) Decodificação

• Foram utilizados os modelos ocultos de Markov obtidos no Experimento II da base LPDF Dígitos;

• Não foi utilizado o modelo de duração de palavra;

• A busca feita pelo algoritmo de decodificação simulou frases com número de dígitos variando de um a dez.

5.5.2 Resultados

Os resultados são apresentados por locutor. Cada locutor é identificado por sua cidade de origem. Além disso, o sistema desconhece o número de dígitos que será falado.

Locutor 1 – Brasília/DF

Frase Pronunciada	Frase reconhecida
3 2 3 5 2 3 8 9	3 2 3 5 2 3 8 9
3 2 8 7 5 7 4 7	3 2 8 7 5 7 4 7
3 2 4 8 0 2 0 7	3 2 4 8 0 2 0 7
9 m[2] 0 2 4 4 7 3	9 m 0 2 4 4 7 **6**

Locutor 2 – Rio de Janeiro/RJ

Frase Pronunciada	Frase reconhecida
3 2 7 3 1 4 0 8	**6** 2 7 3 1 4 **6** 8
3 3 5 0 8 8 0 m	**6 6** 5 0 8 8 0 m
9 1 1 m 7 5 7 m	9 1 1 m 7 5 7 m
3 2 8 7 m m 2 1	3 2 8 7 m m 2 1

[2] A letra "m" equivale ao dígito meia.

Locutor 3 – Receife/PE

Frase Pronunciada	Frase reconhecida
3 2 0 9 0 9 m 5	3 2 0 9 0 9 m 5
3 3 2 m 0 m 7 m	3 3 2 m 0 m 7 m
3 2 0 8 2 3 9 8	3 2 0 8 2 3 9 8
3 4 2 1 7 3 1 2	**7** 4 2 1 7 3 1 2

Locutor 4 – Goiânia/GO

Frase Pronunciada	Frase reconhecida
3 2 3 3 3 5 1 5	**6** 2 3 3 3 5 1 5
3 2 4 1 m 7 9 1	3 2 4 1 m 7 9 1
3 2 4 2 0 3 m 2	3 2 4 2 0 3 m 2
3 4 2 1 5 9 2 5	3 4 2 1 5 9 2 5

Locutor 5 – Londrina/PR

Frase Pronunciada	Frase reconhecida
3 2 3 7 4 2 9 2	3 2 3 7 4 2 9 2
3 2 3 7 4 1 7 1	3 2 3 7 4 1 7 1 **0**
9 9 9 5 0 m 1 5	9 9 9 5 0 m 1 5
9 9 1 2 2 7 m 7	9 9 1 2 2 7 m 7

Locutor 6 – Fortaleza/CE

Frase Pronunciada	Frase reconhecida
3 2 0 8 1 m 2 0	3 2 0 8 1 m 2 0
3 2 0 5 7 m 5 3	3 2 0 5 7 m 5 3
3 1 5 9 0 5 3 3	3 1 5 9 0 5 3 3
3 5 2 8 7 9 5 4	3 5 2 8 7 9 5 4

Locutor 7 – Rio de Janeiro/RJ

Frase Pronunciada	Frase reconhecida
3 2 5 m 8 0 3 5	3 2 5 m 8 0 3 5
3 2 3 3 8 1 0 m	3 2 3 3 8 1 0 m
3 2 5 1 7 7 9 9	**6** 2 5 1 7 7 9 9
3 2 5 4 4 3 4 5	3 2 5 4 4 3 4 5

Locutor 8 – São João da Boa Vista/SP

Frase Pronunciada	Frase reconhecida
3 2 5 8 5 4 8 8	**6** 2 5 8 5 4 8 8
3 7 8 8 3 8 1 8	3 7 8 8 3 8 1 8
9 1 1 2 7 9 2 7	9 1 1 2 7 9 2 7
3 2 5 8 3 3 2 5	**6** 2 5 8 3 3 2 5

A partir dos testes realizados, o sistema apresentou uma taxa de acerto de palavras de 95,70%. Os resultados mostram que o sistema respondeu razoavelmente bem ao teste, levando-se em conta a independência de locutor, as condições de teste e a característica da base de sinais utilizado no treinamento dos HMM's. Vale ressaltar que os locutores da base de treinamento são, em sua maioria, da região de São Paulo [21], e os locutores utilizados no teste são de diferentes regiões do país.

Entre os erros encontrados, o mais significativo foi a troca do dígito três pelo dígito seis, representando 82% dos erros. Isso se repetiu até quando foi pedido para o locutor pronunciar isoladamente esse dígito. Necessita-se, portanto, investigar um pouco mais o processo de geração do modelo para o dígito três.

Capítulo 6
CONCLUSÃO

O trabalho levantou várias questões e algumas delas são propostas aqui como trabalhos futuros. Inicialmente, necessitou-se avaliar o desempenho do sistema, como também as alternativas presentes na literatura para torná-lo consistente. A escolha das características do sistema, incluindo a ordem dos vetores de parâmetros, o tamanho dos *codebooks* e o número de estados de cada modelo ocultos de Markov, é resultado de testes exaustivos. Alguns autores propõem até mudanças na análise espectral, variando-se a largura dos filtros passa-banda e acrescentando-se outros filtros, visando abordar mais detalhadamente as características do sinal de fala e melhorar o desempenho do sistema. Entretanto, neste trabalho, o enfoque foi voltado para os procedimentos de treinamento dos modelos ocultos de Markov. A escolha das características do sistema ficou em segundo plano, e por isso, optou-se por tomar resultados de outros trabalhos cujo enfoque foi definir essas características. Através de refinamentos futuros, pode-se conseguir configurações mais adequadas que proporcionem resultados ainda melhores.

O procedimento de treinamento dos modelos ocultos de Markov tem como base a proposta de treinamento apresentada em [18]. No entanto, os modelos ocultos de Markov contínuos usados em [18] foram substituídos por modelos ocultos de Markov discretos. Os primeiros resultados obtidos estiveram aquém do apresentado no trabalho na seqüência, e por isso, buscaram-se alternativas para melhorar o desempenho do sistema. A primeira delas foi aumentar o número de estados dos modelos. Algumas referências propõem modelos com mais de 20 estados. Os testes foram feitos com modelos de até 15 estados, que foi o número fixado, apresentando melhora nos resultados. A segunda alternativa foi estudar as características das bases de sinais utilizadas. Constatou-se a necessidade de um modelo adicional para o silêncio

devido à ocorrência freqüente de silêncio em todas as elocuções da base. Essa alternativa, somada à anterior, mostrou um aumento significativo no desempenho do sistema.

Durante os repetidos procedimentos de treinamento, levantou-se uma questão sobre o procedimento de iniciação. A dúvida está em saber se independentemente do procedimento de iniciação, o treinamento dos modelos convergirá para o mesmo "ponto". A iniciação dos modelos foi inicialmente realizada a partir de elocuções da base TI *Digits* contendo apenas um dígito. Em seguida, optou-se por segmentar manualmente algumas elocuções para a retirada do silêncio, e formou-se um pequeno conjunto de treinamento de palavras isoladas. Esse conjunto foi utilizado na iniciação dos modelos para o treinamento de palavras conectadas. Essa iniciação gerou os melhores resultados deste trabalho com a base TI *Digits*. Constata-se a necessidade de se marcar os instantes de silêncio de toda a base TI *Digits* para a realização do procedimento de treinamento de palavras conectadas, para a qual se espera um aumento no desempenho do sistema.

Procedimentos de treinamento mais complexos, que não foram abordados neste trabalho, são propostos na literatura [14]. Sugere-se acrescentar, como um refinamento do treinamento realizado neste trabalho, o treinamento discriminativo [6].

Outra alternativa de melhora no desempenho do sistema é a adição de modelos de duração [18]. Verificou-se que o modelo de duração de palavra aumenta a taxa de acerto. Entretanto, uma modelagem de duração de estados em conjunto com a modelagem de duração de palavras foi descartada nesse trabalho. Necessita-se, portanto, pesquisar um pouco mais a modelagem de duração de estados. Vale notar que os modelos de duração tornam os modelos ocultos de Markov mais consistentes, pois carregam informações adicionais de duração.

Os testes realizados com a base TI *Digits* também foram feitos com uma base de sinais em português, que foi chamada de LPDF Dígitos. Observou-se que as mesmas alternativas propostas trazem aumentos de desempenho sistêmico para essa segunda base. Entretanto, os modelos de duração não provocaram melhorias de desempenho. Vale ressaltar que a base LPDF Dígitos é pequena e possui somente seqüências de 8 dígitos.

Com relação à implementação em tempo real, buscou-se o compromisso de baixo custo computacional e baixa taxa de erro. Entretanto, o sistema de reconhecimento de fala sofre uma queda significativa de desempenho quando testado num ambiente diferente do ambiente de simulação. Essa queda de desempenho é levantada na literatura, mas ainda necessita-se reduzir a

grande diferença entre os resultados de simulações e de teste em tempo real. Pode-se apontar a qualidade do equipamento de aquisição do sinal de fala, microfone e placa de áudio, como um dos pré-requisitos. Além disso, necessita-se prever situações comuns à utilização do sistema como hesitações do locutor, sopros entre outras. Isso pode ser feito, adicionando-se modelos para cada uma dessas situações.

Uma outra proposta de trabalho futuro é refazer todos os testes apresentados nesse trabalho utilizando modelos ocultos de Markov contínuos. Isso exige grande esforço computacional para a implementação em tempo real.

Enfim, o presente trabalho representa o básico da tecnologia de reconhecimento de fala aplicada a uma situação próxima da real. A implementação de um sistema de reconhecimento de fala numa situação mais realista apresenta grandes desafios, exigindo novas pesquisas que poderão torná-lo mais realista e consistente.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] B. S. Atal e L. R. Rabiner, "A Pattern Recognition Approach to Voiced-Unvoiced-Silence Classification with Applications to Speech Recognition", IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-14, no. 3, Junho 1976.

[2] J.K. Baker, "The dragon system – An overview", IEEE Trans. Acoust. Speech Signal Processing, vol. ASSP-23, no. 1, pp. 24-29, Fevereiro 1975.

[3] L.E. Baum, e T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains", Ann. Math. Stat., vol. 37, pp. 1554-1563, 1966.

[4] R. Comerford, J. Makhoul e R. Shwartz, "The Voice of the Computer is Heard in the Land (and It Listens Too!)", IEEE Spectrum, Dezembro 1997.

[5] N. Deshmukh, A. Ganapathiraju e J. Picone, "Hierarchical Search for Large-Vocabulary Conversational Speech Recognition", IEEE Signal Processing Magazine, Setembro 1999.

[6] F.L. Figueiredo, "Segmentação Automática e Treinamento Discriminativo Aplicados a um Sistema de Reconhecimento de Dígitos Conectados", Dissertação de Mestrado, UNICAMP, Campinas, 1999.

[7] A. Gersho e R.M. Gray, "Vector Quantization and Signal Compression", Kluwer Academic Publishers, 1992.

[8] K. Lee, "Automatic Speech Recognition", Kluwer Academic Publishers,1989.

[9] C. Lee e L.R. Rabiner, "A Frame-Synchronous Network Search Algorithm for Connected Word Recognition", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 37, pp. 1649-1658, Novembro 1989.

[10] R.G. Leonard, "A database for speaker-independent digit recognition", Proceedings of the ICASSP - 84 , pp. 42.11.1-4, Março 1984.

[11] J. A. Martins, "Avaliação de Diferentes Técnicas para Reconhecimento de Fala", Tese de Doutorado, UNICAMP, Campinas, Dezembro 1997.

[12] L.G.P. Meloni, "Learning Discrete Hidden Markov Models", Computer Applications in Engineering Educat., vol. 8, no. 3, pp. 141-149, 2000.

[13] L.G.P. Meloni, "Learning Discrete Hidden Markov Models", http://www.decom.fee.unicamp.br/~meloni/educat/learnHMM.htm.

[14] Y. Normandin, "Hidden Markov Models, Maximum Mutual Information Estimation, and the Speech Recognition Problem", PhD Thesis, McGill University, Montreal, Março 1991.

[15] J. Picone, “Signal Modeling Techniques in Speech Recognition”, Proceedings of the IEEE, 81(9):1215-1247, Setembro 1993

[16] L. R. Rabiner e S. E. Levinson, “Isolated and Connected Word Recognition – Theory and Selected Applications”, IEEE Transactions on Communications, vol. COM-29, no. 5, Maio 1981.

[17] L. R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”, Proceedings of the IEEE, vol. 77, no. 2, Fevereiro 1989.

[18] L.R. Rabiner, J.G. Wilpon e F.K. Soong, "High Performance Connected Digit Recognition Using Hidden Markov Models", IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, pp. 1214-1225, Agosto 1989.

[19] L.R. Rabiner e B.H. Juang, “Fundamentals of Speech Recognition”, Prentice Hall, 1993.

[20] W. Strauss, “Digital Signal Processing – The New Semiconductor Industry Technology Driver”, IEEE Signal Processing Magazine, Março 2000

[21] C. Ynoguti, “Reconhecimento de Fala Contínua Usando Modelos Ocultos de Markov”, Tese de Doutorado, UNICAMP, Campinas, Maio 1999.