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

ESTUDO E IMPLEMENTAÇÃO DE UM SISTEMA DE RECONHECIMENTO DE DÍGITOS CONECTADOS USANDO HMMs CONTÍNUOS

Autora: **Jaqueline Vieira Gonçalves**
Orientador: **Prof. Dr. Luís Geraldo Pedroso Meloni**

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: **Engenharia Elétrica**.

Banca Examinadora
Luís Geraldo Pedroso Meloni, Dr. ..... DECOM/FEEC/Unicamp
Rodrigo Varejão Andreão, Dr. ..... UFES
Plinio Almeida Barbosa, Dr. ..... IEL/Unicamp
Dalton Soares Arantes, Dr. ..... DECOM/FEEC/Unicamp

Campinas, SP
Abril/2005

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

Gonçalves, Jaqueline Vieira
G586e Estudo e implementação de um sistema de reconhecimento de dígitos conectados usando HMMs contínuos / Jaqueline Vieira Gonçalves. --Campinas, SP: [s.n.], 2005.

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

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

Título em Inglês: Study and implementation of a connected digit recognition system using continuous HMMs
Palavras-chave em Inglês: Hidden Markov models, Speech recognition, Automatic speech recognition
Área de concentração: Telecomunicações e Telemática
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Rodrigo Varejão Andreão, Plínio Almeida Barbosa e Dalton Soares Arantes.
Data da defesa: 19/04/2005

"As coisas podem chegar até aqueles que esperam, mas são somente as sobras deixadas por aqueles que lutam."

Abraham Lincoln

À minha querida e amada mãe, Fatima.

Agradecimentos

Primeiramente, agradeço a Deus em Quem confio e está sempre presente em minha vida, pois sem Ele não teria conseguido chegar até aqui.

Agradeço ao Professor Meloni pelo apoio e confiança.

A meu amado Alex que sempre confiou em mim, pelo apoio e amor incondicional, agradeço profundamente.

Agradeço a minha irmã Carol, que esteve comigo durante todos esses anos, por sua amizade e carinho.

Agradeço a minha família que mesmo tão longe sempre me apoiou e torceu por mim, em especial à tia Lúcia por sua grande ajuda e confiança e à tia Salma por suas palavras de incentivo e carinho.

Agradeço a minha querida avó Pureza por suas orações e carinho.

Agradeço à Lurdes, Assis, Fofa, Érica, Rodolfo e Edna que se tornaram minha família aqui em Campinas e aos quais serei sempre grata pelo carinho, atenção, acolhida e amizade.

À D. Amália e sua família agradeço de coração por suas orações, carinho e palavras de incentivo.

Agradeço também aos meus amigos Helder, Lívio, Kiki, Betânia, Kelly, Marizinha, Valéria, Sérgio, Glauco, Edmilson, Fábio, Euler, Lucas, Paulo, Rodrigo, Márzio e Leo Cabelo, companheiros de todos os momentos e que contribuíram de forma tão especial, cada um a seu modo, para a conclusão deste trabalho.

Quero agradecer a todos os locutores que me ajudaram na gravação da base de dados de dígitos isolados, em especial à Fernanda Atizani pela coleta e tratamento dos dados.

Finalmente, agradeço ao Fundo de Apoio a Pesquisa do Estado de São Paulo (FAPESP) pelo apoio financeiro.

Resumo

Neste trabalho, Modelos Ocultos de Markov Contínuos (HMMC) baseados em palavras e independentes de locutor são incorporados a um sistema de reconhecimento de dígitos conectados baseado em HMMs discretos do Laboratório de Processamento Digital de Sinais de Multimídia em Tempo Real da Faculdade de Engenharia Elétrica da UNICAMP, visando aperfeiçoar a plataforma existente. A teoria envolvida e detalhes da implementação do sistema de modelos contínuos são apresentados. Os HMMs contínuos empregados durante os experimentos possuem quantidades de estados e misturas dependentes do comprimento da palavra e, assim como no sistema anterior, o processo de treinamento usa um conjunto treinado de dígitos isolados como modelos iniciais no treinamento de dígitos conectados, além da informação adicional de duração de palavra. Durante esta fase de treinamento dos dígitos conectados, também é realizada outra forma de treinamento em que os modelos de dígitos isolados não são usados. As taxas de reconhecimento obtidas com esses dois tipos de treinamento também são avaliadas. Duas bases de dados foram usadas na análise de desempenho do sistema, uma delas em Português brasileiro e outra no Inglês americano. Os experimentos realizados permitiram comparar o desempenho entre os dois tipos de modelos, discreto e contínuo, para esta aplicação de modelos de palavras independentes de locutor, bem como apresentam resultados entre o sistema desenvolvido com HMMs contínuos e o *software* livre HTK (*HMM Toolkit*) sob as mesmas condições de operação. Experimentos também mostram o comportamento do sistema de HMMs contínuos desenvolvido ao variar-se o número de estados e misturas dos modelos separadamente.

Palavras-Chave: Modelos Ocultos de Markov, Dígitos Conectados, Reconhecimento de Fala, HTK.

Abstract

In this work, we incorporate a continuous density Hidden Markov Models (HMMC) to a connected digit speech recognition system, based on speaker-independent word models, of the Real Time Multimedia Digital Signal Processing Laboratory at UNICAMP. The previous system is based on discrete HMMs, and the involved theory and implementation details of the continuous model system are presented. The continuous HMMs used in our experiments have the amount of states and mixtures dependent on word length. As well as in the previous system, the training procedure uses a training set of isolated digits in order to provide initial estimates of the continuous models and it also includes additional information of word duration. Moreover, we have also used another training procedure in which the isolated digits models are not used. The recognition rates obtained with those two training forms are also evaluated. Two databases were used to assess system performance, one is a small database for the Brazilian Portuguese and another one is for the American English. We carried out experiments in order to compare the performance of two types of models, discrete and continuous, in a speaker-independent word model application. We also evaluated the continuous HMMs performance using the open source HTK (HMM Toolkit) under the same operation conditions. Finally, performance results of the developed continuous HMMs system for different number of states and Gaussian mixtures are also shown.

Keywords: Hidden Markov Models, Connected Digits, Speech Recognition, HTK.

Índice

Glossário

CUED	*Cambridge University Engineering Department*
DCT	*Discrete Cosine Transform*
DNA	*Deoxyribonucleic Acid*
EBNF	*Extended Backus Naur Form*
FTM	*Fully Tied Mixture*
HMM	Modelo Oculto de Markov
HMMC	Modelo Oculto de Markov Contínuo
HMMD	Modelo Oculto de Markov Discreto
HTK	HMM *Toolkit*
LBG	Algoritmo de Linde, Buzo e Gray
LPDF	Laboratório de Processamento Digital de Fala
MFCCs	*Mel Frequency Cepstral Coefficients*
MLF	*Master Label Files*
MMF	*Master Macro File*
PDF	Função Densidade de Probabilidade
Perl	*Practical Extraction and Report Language*
RT$^{\underline{M}}$DSP	Laboratório de Processamento Digital de Sinais de Multimídia em Tempo Real
SLF	*Standard Lattice Format*
SNR	Relação Sinal-Ruído
TIDIGITS	*Texas Instruments Digits*
WSJ	*Wall Street Journal Dictation*

Lista de Símbolos

$H_{pb}(z)$	Função de transferência do filtro passa-banda
$H_{pe}(z)$	Função de transferência do filtro de pré-ênfase
μ_{pe}	Fator de pré-ênfase
$h(n)$	função que define a Janela de Hamming
n	Índice da amostra da janela de Hamming ou índice do coeficiente mel cepstral
ξ	Número total de amostras da janela de Hamming
p	Índice do quadro do sinal acústico
$c_p(n)$	n-ésimo coeficiente mel cepstral pertencente ao p-ésimo quadro do sinal
f	Índice do filtro
$E(f)$	Energia de saída do f-ésimo filtro
N_{tot}	Número total de filtros
$\Im$	Número total de coeficientes mel cepstrais
$D\dot{c}_p(n)$	Derivada primeira do n-ésimo coeficiente mel cepstral do quadro p
F	Número de quadros utilizados no cálculo das derivadas
$D\ddot{c}_p(n)$	Derivada segunda do n-ésimo coeficiente mel cepstral do quadro p
Y_g	Variável aleatória
i	Índice do estado
j	Índice do estado
S	Seqüência de estados
N	Número total de estados de um modelo
$\mathbf{A}$	Matriz de probabilidade de transição de estados
a_{ij}	Coeficiente da matriz de probabilidade de transição de estados
t	Instante de tempo

s_t	Estado no instante t
$\mathbf{O}$	Seqüência de observações
$\mathbf{o}_t$	Vetor de observação no instante t
T	Tamanho da seqüência
k	Índice do símbolo observado
K	Quantidade de símbolos
$\mathbf{B}$	Matriz de probabilidade de emissão de símbolo
$b_i(\mathbf{o}_t)$	Coeficiente da matriz de emissão de símbolo
$\boldsymbol{\pi}$	Vetor de probabilidade de estado inicial
λ	Modelo HMM
$\alpha_t(i)$	Variável *Forward*
$\beta_t(i)$	Variável *Backward*
$\delta_t(i)$	Maior probabilidade ao longo de uma dada seqüência de estados
$\varphi_t(j)$	Variável que armazena os estados que maximizam a equação 2.24
s_t^*	Melhor seqüência de estados ou seqüência ótima do algoritmo de Viterbi
$\widetilde{b}_i(\mathbf{o}_t)$	Forma logarítmica de $b_i(\mathbf{o}_t)$
$\widetilde{\pi}_i$	Forma logarítmica de π_i
$\widetilde{a}_{ij}$	Forma logarítmica de a_{ij}
$\widetilde{\delta}_t(j)$	Forma logarítmica de $\delta_t(j)$
$\overline{a}_{ij}$	Coeficiente re-estimado da matriz de probabilidade de transição de estados
$\mathbf{v}_k$	Símbolo emitido
$\overline{b}_i(k)$	Coeficiente re-estimado da matriz de emissão de símbolo
m	Índice da mistura ou região
r	Índice da mistura ou região
c_{jm}	Coeficiente de peso da m-ésima mistura do estado j
M	Número total de misturas ou regiões de um estado
$\mathcal{F}(\cdot)$	Qualquer densidade elipticamente simétrica ou log-côncava
$\boldsymbol{\mu}_{jm}$	Vetor média da m-ésima mistura do estado j
$\mathbf{U}_{jm}$	Matriz covariância da m-ésima mistura do estado j
$\overline{c}_{jm}$	Coeficiente de peso re-estimado da m-ésima mistura do estado j
$\overline{\boldsymbol{\mu}}_{jm}$	Vetor média re-estimado da m-ésima mistura do estado j

$\gamma_t(j,m)$	Probabilidade. de estar no estado j no instante t com a m-ésima componente de mistura associada a observação $\mathbf{o}_t$
$\overline{\mathbf{U}}_{jm}$	Matriz covariância re-estimada da m-ésima mistura do estado j
$\widehat{c}_t$	Coeficiente de normalização independente do estado
$\widehat{\alpha}_t(i)$	Variável *Forward* escalonada
$\widehat{\beta}_t(i)$	Variável *Backward* escalonada
$\widetilde{\alpha}_t(i)$	Variável *Forward* atualizada a partir da variável *Forward* escalonada
$\widetilde{\beta}_t(i)$	Variável *Backward* atualizada a partir da variável *Backward* escalonada
d	Índice da elocução ou da seqüência de observações
D	Número total de elocuções
w	Dígito
$\breve{\mathbf{x}}_{im}$	Vetor centróide pertencente a região m do estado i
$[\breve{x}_q]^+_{i\rho}$	Componente do vetor centróide acrescido de ϵ
$[\breve{x}_q]^-_{i\rho}$	Componente do vetor centróide decrescido de ϵ
ϵ	Variável usada na criação de novos vetores média para formar novas regiões
ρ	Número inteiro que varia de 1 até o tamanho corrente final do *codebook*
η	Dimensão do vetor centróide ou vetor média, no caso, *trinta e nove*
v	Índice do vetor
σ^2_v	Componente v do vetor variância da região m
x_v	Componente v de um vetor de parâmetros pertencente a região m
$\breve{x}_v$	v-ésimo componente do vetor média (centróide) da região m
N_m	Número total de vetores classificados na região m
$G(\cdot)$	Função densidade de probabilidade Gaussiana multidimensional
$\bar{\lambda}_w$	Modelo ou *template* re-estimado do dígito w
P^w_{new}	Probabilidade média corrente
P^w_{old}	Probabilidade média anterior
Γ^w_{new}	Distância corrente entre o modelo atualizado e o modelo anterior
$P_w(d_w)$	Probabilidade de duração do dígito w
d_w	Duração da palavra w em número de quadros
σ^2_w	Variância da duração da palavra w
μ_w	Duração média da palavra w

T_a Taxa de acerto em %

N_t Número total de exemplos

Capítulo 1

Introdução

Em meados de 1950, cientistas da AT&T projetaram o primeiro sistema de reconhecimento de fala para dígitos isolados de zero a nove, na língua inglesa, e adaptado para um único locutor. Os sistemas evoluíram e com o avanço da tecnologia tornaram-se mais eficazes. O uso de técnicas de programação dinâmica e métodos estatísticos mais eficientes, tais como Modelos Ocultos de Markov e Redes Neurais, permitiram o desenvolvimento de sistemas com melhor desempenho, incluindo também reconhecimento de fala contínua independente de locutor.

Dentro deste contexto, sistemas de reconhecimento automático de dígitos conectados independentes de locutor também têm sido alvo de muitas pesquisas, com o propósito de desenvolver produtos de alto desempenho, devido a sua grande empregabilidade no mundo real. Tais aplicações englobam operadoras de cartões de crédito, discagem através da fala, validação de contas por meio da fala em sistemas bancários e outros. A independência de locutor é de fundamental importância sobretudo em aplicações em que uma base de dados específica para cada locutor não existe ou não está disponível, como é o caso da maioria dos sistemas. O problema de sistemas independentes de locutor é que a taxa de erro, em relação a sistemas dependentes de locutor, é cerca de duas a três vezes maior [PAUL 89]. Uma boa modelagem durante a fase de treinamento torna-se portanto uma tarefa essencial.

Existem diversas formas de comparação de desempenho no reconhecimento de fala entre o homem e a máquina. A Tabela 1.1, adaptada de [HUAN 01], apresenta *cinco* resultados de desempenho no reconhecimento da fala contínua independente de locutor obtidos de experimentos realizados com o ser humano (Homem-Homem) e com o computador (Homem-

Máquina). O vocabulário empregado nos testes varia de 10 a 5000 palavras. Uma das bases de dados empregadas, o *Wall Street Journal Dictation* (*WSJ*), é composta por 5000 palavras em que os textos são extraídos de seus artigos na forma de ditados contínuos. As taxas de acerto obtidas pela máquina são baseadas no estado da arte de sistemas de reconhecimento de fala (2001) e as taxas obtidas pelo homem foram obtidas através de testes subjetivos semelhantes aos usados com o computador.

Tabela 1.1: Comparações da Taxa de Acerto de Palavras para o Homem e para a Máquina em Experimentos Semelhantes. (Adaptado de [HUAN 01]).

Experimentos	**Vocabulário**	**Humanos**	**Máquina**
Dígitos Conectados	10	$99,9\%$	$99,28\%$
Letras do Alfabeto	26	99%	95%
Fala Espontânea por Telefone	2000	$96,2\%$	$63,3\%$
WSJ sem Ruído	5000	$99,1\%$	$95,5\%$
WSJ com Ruído (SNR = 10 dB)	5000	$98,9\%$	$91,4\%$

Deduz-se pela Tabela 1.1 que o ser humano tem uma taxa de erro em relação às máquinas, ao menos *cinco* vezes menor, possuindo um desempenho muito mais robusto. Já a máquina tem seu desempenho significativamente prejudicado quando experimentos são realizados em ambientes com ruído e/ou com falta de informação espectral (telefone), não sendo observado o mesmo comportamento drástico com o ser humano.

Três diferentes formas de elocução das palavras podem ser consideradas quando se trata de sistemas de reconhecimento de fala. As palavras podem ser pronunciadas isoladamente, de forma conectada ou contínua. O sinal de fala representando uma palavra isolada é caracterizado por um período de silêncio mais longo antes e depois da produção acústica. Palavras conectadas são aquelas produzidas pausadamente com brevíssimos intervalos de silêncio entre elas ou não. A pronúncia de palavras conectadas é muito comum no dia a dia. Um exemplo prático é quando se produz uma seqüência de números ao passar um número de telefone ou de cartão de crédito. A fala contínua, como o próprio nome diz, é produzida de forma contínua como na fala corrente, por exemplo em diálogos ou na reprodução de textos por voz. Devido a suas características específicas de produção, cada tipo de sinal de fala precisa ser tratado de maneira diferente no reco-nhecimento automático de fala. Métodos de

processamento adequados a cada tipo de sinal de fala precisam ser utilizados para um bom desempenho no reconhecimento.

Uma das ferramentas mais utilizadas na modelagem de sinais de fala são os Modelos Ocultos de Markov (HMMs) [RABI 93]. Os HMMs podem ser discretos, contínuos ou ainda semi-contínuos. O Modelo Oculto de Markov é uma variação da Cadeia de Markov e é caracterizado por um certo número de estados conectados por transições probabilísticas cujos símbolos de saída associados aos estados ou às transições entre estados são descritos por funções de probabi-lidade. A seqüência de estados associada à seqüência de observações é desconhecida, daí o nome Modelo Oculto de Markov.

HMMs discretos diferem essencialmente de HMMs contínuous na forma de suas funções de probabilidade de saída, além, é claro, da necessidade de quantização para mapear os vetores de observações do espaço contínuo para o discreto. Os HMMs contínuos não utilizam quantização, eliminando assim, erros inerentes a este processo. HMMs contínuos usam múltiplas misturas de funções densidade de probabilidade Gaussianas no cálculo das probabilidades de saída do modelo resultando num número maior de variáveis a serem treinadas em relação aos HMMs discretos, elevando a complexidade computacional do treinamento de HMMs contínuos.

Já os HMMs semi-contínuos são uma espécie de junção dos outros dois modelos: usam técnicas de quantização com funções densidade de probabilidade contínuas. Eles combinam coeficientes de peso dependentes de modelos discretos com um *codebook* de função densidade de probabilidade contínua, ou seja, *codebooks* de funções densidade são combinadas com probabilidades de saída discretas. Além de diminuir a quantidade de parâmetros livres e a complexidade computacional inerentes ao treinamento de HMMs contínuos, os HMMs semi-contínuos podem manter a robustez na modelagem de funções densidade de probabilidade de grandes misturas como no HMMs contínuos.

Os HMMs podem modelar fones, difones, trifones ou palavras. Quando empregados no tratamento de vocabulários pequenos como dígitos (de *zero* a *nove* por exemplo), modelos de palavras podem ser usados. Neste caso haverá um HMM para cada palavra do dicionário. Já para a fala contínua seria impossível este tipo de modelo, pois a complexidade tornar-se-ia absurdamente alta para tal emprego. Como o vocabulário é vasto, o uso de modelos de fones, difones ou trifones é mais indicado para este tipo de aplicação, pois os modelos podem ser

combinados para formar cada palavra.

No reconhecimento de fala os modelos treinados também podem ser dependentes ou independetes de locutor. No primeiro caso, os modelos são treinados com uma base de treinamento gravada por um locutor e o mesmo também faz parte da etapa de reconhecimento. Desta forma os modelos só terão máximo desempenho no reconhecimento da fala do locutor que os "treinou". Os modelos independentes de locutor, como o nome já diz, usam vastas bases de dados com diferentes locutores que não são usados na fase de reconhecimento. Para um bom desempenho deste tipo de modelo, faz-se necessária uma grande diversidade na base utilizada na modelagem com locutores de diferentes idades, sexos e regiões. Quanto maior e diversificada a base de dados, melhor o desempenho.

Além do reconhecimento de fala, existem diversas aplicações em que os HMMs são utilizados destacando-se o reconhecimento de padrões eletroencefalográficos, modelagem de erros em canais digitais, modelagem de equalização cega em canais de comunicação, sequenciamento de DNA, entre outros.

1.1 Organização do Trabalho

Este trabalho foi realizado com o intuito de aperfeiçoar o sistema de reconhecimento automático de fala do Laboratório de Processamento Digital de Sinais de Multimídia em Tempo Real da Faculdade de Engenharia Elétrica e de Computação da UNICAMP, introduzindo-se modelos contínuos à plataforma que usava apenas HMMs discretos [ANDR 01]. Futuramente pretende-se implantar esta plataforma em um sistema embarcado para aplicações em tempo real. Com a introdução dos HMMs contínuos procurou-se também verificar o desempenho destes modelos de palavras independentes de locutor aplicados ao reconhecimento de dígitos conectados em relação ao sistema desenvolvido em [ANDR 01]. Além disso, procurou-se avaliar também o seu comportamento quando alterados o número de estados e misturas dos modelos. Para tal, utilizaram-se duas bases de dados, uma em português brasileiro (LPDF Dígitos) e uma do inglês americano (TIDIGITS). O trabalho também apresenta uma comparação de desempenho da base TIDIGITS entre o sistema desenvolvido e a ferramenta livre HTK (*HMM Toolkit*), além de detalhar a teoria envolvida e os procedimentos aplicados na utilização do HTK da forma mais simples possível. Isto foi feito com o objetivo de facilitar

a compreensão e o uso deste *software* tão empregado e cuja literatura deixa a desejar.

O trabalho está dividido em seis capítulos descritos a seguir.

O Capítulo 2 relata a teoria básica que envolve o treinamento e o reco-nhecimento de fala. Neste capítulo são descritas as etapas do processamento do sinal de fala como também a teoria sobre os HMMs e os algoritmos utilizados no treinamento dos mesmos.

O Capítulo 3 trata do sistema desenvolvido neste trabalho detalhando todas as etapas realizadas, a topologia empregada, a estruturação dos dados, os métodos de treinamento utilizados, além de apresentar a interface de usuário da plataforma com explicações sobre o funcionamento do sistema.

O Capítulo 4 é dedicado ao *software* HTK. Ele descreve a teoria envolvida na utilização desta ferramenta com destaques para sua arquitetura e processamento. Nesse capítulo também são apresentadas as etapas de programação utilizadas no procedimento experimental, com destaque para todas as linhas de comando, detalhamento dos arquivos utilizados bem como aqueles gerados pelo HTK.

O Capítulo 5 apresenta os resultados experimentais obtidos com descrição das bases de dados empregadas e dos testes realizados neste trabalho.

O Capítulo 6 apresenta as conclusões e as sugestões de trabalhos futuros.

Finalmente, o Apêndice A traz informações complementares para o processo de treinamento de dígitos conectados realizado pelo HTK e o Apêndice B apresenta o trabalho "Primeiros Experimentos com Dados Articulatórios e sua Relação com a Segmentação Acústica", G. F. G. Yared, J. V. Gonçalves, P. A. Barbosa, L. G. P. Meloni, publicado na Revista de Estudos da Linguagem - Faculdade de Letras da UFMG, Vol 12 - N 1, 39 - 52, 2004.

Este documento foi elaborado da forma mais didática possível, com muitas ilustrações que possam ajudar na compreensão da teoria sobre o reconhecimento de fala utilizando HMMs contínuos e do *software* desenvolvido nesta aplicação. Espera-se que este trabalho seja uma fonte de consulta de fácil compreensão e que possa auxiliar em estudos e pesquisas futuras.

Capítulo 2

Teoria Básica do Reconhecimento de Fala

2.1 Introdução

Processar a fala usando o computador sempre foi um grande desafio para a comunidade científica. Trata-se de uma área multidisciplinar envolvendo conhecimentos de lingüística, engenharia elétrica, estatística, ciência da computação, entre outras, requerendo uma visão ampla de todas as etapas do processo, como também um envolvimento cada vez maior de profissionais das diversas áreas para o desenvolvimento de trabalhos em conjunto. Em linhas gerais, no sistema de reconhecimento automático de fala, procura-se "ensinar" o computador a processar o que foi dito e a "compreender" o significado da mensagem. Portanto, a repetição continuada de várias palavras para a formação de um vocabulário diversificado faz-se necessária e indispensável para o "aprendizado" e conseqüente sucesso do reconhecimento da fala pelo computador. O método de "ensino" mais didático é constituído de etapas específicas que procuram simular a forma de processamento e decodificação da fala normalmente realizada pelo cérebro humano.

Um sistema de reconhecimento de fala pode ser associado ao sistema auditivo humano. O processo de reconhecimento de fala pelo computador pode ser resumido nas seguintes etapas: Análise Espectral, Extração de Parâmetros, Geração de Modelos de Referência e o Reconhecimento propriamente dito. De forma bem resumida e simplista pode-se fazer um paralelo entre o processo de reconhecimento de fala realizado pelo computador e aquele que

ocorre no sistema auditivo humano conforme mostrado na Figura 2.1. A análise espectral estaria relacionada à vibração da membrana basilar, estimulada pelas ondas sonoras, no ouvido humano. A transdução neural, que é a transformação dos sinais em impulsos elétricos, estaria representada pela extração de parâmetros do sinal de fala realizada pelo computador. Os modelos de referência representariam todo o vocabulário de um determinado idioma, adquirido com o tempo, correspondendo a todo o conhecimento absorvido a priori por um determinado indivíduo e, finalmente, a semântica seria a fase de reconhecimento ou a compreensão do significado da mensagem. Obviamente, o processo de reconhecimento de fala realizado pela mente humana é muito mais complexo e dependente de outros fatores que contribuem na compreensão da mensagem, como o contato visual por exemplo, tornando o reconhecimento.de fala, porque não dizer, perfeito. Não é por menos que o cérebro humano é tão fascinante e objeto de muitas pesquisas no mundo inteiro. O esquema do processo de reconhecimento de fala apresentado na Figura 2.1 pretende apenas ilustrar uma comparação muito simples e, de certa forma, grosseira entre a máquina e o homem.

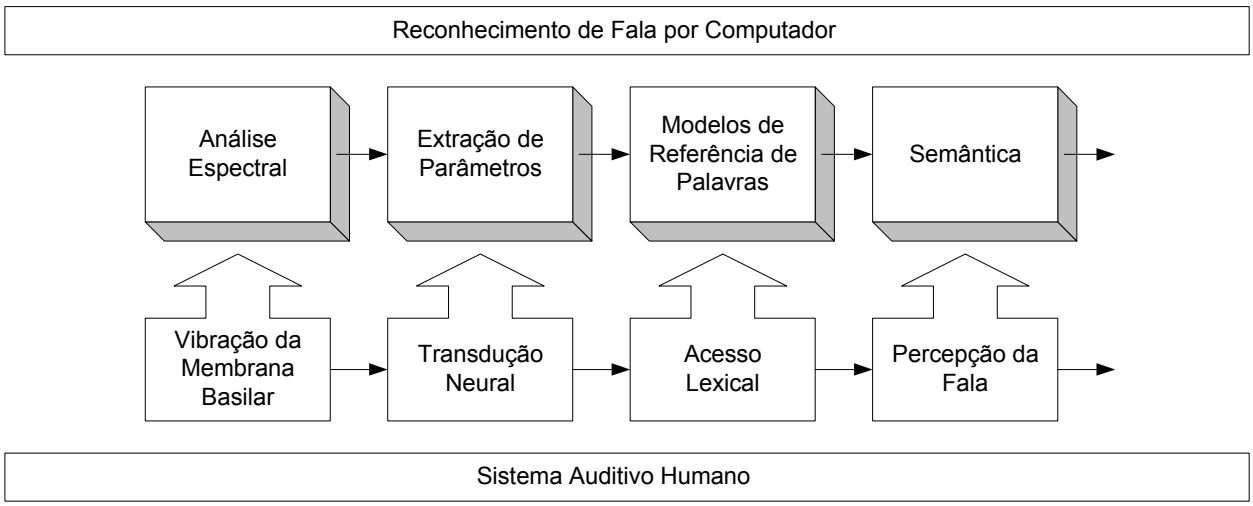

Figura 2.1: Reconhecimento Automático de Fala e o Sistema Auditivo Humano.

A análise espectral e a extração de parâmetros no computador compõem o pré-processamento do sinal de fala. A busca por modelos de referência, representados pelos Modelos Ocultos de Markov (HMMs), e a semântica correspondem à etapa de decodificação, ou seja, à partir dos modelos padrões de palavras ou de unidades fonéticas previamente treinados, além de técnicas de reconhecimento de padrões, é tomada a decisão pela palavra ou sentença correta. Todas essas etapas de processamento estão ilustradas na Figura 2.2 e são explanadas

a seguir.

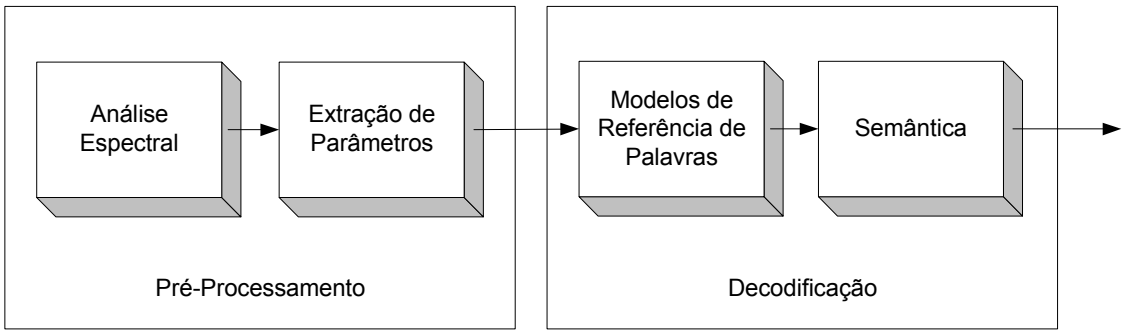

Figura 2.2: Pré-Processamento e Decodificação de um Sistema de Reconhecimento Automático de Fala.

2.2 Análise Espectral e Extração de Parâmetros

As ferramentas de processamento de sinais utilizadas em sistemas de reconhecimento de fala são muitas e devem representar da melhor forma possível as características espectrais do sinal de fala. As diversas fases do pré-processamento do sistema de reconhecimento de fala empregado são descritas a seguir. Todas as especificações relativas à etapa de pré-processamento do sistema são retiradas de [ANDR 01].

2.2.1 Filtragem Passa-Banda

Como futuramente pretende-se simular o sistema de reconhecimento de fala em um canal telefônico, emprega-se a filtragem passa-banda com freqüências de corte em 300 Hz e 3400 Hz. A função de transferência é especificada por

$$H_{pb}(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_7 z^{-7}}{a_0 + a_1 z^{-1} + \ldots + a_7 z^{-7}} \tag{2.1}$$

enquanto os coeficientes do filtro são mostrados na Tabela 2.1.

2.2.2 Filtragem de Pré-Ênfase

A filtragem de pré-ênfase compensa a atenuação nas altas freqüências do sinal de fala e suaviza o espectro do sinal tornando-o menos sensível aos efeitos de precisão finita. A

Tabela 2.1: 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	$0,00480775$

função de transferência do filtro de pré-enfase utilizado é

$$H_{pe}(z) = 1 - \mu_{pe} z^{-1} \tag{2.2}$$

em que o fator de pré-ênfase é $\mu_{pe} = 0,95$.

2.2.3 Janelas de Hamming Superpostas

O sinal que sai da filtragem de pré-ênfase é dividido em quadros, em geral de $10ms$. Em seguida, aplica-se a técnica de janelamento por Hamming através da qual o sinal é dividido em janelas com comprimento de $20ms$ e superposição de 50%, a fim de minimizar as descontinuidades do sinal no início e fim de cada quadro. Com a divisão do sinal de fala em intervalos pequenos de tempo, pode-se considerá-lo estacionário, ou seja, cujas características estatísticas são invariantes no tempo, permitindo, assim, o uso de métodos tradicionais de análise espectral. A janela de Hamming é definida por

$$h(n) = 0,54 - 0,46 \cos\left(\frac{2\pi n}{\xi - 1}\right) \qquad 0 \leq n \leq \xi - 1, \tag{2.3}$$

em que n é o índice da amostra e ξ é o número total de amostras da janela.

2.2.4 Extração de Parâmetros

Nesta fase efetua-se a extração das características do sinal da fala que perfazem um total de trinta e nove parâmetros acústicos. São eles: 12 parâmetros mel cepstrais, 12 derivadas primeira e 12 derivadas segunda dos parâmetros mel cepstrais, 1 parâmetro de energia, 1

derivada primeira e 1 derivada segunda do parâmetro de energia ($\mathbf{c}_p(n)$, $\mathbf{D\dot{c}}_p(n)$, $\mathbf{D\ddot{c}}_p(n)$, e_p, $D\dot{e}_p$, $D\ddot{e}_p$), respectivamente. Apenas o uso dos parâmetros mel cepstrais e energia não traz informação sobre a evolução dinâmica do sinal de fala, daí a necessidade da obtenção de suas derivadas.

Estudos científicos comprovaram que a percepção humana para as freqüências sonoras não segue uma escala linear. A extração das características do sinal de fala é realizada através da utilização de um banco de filtros triangulares passa-banda, em escala mel - uma escala perceptual amplamente utilizada em reconhecimento de fala - que melhor simula o processo da audição humana. Neste tipo de escala a freqüência central de cada filtro é espaçada uniformemente para as freqüências de até 1 kHz enquanto que para freqüências acima deste valor a freqüência central de cada filtro segue uma escala logarítimica. Neste trabalho utiliza-se um banco de 20 filtros triangulares na escala mel cujos espaçamentos e larguras de faixa são especificados em [MART 97].

A obtenção dos parâmetros mel cepstrais do sinal de fala consiste no cálculo da inversa da DCT (*Discrete Cosine Transform*) sobre o logaritmo da magnitude da energia de saída do banco de filtros

$$c_p(n) = \sum_{f=1}^{N_{tot}} \left(\log_{10} E(f)\right) \cos\left(\frac{n(f-0.5)\pi}{N_{tot}}\right) \qquad 1 \leq n \leq \Im. \tag{2.4}$$

Os coeficientes $c_p(0)$ são função da soma das energias de todos os filtros e não foi utilizado neste trabalho, assim como em [ANDR 01].

As derivadas primeira e segunda dos coeficientes mel cepstrais são calculadas de acordo com as equações a seguir

$$D\dot{c}_p(n) = \sum_{f=-F}^{F} \frac{f c_{p-f}(n)}{2F+1} \tag{2.5}$$

$$D\ddot{c}_p(n) = \sum_{f=-F}^{F} \frac{f D\dot{c}_{p-f}(n)}{2F+1} \tag{2.6}$$

onde

- $c_p(n)$ é o n-ésimo coeficiente mel cepstral pertencente ao p-ésimo quadro do sinal;
- n é o índice do coeficiente mel cepstral;

- N_{tot} é o número total de filtros;
- f é o índice do filtro;
- $E(f)$ é a energia de saída do f-ésimo filtro;
- $\Im$ é o número total de coeficientes mel cepstrais;
- $D\dot{c}_p(n)$ é a derivada primeira do n-ésimo coeficiente mel cepstral pertencente ao p-ésimo quadro do sinal;
- p é o índice do quadro do sinal;
- F é o número de quadros utilizados no cálculo das derivadas e
- $D\ddot{c}_p(n)$ é a derivada segunda do n-ésimo coeficiente mel cepstral pertencente ao p-ésimo quadro do sinal.

O cálculo das derivadas da energia foi realizado com as mesmas equações usadas no cálculo das derivadas dos coeficientes mel cepstrais substituindo o vetor de coeficientes $c_p(n)$ pela energia do quadro.

2.2.5 *Codebooks*

Quando o sistema de reconhecimento de fala utiliza Modelos Ocultos de Markov discretos na fase de treinamento, faz-se necessária uma representação dos vetores de coeficientes cepstrais e energia no domínio discreto. Para tal, emprega-se a Quantização Vetorial (VQ) que tem como objetivo reduzir a taxa de informação do sinal de fala.A quantização vetorial faz uso de um livro de código, também conhecido como *codebook,* que é acessado a cada etapa de quantização de uma coleção de parâmetros fazendo uso de uma medida de distorção. A mais utilizada é a distorção Euclidiana [MART 97].

O *codebook* é gerado a partir da base de dados de treinamento seguindo um critério de otimização. Um algoritmo muito eficiente para geração dos vetores do *codebook* foi proposto por Linde, Buzo e Gray, também conhecido como algoritmo LBG [GERS 92]. Para a formação do *codebook,* o espaço vetorial é particionado em células também chamado de regiões ou *clusters*. O número de células depende do tamanho do *codebook* desejado e cada célula contém um centróide obtido da média de todos os vetores classificados nessa região.

Sistemas de reconhecimento de fala que utilizam HMMs contínuos também utilizam *codebooks* cujas células contêm informações não apenas da média (ou centróide) mas também medidas de variância que são utilizadas no cálculo das misturas de funções Gaussianas que modelam os vetores contidos em cada região. Porém, a fase de quantização vetorial é eliminada do pré-processamento visto que HMMs contínuos modelam os sinais contínuos diretamente sem a necessidade de discretizá-los. Desta forma, evita-se perdas de informação do sinal de fala inerentes ao processo de quantização.

2.3 Modelos Ocultos de Markov

As cadeias de Markov são um dos mais importantes e utilizados conceitos da teoria probabilística. Elas são empregadas no tratamento de processos estocásticos, em geral não-estacionários, e incorporam uma quantidade mínima de memória [JELI 97].

Considerando uma seqüência de variáveis aleatórias Y_1, Y_2, Y_3, ..., Y_g cujos valores pertencem a um alfabeto finito $\hat{Y} = \{1, 2, \ldots, y\}$ e aplicando a fórmula de Bayes tem-se

$$P(Y_1, Y_2, Y_3, \ldots, Y_g) = \prod_{i=1}^{g} P(Y_i | Y_1, Y_2, Y_3, \ldots, Y_{i-1}). \tag{2.7}$$

A seqüência de variáveis aleatórias Y_1, Y_2, Y_3, ..., Y_g compõem uma cadeia de Markov de primeira ordem se

$$P(Y_i | Y_1, Y_2, Y_3, \ldots, Y_{i-1}) = P(Y_i | Y_{i-1}) \tag{2.8}$$

para todos os valores de i. Ou seja, o valor no instante i depende apenas do valor no instante imediatamente anterior, $i - 1$, e não de todos os valores nos instantes passados. Portanto, aplicando 2.8em 2.7 obtém-se

$$P(Y_1, Y_2, Y_3, \ldots, Y_g) = \prod_{i=1}^{g} P(Y_i | Y_{i-1}). \tag{2.9}$$

Uma cadeia ou máquina de estados de Markov de primeira ordem é caracterizada por um conjunto S de N estados, por uma matriz de probabilidade de transição de estados $\mathbf{A}$, de tamanho $N \times N$, e por um vetor de probabilidade de estar em um determinado estado no instante inicial, $\boldsymbol{\pi}$, de dimensão N. A Figura 2.3 a seguir é um exemplo de uma máquina de estados de Markov.

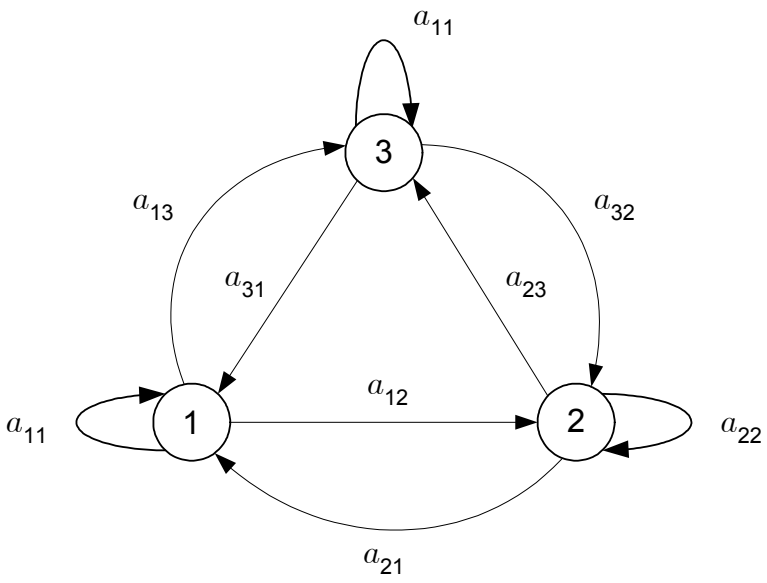

Figura 2.3: Máquina de Estados de Markov.

Os círculos representados na máquina de estados finita constituem os estados ou eventos de um determinado processo aleatório, no caso $N = 3$, e as setas indicam as probabilidades de transição do estado i para o estado j representadas pelos coeficientes a_{ij}. A matriz de probabilidade de transição de estados $\mathbf{A}$ fica assim caracterizada,

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \tag{2.10}$$

com $a_{ij} = P\left(s_{t+1} = j \mid s_t = i\right)$ para $1 \leq i, j \leq N$, em que s_t é o estado no instante t. Os coeficientes que compõem a matriz $\mathbf{A}$ devem satisfazer as seguintes condições

$$a_{ij} \geq 0 \qquad \forall i, j \tag{2.11}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \forall i, \tag{2.12}$$

ou seja, as probabilidades de transições de estado devem ser maiores ou iguais a *zero* e a soma de todas as probabilidades provenientes de um certo estado deve ser igual a *um*.

O Modelo Oculto de Markov é um tipo de Cadeia de Markov usado para modelar as propriedades estacionárias e transientes de um sinal. Os HMMs (*Hidden Markov Models*) possuem as mesmas características das Cadeias de Markov acrescidas de funções probabilísticas

que descrevem os símbolos de saída associados a estados ou a transições entre estados [RABI 86]. Além do mais, apenas a seqüência de observações é conhecida, mas a seqüência de estados associada a ela é oculta, daí porque se chamar Modelo Oculto de Markov. Os Modelos Ocultos de Markov podem ser contínuos, semicontínuos ou discretos e baseados em palavras ou unidades fonéticas. Dependendo do tipo de aplicação em reconhecimento de fala (fala contínua, palavras conectadas ou isoladas), o desempenho do sistema baseado em modelos contínuos, semicontínuos ou discretos pode ser consideravelmente diferente [CARD 93, EULE 96, HUAN 88].

Utiliza-se a seguinte notação para HMMs discretos:

- Seqüência de observações $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T\}$, de tamanho T, onde $\mathbf{o}_t = k$, $k \in \{1, 2, \ldots, K\}$, k é um símbolo e K é a quantidade de símbolos;
- Seqüência de estados $S = \{s_1, s_2, \ldots, s_T\}$, de tamanho T, onde $s_t = i$ e $i \in \{1, 2, \ldots, N\}$ (N é o número total de estados do modelo);
- Matriz de probabilidade de transição de estados $\mathbf{A} = \{a_{ij}\}$, de tamanho $N \times N$, onde $a_{ij} = P\left(s_{t+1} = j \,/\, s_t = i\right)$;
- Matriz de probabilidade de observação de símbolo de tamanho $K \times N$ é dada por $\mathbf{B} = \{b_{ki}\}$, onde $b_{ki} = b_i\left(\mathbf{o}_t = k\right) = b_i\left(k\right) = P\left(\mathbf{o}_t = k \,/\, s_t = i\right)$;
- Vetor de probabilidade de estado inicial $\boldsymbol{\pi} = \{\pi_i\}$, de tamanho N, onde $\pi_i = P\left(s_1 = i\right)$ e $1 \leq i \leq N$.

Um HMM é caracterizado por uma matriz de probabilidade de transição de estados, por uma matriz de emissão de símbolos e por um vetor de probabilidade inicial. O modelo é representado por $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$.

Para o caso de HMMs contínuos, o cálculo da matriz de probabilidade de observação de símbolo é diferente. Utilizam-se múltiplas misturas de funções densidade de probabilidade como veremos adiante.

Há três problemas importantes relacionados aos HMMs [RABI 93]:

1. Problema do Reconhecimento: dada uma seqüência de observações $\mathbf{O}$ e o modelo $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$, como calcular $P\left(\mathbf{O}/\lambda\right)$?

2. Problema da Seqüência: dada uma seqüência de observações $\mathbf{O}$, como escolher a melhor seqüência de estados S?

3. Problema do Treinamento: como estimar o modelo $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ de forma a maximizar $P(\mathbf{O}/\lambda)$?

Uma estrutura de HMM normalmente usada em reconhecimento de fala é a *left-right* por ser a mais apropriada para a modelagem das variabilidades temporais do sinal de fala e devido a sua característica de possuir transições de estados apenas da esquerda para a direita, com um ou mais saltos entre estados diferentes.A Figura 2.4 é um exemplo de uma estrutura *left-right* com um salto (transição) entre diferentes estados.

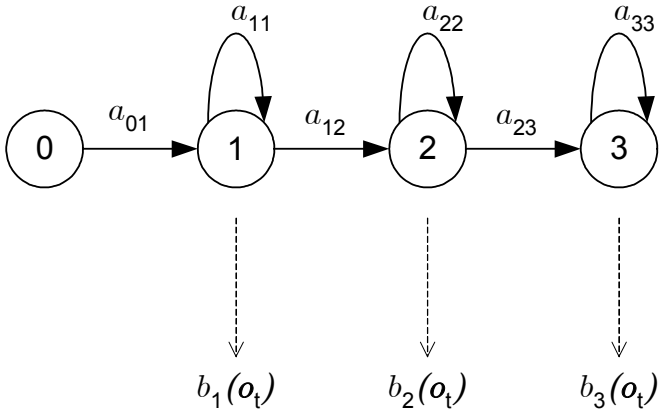

Figura 2.4: Estrutura *Left-Right*.

2.3.1 Algoritmos Aplicados ao Tratamento dos HMMs

As aplicações práticas requerem soluções dos problemas acima. Para tal, há *três* algoritmos que resolvem cada um deles, respectivamente: o algoritmo *Forward* (ou o algoritmo *Backward*), o algoritmo de Viterbi e o algoritmo de Baum-Welch.

Algoritmo *Forward*

A probabilidade de ocorrência de uma seqüência de observações parcial até o instante t e ocorrendo o estado i no instante t, dado o modelo λ, é definida pela variável $\alpha_t(i)$ [RABI

86]

$$\alpha_t(i) = P(\mathbf{o}_1\mathbf{o}_2 \ldots \mathbf{o}_t, s_t = i / \lambda). \tag{2.13}$$

1. Iniciação

$$\alpha_1(i) = \pi_i b_i(\mathbf{o}_1) \qquad 1 \leq i \leq N \tag{2.14}$$

2. Recursão

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\, a_{ij}\right] b_j(\mathbf{o}_{t+1}) \qquad 1 \leq t \leq T-1 \qquad 1 \leq j \leq N \tag{2.15}$$

3. Finalização

$$P(\mathbf{O}/\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.16}$$

A probabilidade de ocorrer uma seqüência de observações dado o modelo - $P(\mathbf{O}/\lambda)$ - é obtida pelo somatório de todas as variáveis $\alpha_t(i)$, para $t = T$, em todos os estados.

Algoritmo *Backward*

O algoritmo *Backward* também pode ser usado na resolução do primeiro problema envolvendo os HMMs descrito anteriormente. De forma similar à variável $\alpha_t(i)$, a probabilidade de ocorrência de uma seqüência de observações tomada parcialmente a partir do instante $t+1$ até o instante final T, dado que ocorre o estado i no instante t, sendo ainda dado o modelo λ, é definida pela variável $\beta_t(i)$ [RABI 86]

$$\beta_t(i) = P(\mathbf{o}_{t+1}\mathbf{o}_{t+2} \ldots \mathbf{o}_T / s_t = i, \lambda) \tag{2.17}$$

1. Iniciação

$$\beta_T(i) = 1 \qquad 1 \leq i \leq N \tag{2.18}$$

2. Recursão

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{o}_{t+1})\, \beta_{t+1}(j) \qquad t = T-1, T-2, \ldots, 1 \qquad 1 \leq i \leq N \tag{2.19}$$

3. Finalização

$$P\left(\mathbf{O}/\lambda\right) = \sum_{j=1}^{N} \pi_j b_j\left(\mathbf{o}_1\right) \beta_1\left(j\right) \tag{2.20}$$

O algoritmo *Backward* é semelhante ao algoritmo *Forward*, porém a recursão ocorre no sentido inverso do tempo. Esses dois algoritmos são de fundamental importância, pois também são usados na resolução do último problema que envolve os HMMs.

Algoritmo de Viterbi

Para a resolução do segundo problema, ou seja, encontrar a melhor seqüência de estados $S = \{s_1, s_2, \ldots, s_T\}$ para uma dada seqüência de observações $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T\}$ define-se a variável $\delta_t\left(i\right)$ que representa a maior probabilidade ao longo de uma dada seqüência de estados que finaliza no estado i, no instante t, dado o modelo λ [RABI 86].

$$\delta_t\left(i\right) = \max_{s_1, s_2, \ldots, s_{t-1}} P\left(s_1 s_2 \ldots s_{t-1}, s_t = i, \mathbf{o}_1 \mathbf{o}_2 \ldots \mathbf{o}_t \,/\, \lambda\right) \tag{2.21}$$

1. Iniciação

$$\delta_1\left(i\right) = \pi_i b_i\left(\mathbf{o}_1\right) \qquad 1 \leq i \leq N \tag{2.22}$$

$$\varphi_1\left(i\right) = 0 \tag{2.23}$$

2. Recursão

$$\delta_t\left(j\right) = \max_{1 \leq i \leq N} \left[\delta_{t-1}\left(i\right) a_{ij}\right] b_j\left(\mathbf{o}_t\right) \qquad 2 \leq t \leq T \qquad 1 \leq j \leq N \tag{2.24}$$

$$\varphi_t\left(j\right) = \arg\max_{1 \leq i \leq N} \left[\delta_{t-1}\left(i\right) a_{ij}\right] \qquad 2 \leq t \leq T \qquad 1 \leq j \leq N \tag{2.25}$$

3. Finalização

$$P^* = \max_{1 \leq i \leq N} \left[\delta_T\left(i\right)\right] \tag{2.26}$$

$$s_T^* = \arg\max_{1 \leq i \leq N} \left[\delta_T\left(i\right)\right] \tag{2.27}$$

4. *Backtracking*

$$s_t^* = \varphi_{t+1}\left(s_{t+1}^*\right) \qquad t = T-1, T-2, \ldots, 1$$

A variável $\varphi_t(j)$ armazena os estados que maximizam (2.24) a cada instante t e estado j. A melhor seqüência de estados, ou seqüência ótima, é determinada dentre todas as possíveis seqüências S e está representada por s_t^*.

O algoritmo de Viterbi pode ser implementado também na sua forma logarítmica, reduzindo-se, desta forma, a faixa dinâmica nos cálculos intermediários e melhorando-se a precisão numérica consequentemente [RABI 86].

1. Pré-Processamento

$$\widetilde{\pi}_i = \log(\pi_i) \qquad 1 \leq i \leq N \tag{2.28}$$

$$\widetilde{b}_i(\mathbf{o}_t) = \log[b_i(\mathbf{o}_t)] \qquad 1 \leq i \leq N, \quad 1 \leq t \leq T \tag{2.29}$$

$$\widetilde{a}_{ij} = \log(a_{ij}) \qquad 1 \leq i, j \leq N \tag{2.30}$$

2. Iniciação

$$\widetilde{\delta}_1(i) = \log(\delta_1(i)) = \widetilde{\pi}_i + \widetilde{b}_i(\mathbf{o}_1) \qquad 1 \leq i \leq N \tag{2.31}$$

$$\varphi_1(i) = 0 \qquad 1 \leq i \leq N \tag{2.32}$$

3. Recursão

$$\widetilde{\delta}_t(j) = \log(\delta_t(j)) = \max_{1 \leq i \leq N}\left[\widetilde{\delta}_{t-1}(i) + \widetilde{a}_{ij}\right] + \widetilde{b}_j(\mathbf{o}_t) \tag{2.33}$$

$$\varphi_t(j) = \arg\max_{1 \leq i \leq N}\left[\widetilde{\delta}_{t-1}(i) + \widetilde{a}_{ij}\right] \qquad 2 \leq t \leq T, \quad 1 \leq j \leq N \tag{2.34}$$

4. Finalização

$$\widetilde{P}^* = \max_{1 \leq i \leq N}\left[\widetilde{\delta}_T(i)\right] \tag{2.35}$$

$$s_T^* = \arg\max_{1 \leq i \leq N}\left[\widetilde{\delta}_T(i)\right] \tag{2.36}$$

5. *Backtracking*

$$s_t^* = \varphi_{t+1}\left(s_{t+1}^*\right) \qquad t = T-1, T-2, \ldots, 1 \tag{2.37}$$

Algoritmo de Baum-Welch

O terceiro e mais complexo problema envolvendo os HMMs é determinar o melhor método de estimação dos parâmetros do modelo $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ que maximize $P(\mathbf{O}/\lambda)$, o que corresponde a fase de treinamento dos HMMs. Essa fase é muito importante para um bom desempenho do sistema. O algoritmo de Baum-Welch, apresentado em termos das variáveis *Forward* e *Backward*, é utilizado na estimação das matrizes de transição e emissão [RABI 93].

Para uma única seqüência de observações $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T\}$, a re-estimação da probabilidade de transição do estado i para o estado j da matriz de transição de estados $\mathbf{A}$ é dada por

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i)\, a_{ij} b_j(\mathbf{o}_{t+1})\, \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i)\, \beta_t(i)}. \tag{2.38}$$

Também para uma única elocução, as funções de probabilidade dos símbolos de saída para os tipos de HMM discreto e contínuo são descritas a seguir.

- HMMs Discretos

No caso discreto, a quantidade de símbolos de saída é finita. A função de probabilidade re-estimada que um estado i emita um símbolo $\mathbf{o}_t = \mathbf{v}_k$ é obtida por [RABI 93]

$$\overline{b}_i(k) = \frac{\sum_{\substack{t=1 \\ t.q.\ \mathbf{o}_t = k}}^{T} \alpha_t(i)\, \beta_t(i)}{\sum_{t=1}^{T-1} \alpha_t(i)\, \beta_t(i)} \tag{2.39}$$

que possui as seguintes propriedades

$$\overline{b}_i(k) \geq 0 \qquad 1 \leq i \leq N, \quad 1 \leq k \leq K \tag{2.40}$$

$$\sum_{k=1}^{K} \overline{b}_i(k) = 1 \qquad 1 \leq i \leq N. \tag{2.41}$$

- HMMs Contínuos

Neste tipo de HMM a função densidade de probabilidade é contínua (PDF). Usa-se HMMs com PDFs contínuas para a modelagem direta de sinais contínuos sem a necessidade de quantizá-los. Uma PDF muito utilizada é a mistura de Gaussianas [RABI 93]. A PDF contínua apresenta-se na forma de misturas finitas, definida como

$$b_j(\mathbf{o}_t) = \sum_{m=1}^{M} c_{jm} \mathcal{F}\left(\mathbf{o}_t, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm}\right) \qquad 1 \leq j \leq N \tag{2.42}$$

onde

- $\mathbf{o}_t$ é o vetor de observações no instante t;
- c_{jm} é o coeficiente de peso da m-ésima mistura do estado j;
- M é o número de misturas ou regiões de um estado;
- $\mathcal{F}(\cdot)$ é qualquer densidade elipticamente simétrica ou log-côncava;
- $\boldsymbol{\mu}_{jm}$ é o vetor média da m-ésima mistura do estado j e
- $\mathbf{U}_{jm}$ é a matriz covariância da m-ésima mistura do estado j.

Neste trabalho assume-se que $\mathcal{F}(\cdot)$ é uma função densidade de probabilidade Gaussiana. Os coeficientes de mistura c_{jm} e a função densidade de probabilidade contínua devem satisfazer às seguintes restrições

$$\sum_{m=1}^{M} c_{jm} = 1 \qquad 1 \leq j \leq N \tag{2.43}$$

$$c_{jm} \geq 0 \qquad 1 \leq j \leq N, \quad 1 \leq m \leq M. \tag{2.44}$$

$$\int_{-\infty}^{\infty} b_j(\mathbf{x})\, d\mathbf{x} = 1 \qquad 1 \leq j \leq N \tag{2.45}$$

No caso de HMMs contínuos, além da matriz $\mathbf{A}$, os coeficientes de mistura c_{jm}, o vetor média $\boldsymbol{\mu}_{jm}$ e a matriz covariância $\mathbf{U}_{jm}$ também precisam ser re-estimados [RABI 93].

$$\overline{c}_{jm} = \frac{\sum_{t=1}^{T} \gamma_t(j,m)}{\sum_{t=1}^{T} \sum_{m=1}^{M} \gamma_t(j,m)} \tag{2.46}$$

$$\overline{\boldsymbol{\mu}}_{jm} = \frac{\sum_{t=1}^{T} \gamma_t(j,m)\,\mathbf{o}_t}{\sum_{t=1}^{T} \gamma_t(j,m)} \tag{2.47}$$

$$\overline{\mathbf{U}}_{jm} = \frac{\sum_{t=1}^{T} \gamma_t(j,m)\left(\mathbf{o}_t - \boldsymbol{\mu}_{jm}\right)\left(\mathbf{o}_t - \boldsymbol{\mu}_{jm}\right)'}{\sum_{t=1}^{T} \gamma_t(j,m)}, \tag{2.48}$$

para $1 \leq j \leq N$ e $1 \leq m \leq M$. O número total de observações da seqüência é especificado por T. A variável $\gamma_t(j,m)$ é a probabilidade de estar no etado j no instante t com o m-ésimo componente de mistura associado à observação $\mathbf{o}_t$, ou seja,

$$\gamma_t(j,m) = \left[\frac{\alpha_t(j)\,\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\,\beta_t(j)}\right]\left[\frac{c_{jm}\boldsymbol{\mathcal{F}}\left(\mathbf{o}_t, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm}\right)}{\sum_{r=1}^{M} c_{jr}\boldsymbol{\mathcal{F}}\left(\mathbf{o}_t, \boldsymbol{\mu}_{jr}, \mathbf{U}_{jr}\right)}\right]. \tag{2.49}$$

2.3.2 Escalonamento

Em aplicações práticas podem ocorrer problemas de *underflow* nas equações acima. Como os coeficientes a_{ij} e $b_i(k)$ são menores que 1, as variáveis *Forward* $\alpha_t(i)$ e *Backward* $\beta_t(i)$ aproximam-se de zero para instantes de tempo elevados, daí a necessidade de se aplicar um escalonamento dinâmico dessas variáveis [RABI 93], [MELO 00] para evitar que excedam a faixa de precisão de qualquer computador.

Assumindo que $\widehat{c}_t$ é um coeficiente de normalização independente do estado i, também que $\alpha_t(i)$ e $\beta_t(i)$ são as variáveis não escalonadas, $\widehat{\alpha}_t(i)$ e $\widehat{\beta}_t(i)$ as variáveis escalonadas e $\widetilde{\alpha}_t(i)$ e $\widetilde{\beta}_t(i)$ as variáveis atualizadas a partir das variáveis escalonadas, tem-se

1. Iniciação

$$\widehat{c}_1 = \frac{1}{\sum_{i=1}^{N} \alpha_1(i)} \tag{2.50}$$

$$\widehat{\alpha}_1(i) = \widehat{c}_1\alpha_1(i) \qquad 1 \leq i \leq N \tag{2.51}$$

2. Recursão *Forward*

$$\widetilde{\alpha}_{t+1}(j) = \left[\sum_{i=1}^{N} \widehat{\alpha}_t(i)\, a_{ij}\right] b_j(\mathbf{o}_{t+1}) \tag{2.52}$$

$$\widehat{c}_{t+1} = \frac{1}{\sum_{i=1}^{N} \widetilde{\alpha}_{t+1}(i)} \tag{2.53}$$

$$\widehat{\alpha}_{t+1}(i) = \widehat{c}_{t+1}\widetilde{\alpha}_{t+1}(i) \tag{2.54}$$

para $1 \leq t \leq T-1$, $1 \leq i \leq N$ e $1 \leq j \leq N$.

3. Recursão *Backward*

$$\widehat{\beta}_T(i) = \widehat{c}_T \tag{2.55}$$

$$\widetilde{\beta}_t(j) = \sum_{i=1}^{N} a_{ij} b_i(\mathbf{o}_{t+1})\, \widehat{\beta}_{t+1}(i) \tag{2.56}$$

$$\widehat{\beta}_t(j) = \widetilde{\beta}_t(j)\, \widehat{c}_t \tag{2.57}$$

para $T-1 \geq t \geq 1$, $1 \leq i \leq N$ e $1 \leq j \leq N$.

Devido ao escalonamento, a $P(\mathbf{O}/\lambda)$ é dada em escala logarítmica, ou seja,

$$\log\left(P(\mathbf{O}/\lambda)\right) = -\sum_{t=1}^{T} \log\left(\widehat{c}_t\right). \tag{2.58}$$

2.3.3 Múltiplas Seqüências de Observações

Especialmente com o uso de HMMs contínuos, para uma boa modelagem é necessária uma quantidade representativa de elocuções da mesma palavra, haja visto o número de parâmetros a serem re-estimados. Para múltiplas seqüências de observações e fazendo uso das variáveis escalonadas, as fórmulas de re-estimação devem ser modificadas. As novas fórmulas obtidas são dadas a seguir.

Matriz de Probabilidade de Transição para Múltiplas Seqüências de Observações (Discreto e Contínuo)

$$\overline{a}_{ij} = \frac{\sum_{d=1}^{D}\sum_{t=1}^{T_d-1} \widehat{\alpha}_t^d(i)\, a_{ij} b_j\left(\mathbf{o}_{t+1}^d\right) \widehat{\beta}_{t+1}^d(j)}{\sum_{d=1}^{D}\sum_{t=1}^{T_d-1} \left(\frac{\widehat{\alpha}_t^d(i)\, \widehat{\beta}_t^d(i)}{\widehat{c}_t^d}\right)} \qquad 1 \leq i, j \leq N \tag{2.59}$$

HMM Discreto para Múltiplas Seqüências de Observações

$$\overline{b}_i(k) = \frac{\sum_{d=1}^{D} \sum_{\substack{t=1 \\ t.q.\mathbf{o}_t=k}}^{T_d} \frac{\widehat{\alpha}_t^d(i)\,\widehat{\beta}_t^d(i)}{\widehat{c}_t^d}}{\sum_{d=1}^{D} \sum_{t=1}^{T_d} \frac{\widehat{\alpha}_t^d(i)\,\widehat{\beta}_t^d(i)}{\widehat{c}_t^d}} \tag{2.60}$$

HMM Contínuo para Múltiplas Seqüências de Observações

$$\overline{c}_{jm} = \frac{\sum_{d=1}^{D} \sum_{t=1}^{T_d} \left(\frac{\widehat{\alpha}_t^d(j)\,\widehat{\beta}_t^d(j)\,N_t^d(j,m)}{\widehat{c}_t^d} \right)}{\sum_{d=1}^{D} \sum_{t=1}^{T_d} \left(\frac{\widehat{\alpha}_t^d(j)\,\widehat{\beta}_t^d(j)}{\widehat{c}_t^d} \right)} \tag{2.61}$$

$$\overline{\boldsymbol{\mu}}_{jm} = \frac{\sum_{d=1}^{D} \sum_{t=1}^{T_d} \left(\frac{\widehat{\alpha}_t^d(j)\,\widehat{\beta}_t^d(j)\,N_t^d(j,m)\,\mathbf{o}_t^d}{\widehat{c}_t^d} \right)}{\sum_{d=1}^{D} \sum_{t=1}^{T_d} \left(\frac{\widehat{\alpha}_t^d(j)\,\widehat{\beta}_t^d(j)\,N_t^d(j,m)}{\widehat{c}_t^d} \right)} \tag{2.62}$$

$$\overline{\mathbf{U}}_{jm} = \frac{\sum_{d=1}^{D} \sum_{t=1}^{T_d} \left(\frac{\widehat{\alpha}_t^d(j)\,\widehat{\beta}_t^d(j)\,N_t^d(j,m)\left(\mathbf{o}_t^d - \boldsymbol{\mu}_{jm}\right)\left(\mathbf{o}_t^d - \boldsymbol{\mu}_{jm}\right)'}{\widehat{c}_t^d} \right)}{\sum_{d=1}^{D} \sum_{t=1}^{T_d} \left(\frac{\widehat{\alpha}_t^d(j)\,\widehat{\beta}_t^d(j)\,N_t^d(j,m)}{\widehat{c}_t^d} \right)} \tag{2.63}$$

para $1 \leq j \leq N$ e $1 \leq m \leq M$. A variável D indica o número total de elocuções ou seqüências de observações de uma mesma palavra usadas no treinamento de cada modelo correspondente. Cada sequência de observação possui tamanho T_d. Nas equações acima, $N_t^d(j,m)$ é dado por

$$N_t^d(j,m) = \frac{c_{jm} \mathcal{F}\left(\mathbf{o}_t^d, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm}\right)}{\sum_{r=1}^{M} c_{jr} \mathcal{F}\left(\mathbf{o}_t^d, \boldsymbol{\mu}_{jr}, \mathbf{U}_{jr}\right)} \qquad 1 \leq j \leq N \qquad 1 \leq m \leq M. \tag{2.64}$$

2.4 Reconhecimento

Dois algoritmos de reconhecimento são utilizados: o algoritmo *Level Building* [RABI 89a, RABI 93] e o *One Step* [RABI 93] também conhecido como *Frame Synchronous* [LEEC 89].

2.4.1 Algoritmo *Level Building*

O algoritmo *Level Building* faz uma busca assíncrona no tempo em busca da melhor seqüência de dígitos. O *Level Building* consiste em identificar a fronteira entre os dígitos de uma seqüência e, por meio do algoritmo de Viterbi, determinar o HMM mais provável. Por ser assíncrono no tempo, o decodificador *Level Building* necessita que toda a elocução seja armazenada em memória, para que se possa iniciar o processamento e tomar suas decisões.

2.4.2 Algoritmo *Frame Synchronous*

Este algoritmo realiza uma busca síncrona no tempo da seqüência de dígitos mais provável. Um parâmetro de entrada para este algoritmo é o número máximo de dígitos representados por níveis. É muito utilizado no processamento em tempo real, haja visto que sua implementação computacional é mais simples e utiliza programação dinâmica, que faz uso apenas da informação do instante anterior para tomar suas decisões do instante seguinte, ou seja, toma decisões para todos os níveis a cada quadro do sinal de fala [RABI 93].

Capítulo 3

Sistema de Reconhecimento de Dígitos Conectados

3.1 Introdução

Visando aperfeiçoar a plataforma de reconhecimento automático de fala do Laboratório de Processamento Digital de Sinais de Multimídia em Tempo Real da Faculdade de Engenharia Elétrica e de Computação da UNICAMP, foram introduzidos Modelos Ocultos de Markov Contínuos ao sistema de reconhecimento de dígitos conectados independente de locutor apresentado em [ANDR 01]. Utilizando o ambiente de programação Microsoft Visual Studio 6.0 e linguagem C++, toda a programação foi desenvolvida com o cuidado para implementação em tempo real.

Anteriormente, o sistema de reconhecimento de dígitos conectados do laboratório foi implementado utilizando-se modelos de palavras treinados por HMMs discretos. Para tal, existiam quatro fases distintas no processo: a extração de parâmetros, a geração do *codebook* e quantização, o treinamento e o reconhecimento. Ao utilizar HMMs contínuos, foram introduzidas mudanças sistêmicas. Não há mais necessidade da fase de quantização da base de treinamento e a matriz de probabilidade de observação de símbolos (matriz **B**) é calculada de forma diferente, utilizando-se somas ponderadas de múltiplas misturas de funções densidade de probabilidade Gaussianas. A matriz **B**, portanto, representa verossimilhanças e não mais probabilidades. Na fase de treinamento, além da matriz de probabilidade de transição de estados **A**, mais três variáveis são treinadas: os coeficientes de peso de cada mistura, o vetor

de médias e a matriz de covariâncias. A seguir, são apresentadas as modificações realizadas na construção do novo sistema. A Figura 3.1 é um exemplo de recohecimento de palavras isoladas, que também é realizado pelo sistema, cujo processo de reconhecimento consiste basicamente na busca da maior probabilidade de ocorrência de uma sequência de observação desconhecida, dados os modelos previamente treinados.

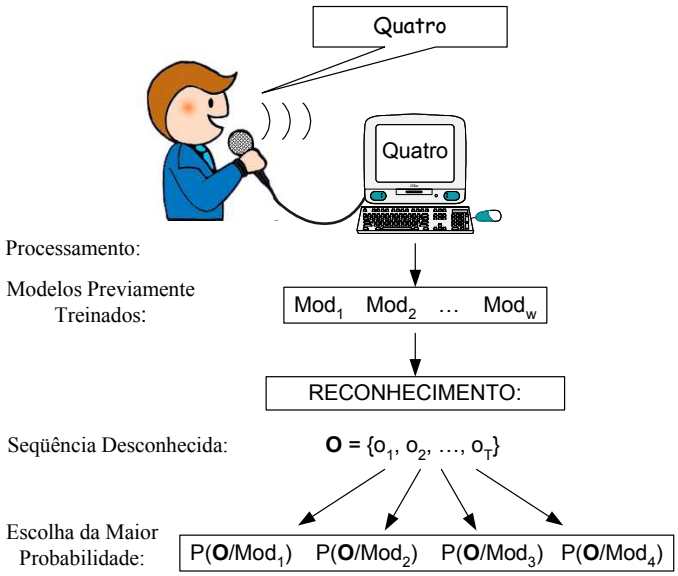

Figura 3.1: Sistema de Reconhecimento de Dígitos Isolados.

3.2 Pré-Processamento

Na fase de pré-processamento do sistema de reconhecimento de dígitos conectados baseado em HMMs contínuos, a análise espectral e a extração de parâmetros do sinal de fala são realizadas da mesma forma que em [ANDR 01] como explanada no capítulo anterior. Cada elocução da base de dados de treinamento é representado por um conjunto de vetores, de tamanho $\eta = 39$ cada um, que contêm as características extraídas do sinal de fala. A quantidade de vetores ou quadros depende da duração total da elocução. Cada vetor é composto pelos parâmetros mel cepstrais, energia e suas derivadas primeira e segunda, explicado anteriormente no Capítulo 2, seção 2.2.4. Esse vetor está disposto na Figura 3.2.

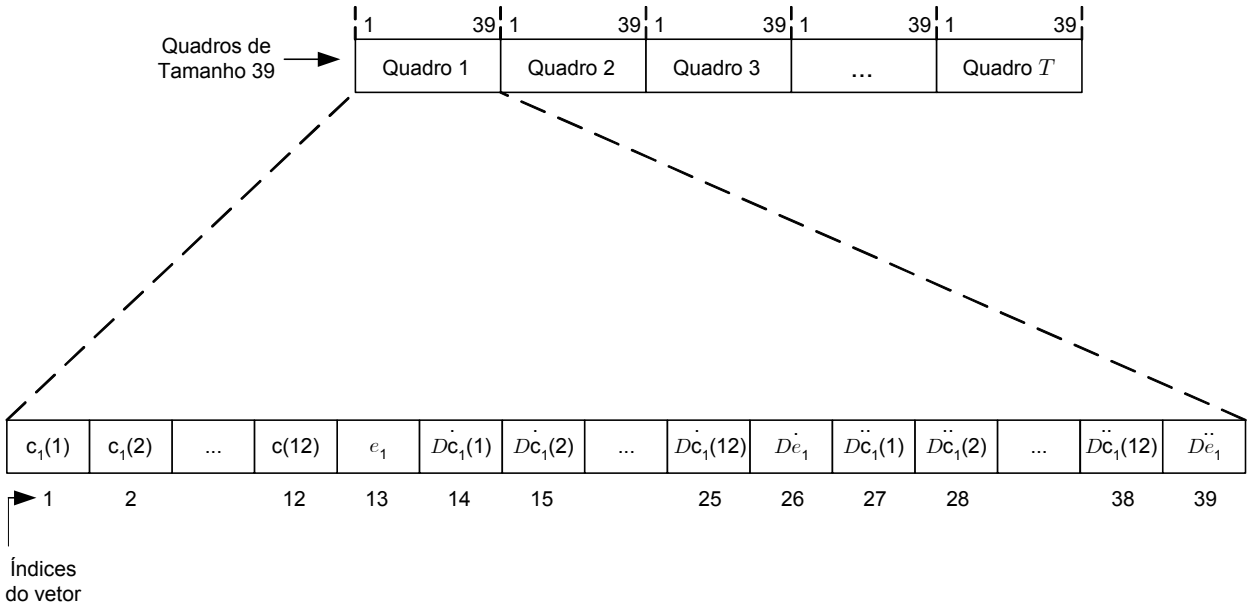

Figura 3.2: Vetores de Parâmetros do Sinal de Fala.

3.2.1 Topologia Empregada

A estrutura empregada nos modelos ocultos de Markov contínuos é a *left-right* com dois saltos. Conhecido também como o modelo de Bakis [JELI 76], a estrutura *left-right* tem como característica principal que os índices dos estados, que formam a seqüência associada ao modelo, crescem com o tempo ou permanecem os mesmos. Desta maneira, esta estrutura é a mais indicada para modelar sinais de fala cujas características mudam com o tempo de forma sucessiva. Na estrutura *left-right* o coeficiente de probabilidade de transição do estado i para o estado j tem como característica que [RABI 93]

$$a_{ij} = 0 \qquad j < i. \tag{3.1}$$

A restrição adicional associada ao número de saltos, ou seja, a quantidade de transições entre estados diferentes, determina ainda que

$$a_{ij} = 0 \qquad j > i + \Delta i. \tag{3.2}$$

Para o sistema desenvolvido, $\Delta i = 2$ como mostrado na Figura 3.3.

A topologia *left-right* também tem como característica que para o último estado do

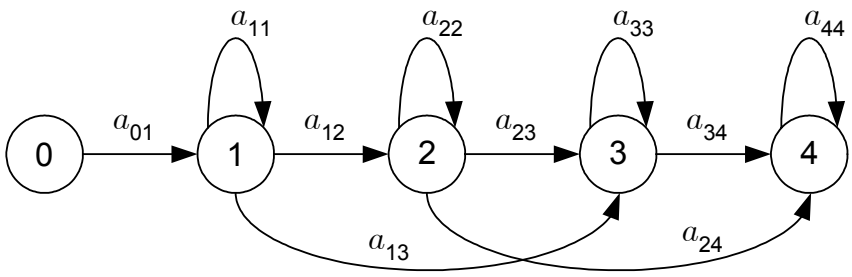

Figura 3.3: Estrutura *Left-Right* com Dois Saltos.

modelo a probabilidade de transição de estados é determinada por

$$a_{NN} = 1 \tag{3.3}$$

$$a_{Ni} = 0 \qquad i < N. \tag{3.4}$$

Devido às particularidades deste tipo de topologia, os componentes do vetor de probabilidade de estado inicial $\boldsymbol{\pi}$ são especificados por

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

3.2.2 Construção do *Codebook*

O dicionário de códigos é definido a partir da base de dados de treinamento para palavras isoladas utilizando-se o algoritmo LBG [GERS 92]. A base de treinamento contém um conjunto de elocuções para cada dígito w do vocabulário que é usado na construção de cada modelo λ_w correspondente. Primeiramente, de acordo com o algoritmo LBG, para cada modelo λ_w, todos os vetores de dimensão $\eta = 39$ de cada elocução representando o dígito w são distribuídos igualmente entre todos os estados de λ_w. Dependendo do número total de partições ou regiões M de cada estado do modelo, ou seja, do tamanho do *codebook*, calcula-se em cada estado i um vetor centróide a partir de todos os vetores pertencentes a este estado. Realiza-se então os seguintes procedimentos [RABI 93, HUAN 01]:

1. A região $m = 1$ é dividida em *duas* ($m = 2$), surgindo assim duas regiões com novos centróides que são calculados da seguinte maneira: primeiramente cada um dos η componentes do vetor centróide $\breve{\mathbf{x}}_{im}$, pertencente a região m do estado i, é dividido em

dois novos componentes $[\breve{x}_q]^+_{i\rho}$ e $[\breve{x}_q]^-_{i\rho}$ dados por

$$[\breve{x}_q]^+_{i\rho} = [\breve{x}_q]^+_{i\rho}(1+\epsilon) \qquad 1 \leq q \leq \eta \tag{3.6}$$

$$[\breve{x}_q]^-_{i\rho} = [\breve{x}_q]^-_{i\rho}(1-\epsilon) \qquad 1 \leq q \leq \eta, \tag{3.7}$$

onde

$$0,01 \leq \epsilon \leq 0,05,$$

que formarão dois centróides. A variável ρ é um número inteiro que varia de 1 até o tamanho corrente final do *codebook* ($m = 2$) e $\eta = 39$ é a dimensão do vetor centróide ou vetor média;

2. Todos os vetores pertencentes ao estado i são então novamente redistribuídos em cada uma das *duas* novas regiões do estado de acordo com um classificador de distância mínima. Calcula-se então novos centróides a partir dos vetores classificados em cada região obtendo-se $\breve{\mathbf{x}}_{i1}$ e $\breve{\mathbf{x}}_{i2}$ os centróides das regiões 1 e 2 do estado i, respectivamente. Este processo para encontrar o melhor conjunto de centróides para o novo *codebook* utiliza o algoritmo *k-means* [HUAN 01];

3. O processo é repetido para cada estado dividindo-se as *duas* regiões obtidas, ou apenas uma (e conseqüentemente o(s) centróide(s)), novamente em *duas* e redistribuindo todos os vetores do estado entre as m regiões obtidas, e assim por diante, até atingir o tamanho do *codebook* desejado (M).

O algoritmo LBG é aplicado no cálculo do *codebook* de cada modelo, resultando num *codebook* de M-*codewords*. Após esse procedimento, obtém-se para cada região de cada estado, um vetor média, um vetor variância e um coeficiente de peso calculados da seguinte forma:

- O vetor média $\boldsymbol{\mu}_{jm}$ é a média amostral dos vetores classificados na região m do estado j e corresponde ao próprio centróide da região obtido com o algoritmo LBG. O vetor média $\boldsymbol{\mu}_{jm}$ tem dimensão η. No sistema desenvolvido, arquivos serão gerados de acordo com o número de estados de cada modelo, ou seja, se há 11 modelos de dígitos e cada um contém 5 estados, então serão 55 arquivos gerados identificados individualmente

como, por exemplo, o arquivo *book_Cesta00_Est_01-02_part.cbk*, que se refere ao estado 1 com *duas* partições do modelo do dígito *zero*. Os dados estão estruturados dentro de cada arquivo da seguinte maneira: para cada estado de cada modelo os dados sobre as médias serão armazenados em uma matriz cujo número de linhas dependerá da quantidade de regiões, ou partições, existentes no estado como mostra a Figura 3.4, para o exemplo do estado 1 com *duas* misturas (regiões). Ao ser utilizada na construção da matriz **B,** cada linha da matriz correspoderá a um vetor centróide ou vetor média da região associada pertencente ao respectivo estado.

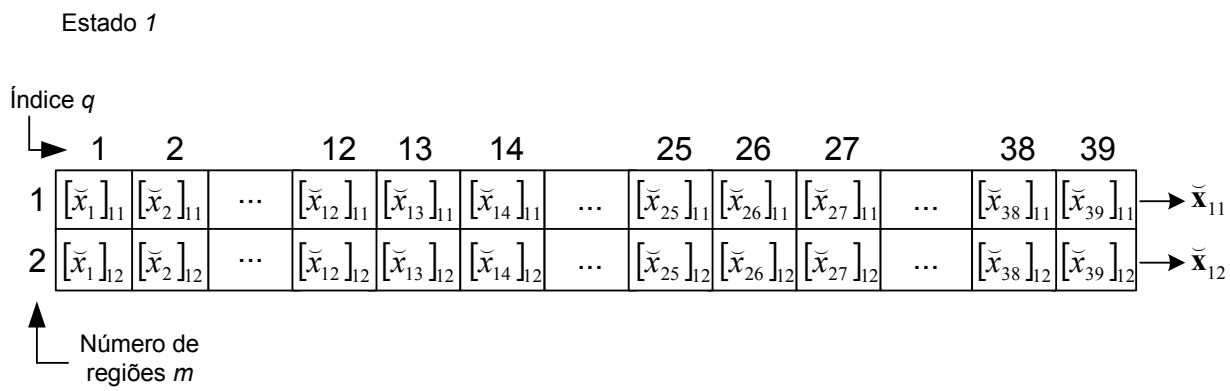

Figura 3.4: Estrutura dos Vetores Média.

- O componente v do vetor variância $\boldsymbol{\sigma}^2$ da região m é calculado por

$$\sigma_v^2 = \frac{1}{N_m - 1} \sum_{N_m} (x_v - \breve{x}_v)^2 \qquad 1 \leq v \leq \eta \tag{3.8}$$

onde x_v é o componente v do vetor de parâmetros $\mathbf{x}$ que pertence a região m, N_m é o conjunto total de vetores classificados na região m, η é a dimensão do vetor variância e $\breve{x}_v$ é o v-ésimo componente do vetor média (centróide) da região m, dado por

$$\breve{x}_v = \frac{1}{N_m} \sum_{N_m} x_v \qquad 1 \leq v \leq \eta, \tag{3.9}$$

que terá a mesma dimensão do vetor média e dos vetores de parâmetros classificados na região m, ou seja, tamanho $\eta = 39$. Os componentes do vetor variância de uma região m pertencente a um determinado estado j compõem a diagonal principal da matriz de covariância, $\mathbf{U}_{jm}$, dos vetores classificados na região m do estado j. A matriz de covariância diagonal tem tamanho $\eta \times \eta$.

- O coeficiente de peso da mistura c_{jm} é dado pelo número de vetores de parâmetros classificados na região m do estado j dividido pelo número total de vetores de parâmetros no estado j.

Arquivos contendo informações sobre as variâncias e coeficientes de peso também serão gerados. A quantidade de arquivos também dependerá do número de estados de cada modelo e será identificado como por exemplo *Variance_Cesta00_Est_01-02_part.var* que identificará as informações de variância e coeficiente de peso do estado 1 com *duas* partições do modelo do dígito *zero*. As informações de variâncias e coeficientes de peso serão armazenadas em matrizes de *quarenta* colunas onde cada linha conterá a informação de variância de cada mistura (η primeiras colunas) e do coeficiente de mistura c_{jm} correspondente, armazenado na última coluna. Portanto, cada arquivo conterá uma matriz cujo número de linhas dependerá da quantidade de partições de cada estado como mostrado na Figura 3.5.

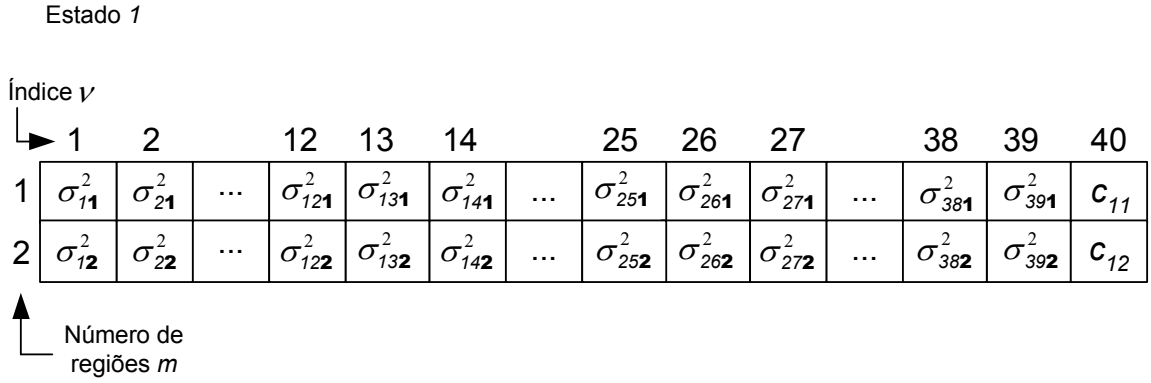

Figura 3.5: Estrutura dos Vetores Variância e Coeficiente de Peso.

A armazenagem das variâncias é feita em forma de vetores para cada mistura de um estado, mas no momento do cálculo da matriz **B** cada componente do vetor variância formará a diagonal principal da matriz covariância $\mathbf{U}_{jm}$ como na Figura 3.6.

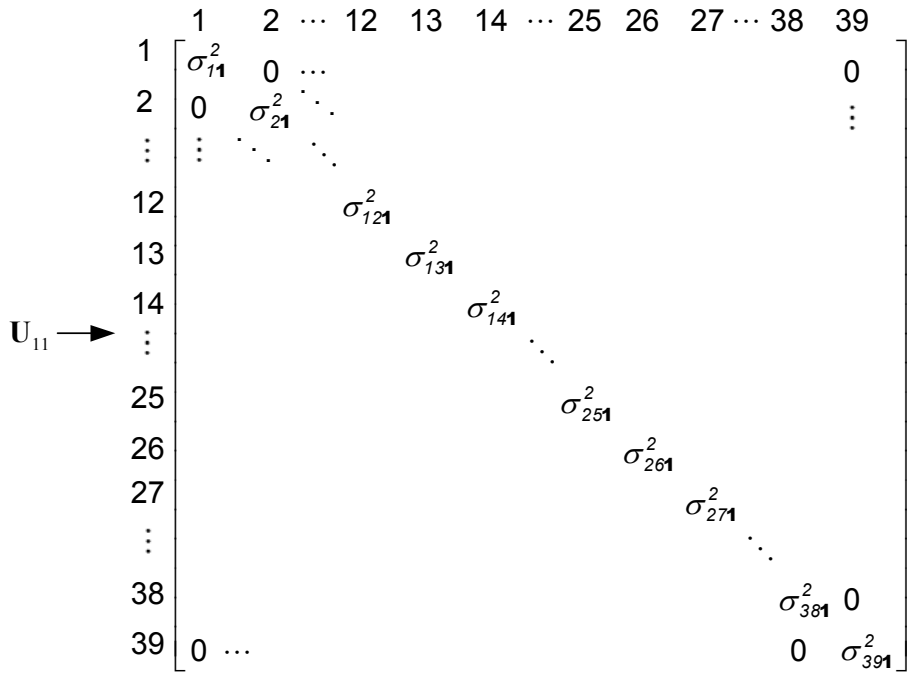

Figura 3.6: Matriz de Variância Diagonal.

3.3 Treinamento dos Modelos Ocultos de Markov

A fase de treinamento é a parte principal do sistema de reconhecimento de fala independente de locutor e onde foram realizadas as modificações mais significativas em relação ao sistema discreto. Para uma boa taxa de reconhecimento, os modelos devem ser muito bem treinados para que possam representar da melhor forma possível cada palavra do vocabulário. O tamanho do conjunto de dados de treinamento para cada palavra também tem um papel importante nesta fase, pois quanto maior a diversidade de informações, ou seja, quanto maior o número de locutores e de repetições da mesma palavra, o modelo será mais representativo e o desempenho geral do sistema será melhor.

3.3.1 Treinamento de Dígitos Isolados

Seguindo a linha de implementação do sistema discreto, os modelos ocultos de Markov obtidos a partir de uma base de treinamento de dígitos isolados são calculados [ANDR 01]. No treinamento de dígitos isolados, os HMMs devem possuir uma iniciação a priori. Para isto, a matriz **A** é iniciada com distribuição uniforme de probabilidade e a matriz **B** é iniciada utilizando-se segmentação uniforme das seqüências de observação. A segmentação uniforme

consiste em distribuir todos os quadros (vetores de 39 parâmetros) de uma seqüência de observação (elocução) igualmente entre todos os estados de um determinado modelo. Caso não seja possível uma distribuição uniforme de todos os quadros entre todos os estados, os quadros restantes serão distribuídos igualmente entre os últimos estados do modelo que conterão, assim, uma quantidade maior de quadros. Se, por exemplo, a seqüência de observação tiver 15 vetores e o modelo *dez* estados, os *cinco* primeiros estados do modelo conterão os *cinco* primeiros vetores da seqüência de observação (*um* vetor em cada estado) e os *cinco* últimos estados conterão os *dez* últimos vetores da seqüência (*dois* vetores em cada estado). Os componentes da matriz **B** devem ser bem iniciados para uma rápida convergência das fórmulas de re-estimação. No caso de HMMs contínuos, o vetor média, matriz covariância e o coeficiente de peso de cada mistura de cada estado de um modelo são iniciados e posteriormente re-estimados, haja visto que os componentes da matriz de emissão de símbolos **B** dependem destas variáveis. Os componentes de **B** são então calculados

$$b_j\left(\mathbf{o}_t^d\right) = \sum_{m=1}^{M} c_{jm} G\left(\mathbf{o}_t^d, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm}\right) \qquad 1 \leq j \leq N \tag{3.10}$$

onde $\mathbf{o}_t^d$ é o t-ésimo vetor de observações da d-ésima elocução ou seqüência de observações e $G(\cdot)$ é a função densidade de probabilidade Gaussiana multidimensional dada a seguir.

$$G\left(\mathbf{o}_t^d, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm}\right) = \frac{1}{(2\pi)^{\eta/2}\left|\mathbf{U}_{jm}\right|^{1/2}} \exp\left(-\frac{\left(\mathbf{o}_t^d - \boldsymbol{\mu}_{jm}\right)\mathbf{U}_{jm}^{-1}\left(\mathbf{o}_t^d - \boldsymbol{\mu}_{jm}\right)'}{2}\right) \tag{3.11}$$

sabendo-se que

- $\eta = 39$ é a dimensão do vetor $\mathbf{o}_t^d$;
- $|\mathbf{U}_{jm}|$ é o determinante da matriz de covariância $\mathbf{U}_{jm}$ e
- $\mathbf{U}_{jm}^{-1}$ é a matriz de covariância inversa.

Os coeficientes de peso da mistura e a função densidade de probabilidade devem sempre satisfazer as condições indicadas em (2.44 - 2.46).

Após a iniciação das matrizes **A** e **B**, efetua-se o treinamento pelo algoritmo de Baum-Welch, onde constam as principais modificações em relação a [ANDR 01]. Os parâmetros de cada modelo de N estados e M misturas são então re-estimados. Com exceção do vetor de probabilidade de estado inicial $\boldsymbol{\pi}$ que não precisa ser re-estimado, a matriz **A**, o vetor

de médias $\boldsymbol{\mu}_{jm}$, a matriz de variâncias $\mathbf{U}_{jm}$ e o coeficiente de peso c_{jm} são atualizados de acordo com as equações (2.60) e (2.62) - (2.65).

Com os novos *templates* ou modelos re-estimados $\bar{\lambda}_w = \left(\bar{\mathbf{A}}, \bar{\mathbf{B}}, \boldsymbol{\pi}\right)$ de cada dígito w, realiza-se um teste de convergência que consiste no cálculo da distância entre o novo *template* $\bar{\lambda}_w$ e o antigo $\lambda_w = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$. Se após uma época de treinamento esta distância for menor que $0,1\%$ a convergência é atingida, caso contrário, o modelo antigo é substituído pelo atual ($\lambda_w = \bar{\lambda}_w$) e o modelo atual $\bar{\lambda}_w$ é re-estimado mais uma vez através do algoritmo de Baum-Welch [RABI 93]. A iniciação dos parâmetros só é necessária na primeira iteração como mostrado na Figura 3.7.

3.3.2 Treinamento de Dígitos Conectados

Após a obtenção dos modelos treinados (*templates*) com dígitos isolados, dá-se início ao treinamento dos dígitos conectados. A técnica de treinamento *segmental k-means* [RABI 93] para sentenças de palavras conectadas também foi utilizada. Este algoritmo é uma variação do algoritmo de iteração *k-means* utilizado no processo de geração do *codebook*.

De posse dos *templates* de dígitos isolados, de posse também da base de treinamento de dígitos conectados e das transcrições respectivas de cada sentença, o algoritmo de reconhecimento *Level Building* é aplicado em cada seqüência de observações (elocução) para segmentar automaticamente cada sentença de dígitos conectados da base de treinamento em dígitos isolados ou *tokens* que são armazenados individualmente. Esta é a fase de segmentação em palavras. Em seguida, realiza-se o cálculo dos vetores média, matriz de covariância e coeficientes de peso de cada modelo. Há duas possibilidades para o cálculo dos vetores média, matriz de covariância e coeficientes de peso desta nova base de dígitos isolados (*tokens*):

1. na primeria iteração utilizam-se as informações originais dos *templates* de dígitos isolados que foram usados durante a fase de segmentação dos dígitos conectados, ou

2. na primeira iteração não são utilizadas as informações dos *templates* de dígitos isolados, porém aqui, cada conjunto de *tokens* é segmentado em estados, ou seja, todos os vetores de parâmetros pertencentes a este conjunto são distribuídos uniformemente entre os N estados do modelo correspondente e, de dentro de cada modelo, são extraídos os

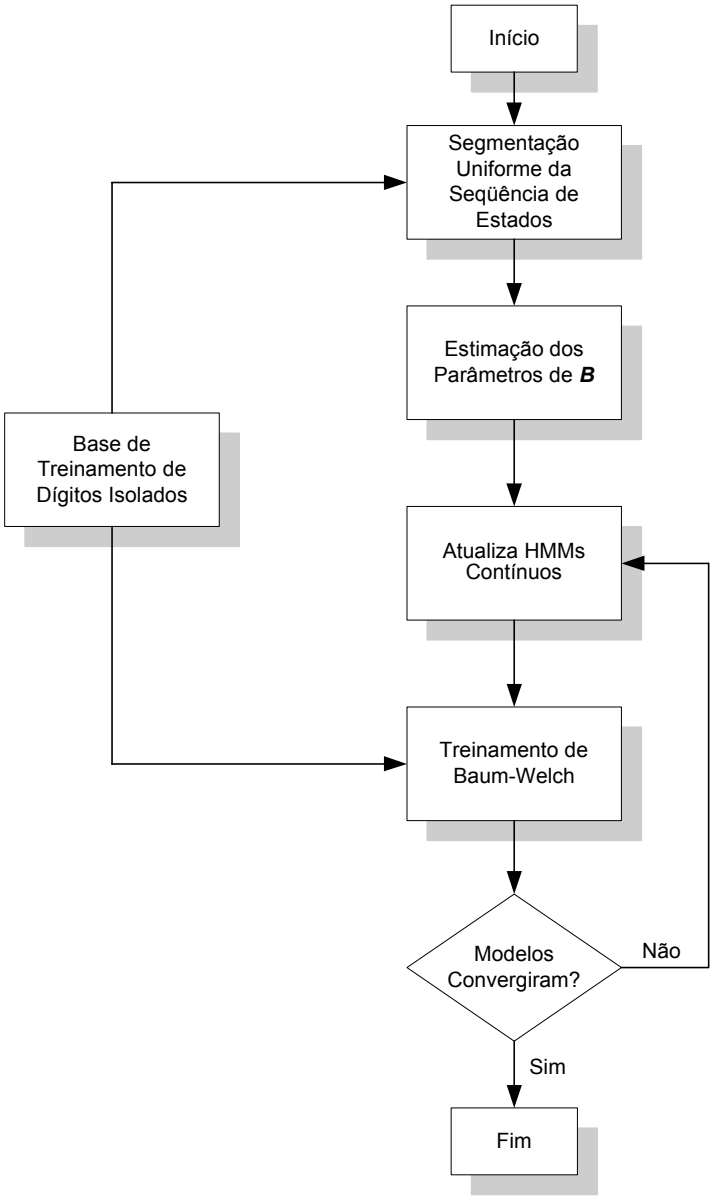

Figura 3.7: Algoritmo de Treinamento de Dígitos Isolados.

vetores média ou centróides, através do algoritmo LBG [GERS 92], bem como as variâncias e os componentes de peso de cada mistura de todos os estados. A Figura 3.8 ilustra de maneira simplificada esta segunda forma de obtenção de *templates* dos dígitos conectados.

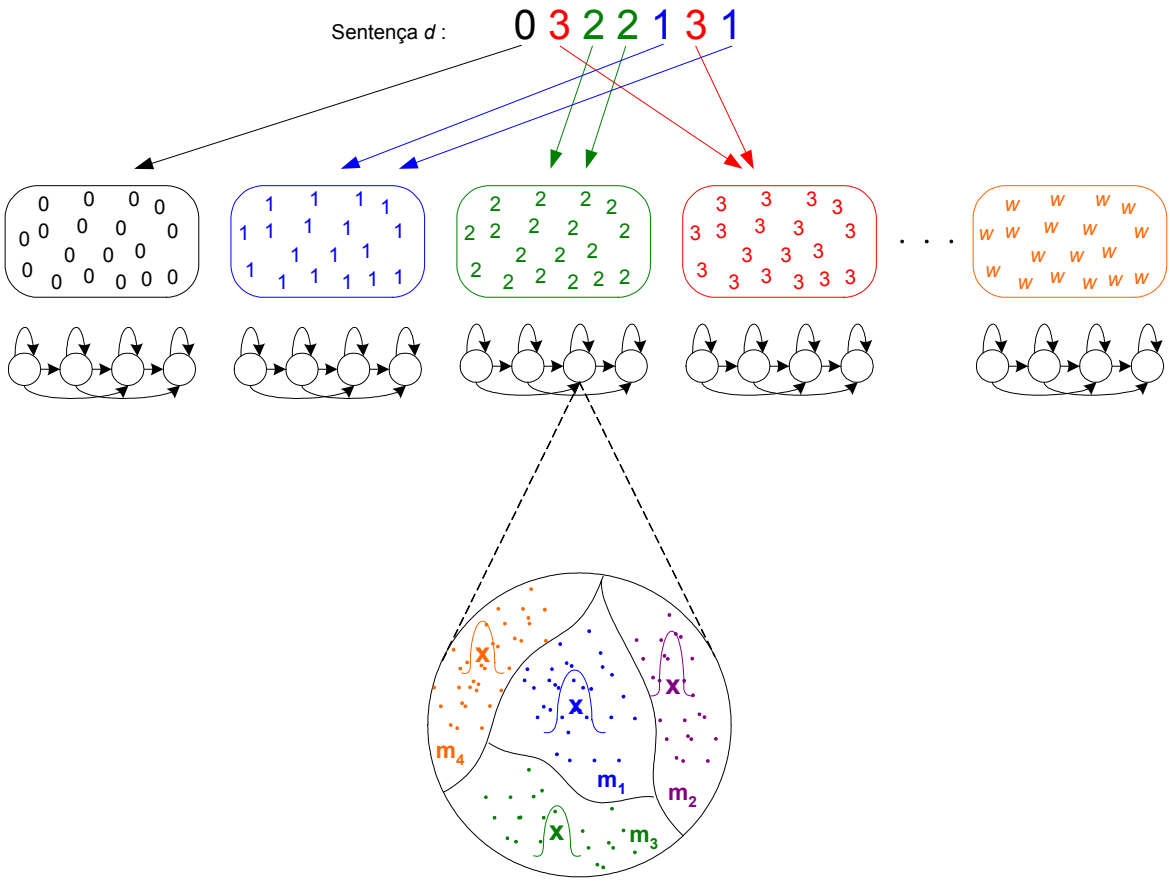

Figura 3.8: Obtenção dos *Tokens* e Segmentação em Estados.

Aplicam-se em seguida o algoritmo de Baum-Welch resultando em um novo conjunto de modelos de palavras re-estimados. Durante esta etapa também são obtidas as informações referentes ao modelo de duração de palavras (ver Capítulo 3.4). Em seguida, um teste de convergência é realizado: quando a distância entre o modelo atualizado e o modelo anterior, após uma iteração, for menor que 0,1%, a convergência é obtida. A distância é dada pela normalização da verossimilhança média de todos os *tokens* de um determinado modelo [YNOG 99].

$$P_{new}^{w} = \frac{\sum_{d=1}^{D} \max_{1 \leq j \leq N} P\left(j, \mathbf{O}^{d} / \bar{\lambda}_{w}\right)}{D} \tag{3.12}$$

$$P_{old}^{w} = \frac{\sum_{d=1}^{D} \max_{1 \leq j \leq N} P\left(j, \mathbf{O}^{d} / \lambda_{w}\right)}{D} \tag{3.13}$$

$$\Gamma_{new}^{w} = \frac{P_{new}^{w} - P_{old}^{w}}{P_{new}^{w}} \tag{3.14}$$

onde P_{new}^{w} é a probabilidade média corrente, P_{old}^{w} é a probabilidade média anterior, N é o número total de estados, D é o número total de elocuções ou seqüências de observações de uma mesma palavra e Γ_{new}^{w} é a distância corrente entre o modelo atualizado e o modelo anterior.

O algoritmo de treinamento *segmental k-means* para dígitos conectados é mostrado na Figura 3.9 [RABI 89].

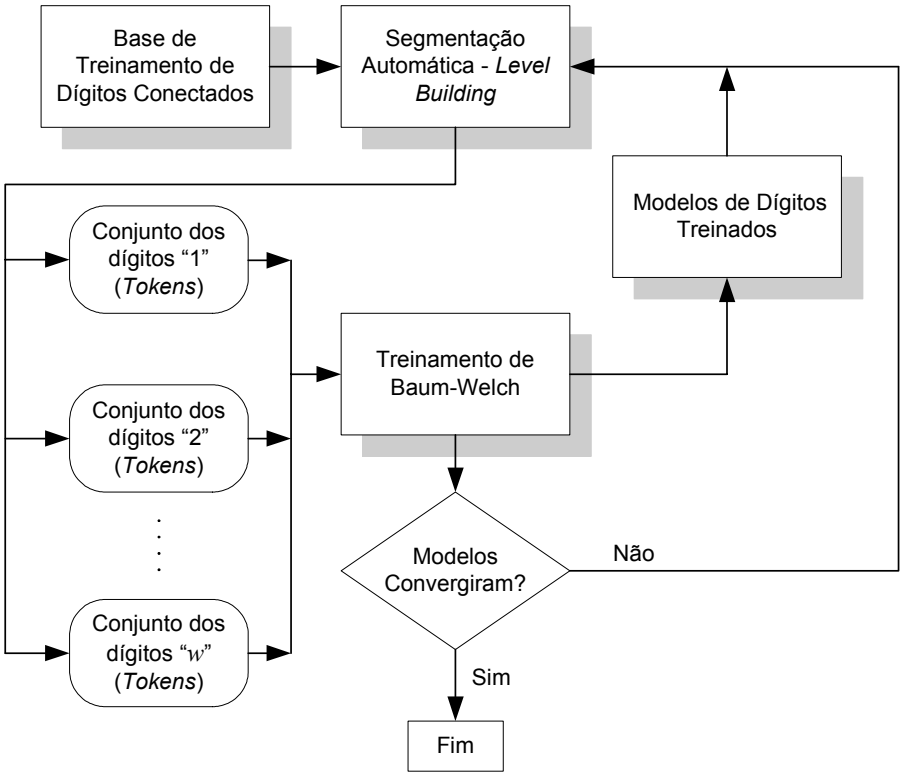

Figura 3.9: Algoritmo de Treinamento *Segmental k-means* para Dígitos Conectados.

3.4 Modelo de Duração de Palavra

Com o objetivo de melhorar o desempenho do sistema, assim como no sistema discreto [ANDR 01], utilizaram-se modelos de duração de palavras, aplicados a posteriori devido ao fato de que os Modelos Ocultos de Markov empregados não incorporam informação de duração de maneira consistente. Parametrizando a duração de palavras por uma função

densidade de probabilidade Gaussiana, a probabilidade de duração do dígito w ficou assim caracterizada

$$P_w(d_w) = \frac{1}{\sqrt{2\pi\sigma_w^2}} \exp\left(-\frac{(d_w - \mu_w)^2}{2\sigma_w^2}\right), \tag{3.15}$$

onde d é a duração da palavra w em número de quadros, σ_w^2 é a variância da duração da palavra w e μ_w é a sua duração média.

3.5 Reconhecimento

Na etapa de reconhecimento utilizou-se os seguintes algoritmos decodificadores: o algoritmo *Level Building* e o *Frame Synchronous* também conhecido como algoritmo *One Step* [RABI 93]. A estrutura original dos algoritmos permaneceram as mesmas do sistema de HMMs discretos [ANDR 01]. Houve uma pequena mudança apenas na construção da matriz **B** utilizada nos cálculos. Como cada modelo fornece além da matriz de probabilidade de transição de estados **A**, também o coeficiente de peso de cada mistura, o vetor média e a matriz de covariância; e não diretamente a matriz **B**, como nos modelos discretos, precisa-se construir a matriz de emissão de símbolos **B** antes de qualquer processo de decodificação.

3.6 Interface de Usuário do Sistema de Reconhecimento

As figuras a seguir mostram a interface de usuário do sistema de reconhecimento de dígitos conectados do Laboratório de Processamento Digital de Sinais de Multimídia em Tempo Real da Faculdade de Engenharia Elétrica da UNICAMP. Embora todos os experimentos realizados neste trabalho tenham sido voltados para o reconhecimento de dígitos conectados independente de locutor, o sistema também possibilita a prática de experimentos para reconhecimento de dígitos isolados e também para o caso dependente de locutor. A interface de usuário é referente apenas aos processos de treinamento e reconhecimento.

A Figura 3.10 mostra a tela inicial do sistema desenvolvido.

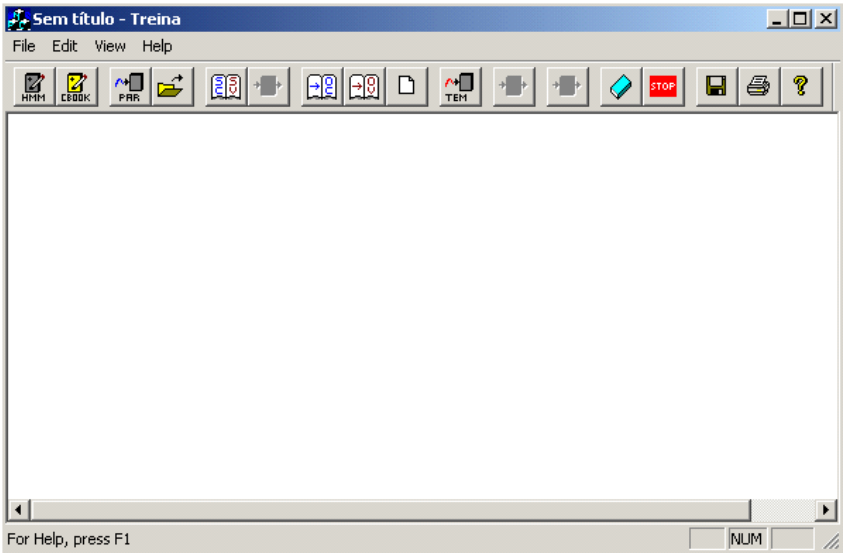

Figura 3.10: Tela Inicial do Sistema de Reconhecimento de Dígitos.

Na Figura 3.11, o ícone de configuração dos modelos permite que o usuário determine as características desejadas para o treinamento bem como para o reconhecimento de dígitos. O tipo de HMM a ser modelado pode se discreto ou contínuo, o processo de treinamento e/ou reconhecimento pode ser de palavras isoladas ou conectadas e o decodificador utilizado no reconhecimento pode ser o *Level Building* ou *One Step*. As características dos *templates* também são especificadas através do número de estados, número de saltos e quantidade de modelos. É possível também determinar que tipo de base de dados será utilizada, se a base TIDIGITS em inglês ou a base LPDF em português. Essas bases serão detalhadas no próximo capítulo.

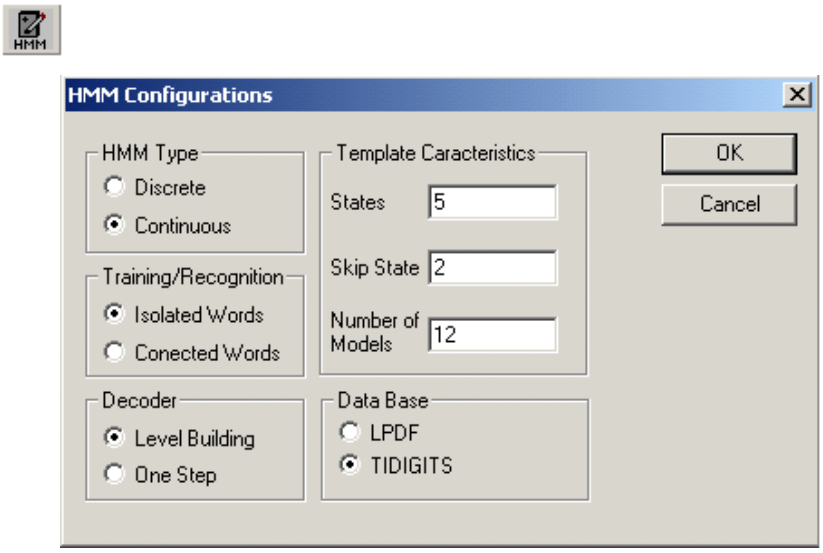

Figura 3.11: Tela de Configuração dos HMMs.

A Figura 3.12 mostra a tela de configuração dos *codebooks* onde é possível determinar o tamanho do mesmo. Para o caso de HMMs discretos, o sistema desenvolvido em [ANDR 01] determina dois tipos de *codebooks*, um apenas com os parâmetros de energia e suas derivadas, de tamanho 32, e outro com os parâmetros mel-cepstrais e suas derivadas, de tamanho 256. Para os HMMs contínuos, utiliza-se apenas um *codebook* que engloba todos os parâmetros mel-cepstrais, energia e suas respectivas primeiras e segundas derivadas. Portanto, ao modelar-se HMMs contínuos, o campo especificado por *Number of Mixtures* precisa ser preenchido, pois vai determinar o número de misturas que cada estado vai conter para todos os modelos.

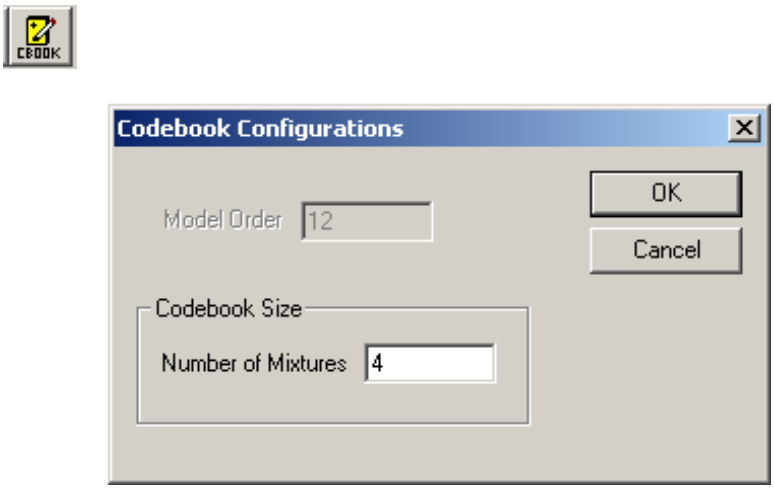

Figura 3.12: Tela de Configuração dos *Codebooks*.

Como mostrado na Figura 3.13, há duas opções para abrir os arquivos que contêm os parâmetros do sinal. A primeira permite selecionar com o *mouse* a quantidade desejada de arquivos dentro do diretório especificado e na segunda alternativa seleciona-se sempre o total de arquivos de parâmetros contidos no diretório, não sendo possível uma seleção manual de um número menor. O terceiro ícone, da esquerda para a direita, permite selecionar o diretório onde serão salvos os arquivos do *codebook* e das variâncias de cada estado de cada modelo. Os arquivos "*Codebooks* "conterão informações relativas aos centróides (vetores média) de todas as regiões do estado de um modelo enquanto que os arquivos "Variâncias" conterão além dos vetores de variâncias de cada região, também os coeficientes de peso correspondentes. O quarto ícone tem a função de gerar os arquivos "*Codebooks*"e "Variâncias" e só é habilitado após a abertura dos arquivos de parâmetros e após selecionado o caminho para salvar os arquivos gerados.

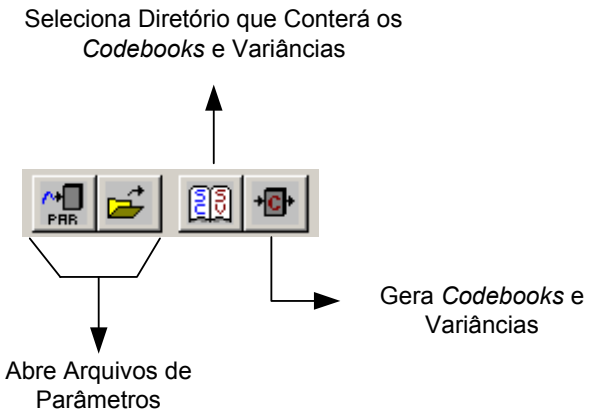

Figura 3.13: Ícones para Abrir Parâmetros e Salvar Variâncias e *Codebooks*.

A Figura 3.14 exibe alguns dos ícones usados no processo de treinamento de dígitos isolados. Após abrir os arquivos de parâmetros da base de dígitos isolados e gerar os *codebooks* e variâncias, como demonstrado anteriormente, abre-se os arquivos dos *codebooks* e variâncias e seleciona-se o diretório onde serão salvos os *templates* ou modelos treinados dos dígitos isolados.

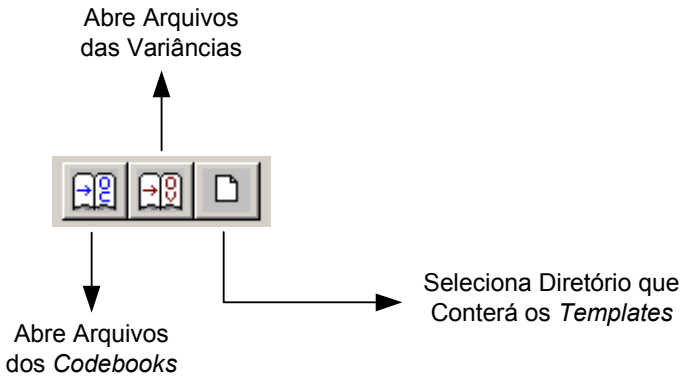

Figura 3.14: Ícones para Abrir *Codebooks* e Variâncias e Selecionar o Diretório de *Templates*.

Com o diretório que conterá os *templates* de dígitos isolados já selecionado, realiza-se o treinamento dos modelos através do ícone do meio indicado na Figura 3.15. Após a convergência de todos os modelos de dígitos isolados, estes serão salvos naquele diretório. Durante esta fase de treinamento de dígitos isolados, o primeiro ícone indicado na Figura 3.15 permanece desabilitado, haja visto que é usado somente no treinamento de dígitos conectados. Este ícone abre os *templates* de dígitos isolados já treinados que serão usados

para segmentar a base de treinamento de dígitos conectados através do algoritmo *Level Building* durante a primeira época de treinamento. O último ícone realiza o reconhecimento de dígitos isolados ou conectados, previamente selecionados, utilizando um dos algoritmos decodificadores indicados na tela de configuração de HMMs (ver Figura 3.11).

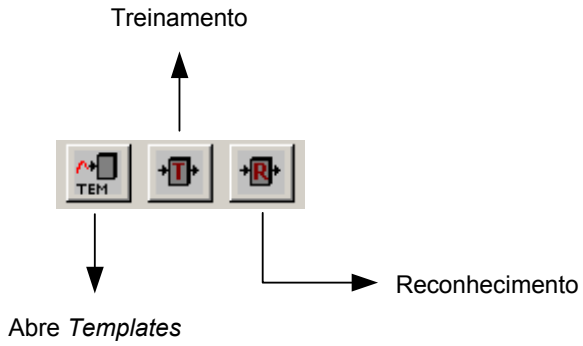

Figura 3.15: Ícones para Abrir *Templates*, Treinamento e Reconhecimento.

Os dois últimos ícones que se destacam no sistema realizam funções básicas que auxiliam a resolver eventuais problemas. O primeiro ícone da Figura 3.16 apaga todos os arquivos que foram selecionados, sejam parâmetros, *codebooks*, variâncias ou *templates*, desta forma pode-se fazer novas seleções de arquivos. O segundo ícone pode ser usado para parar o processamento de treinamento ou de reconhecimento.

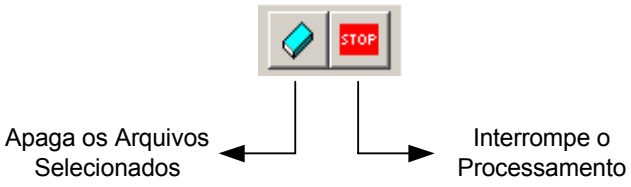

Figura 3.16: Ícones para Apagar Arquivos e Interromper o Processamento.

Capítulo 4

HTK - HMM *Toolkit*

4.1 Introdução

O HTK foi a primeira ferramenta para reconhecimento de fala, baseada em Modelos Ocultos de Markov (HMMs), disponível para a comunidade científica através do site do Departamento de Engenharia da Universidade de Cambrigde (*Cambridge University Engineering Department* - CUED) [HTK 93]. O HTK foi desenvolvido pelo *Speech, Vision and Robotics Group*, do Departamento de Engenharia da Universidade de Cambrigde. Em 1993, os direitos foram adquiridos pela *Entropic Research Laboratory Inc.* e, posteriormente, pela *Microsoft Corporation* que licenciou o HTK novamente para o CUED.

HMMs são usados essencialmente na modelagem de séries temporais e podem ser usados para diversos fins. No HTK não é diferente, mas foi basicamente projetado para construção de HMMs baseados em ferramentas de processamento de fala, mais especificamente no uso de sistemas de reconhecimento de fala, embora possa ser também aplicado no reconhecimento de caracteres e no sequenciamento de DNA.

O HTK engloba dois principais estágios de processamento:

1. as ferramentas de treinamento, usadas na estimação dos parâmetros do HMM por meio da base de treinamento e suas respectivas transcrições e

2. as ferramentas de reconhecimento que vão processar a base de sinais de teste e realizar o reconhecimento da mesma.

Embora projetado para tratar de observações contínuas usando, como probabilidade de

saída, distribuições de densidade contínuas, o HTK também processa seqüências de observações discretas usando probabilidades discretas.

O tópico a seguir discute a arquitetura do *software* HTK, descrevendo as principais ferramentas que são utilizadas no processo de extração de parâmetros, de treinamento e reconhecimento.

4.2 Arquitetura do HTK

De acordo com [YOUN 01] a arquitetura do *software* HTK pode ser bem resumida conforme a Figura 4.1. Ela descreve todas as ferramentas utilizadas pelo HTK no processamento e reconhecimento do sinal de fala.

Começando pelas ferramentas de controle do sistema, têm-se os seguintes módulos e suas respectivas funcionalidades:

- HMEM - gerencia a memória;
- HSHELL - administra as interfaces de entrada e saída com o usuário e possibilita interação com o sistema;
- HMATH - dá o suporte matemático;
- HSIGP - responsável pelas operações de processamento do sinal fundamentais no análise da fala;
- HLABEL - fornece o suporte necessário à interface de arquivos *label* (arquivos padrões próprios do HTK);
- HLM - responsável pela interface para arquivos de modelo de língua;
- HNET - fornece interface de redes e treliças;
- HDICT - fornece interface para os dicionários;
- HVQ - interface para *codebooks* de vetores quantizados;
- HMODEL - responsável pelas especificações dos HMMs.

O HTK é um *software* que permite trabalhar com vários formatos de arquivos de sinais possibilitando, desta forma, a entrada de dados também de outros sistemas. Os módulos a seguir completam a configuração geral do HTK:

- HWAVE - responsável por todo sinal de fala, de entrada e saída, a nível de forma de onda;
- HPARM - responsável por todo sinal de fala a nível de parâmetros;
- HAUDIO - responsável pela entrada direta de áudio;
- HGRAF - gráficos interativos são suportados por este módulo;
- HUTIL - fornece uma variedade de rotinas para a manipulação dos HMMs;
- HTRAIN e HFB - responsáveis pelas diversas ferramentas de treinamento do HTK;
- HADAPT - suporta as ferramentas de adaptação do HTK;
- HREC - suporta as principais funções do processamento de reconhecimento.

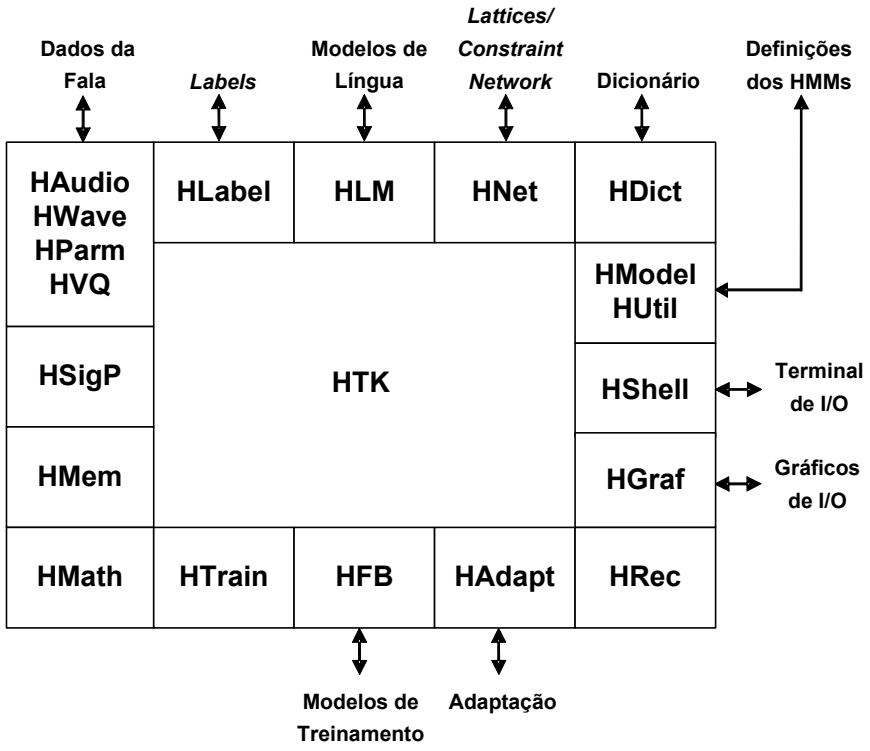

Figura 4.1: Arquitetura do Software HTK (Adaptado de [YOUN 01]).

As ferramentas do HTK funcionam através de linhas de comando e cada ferramenta possui um certo número de argumentos principais mais outros opcionais que são sempre precedidos pelo sinal menos (-). Os argumentos opcionais podem ser letras maiúsculas ou minúsculas e serem seguidas ou não de um número inteiro, um número real ou de uma string. O exemplo a seguir mostra a linha de comando para executar a ferramenta fictícia HFoo [YOUN 01].

HFoo −T 1 −f 34.3 −a −s meu_arquivo arquivo1 arquivo2

Neste exemplo os argumentos principais são os arquivos chamados *arquivo*1 e *arquivo*2. Os argumentos opcionais são $-T$, $-f$, $-a$ e $-s$. Com exceção do argumento opcional $-a$, todos são seguidos, respectivamente, por um número inteiro, um número real e por uma string (*meu_arquivo*). Por não ser seguido por nenhum valor, o argumento opcional $-a$ é usado como um $flag$ para habilitar ou desabilitar alguma característica da ferramenta **HFoo**. Os argumentos opcionais que são letras maiúsculas como $-T$, por exemplo, possuem a mesma função para todas as ferramentas do HTK. Arquivos contendo configurações específicas também podem ser usados para o controle das ferramentas do HTK. Para se obter um resumo da linha de comando e das opções utilizadas em qualquer ferramenta do HTK basta executar a ferramenta desejada, sem argumentos, no *prompt* do MS-DOS.

4.3 Etapas do Processamento

Com o objetivo de se obter uma melhor compreensão do funcionamento do HTK, as seções seguintes abordarão as quatro fases principais do processamento de reconhecimento de fala contínua do HTK baseado em unidades fonéticas. A Figura 4.2 [YOUN 01] descreve os diversos módulos envolvidos em cada fase do processamento que compreendem: Preparação de Dados, Treinamento, Reconhecimento e Análise.

4.3.1 Preparação de Dados

O HTK trabalha tanto com bases de dados obtidas de CD-ROMs como também permite que os sinais sejam gravados através da ferramenta **HSLab**. Durante a fase de preparação dos dados, a ferramenta **HCopy** tem um papel muito importante visto que é usada para copiar

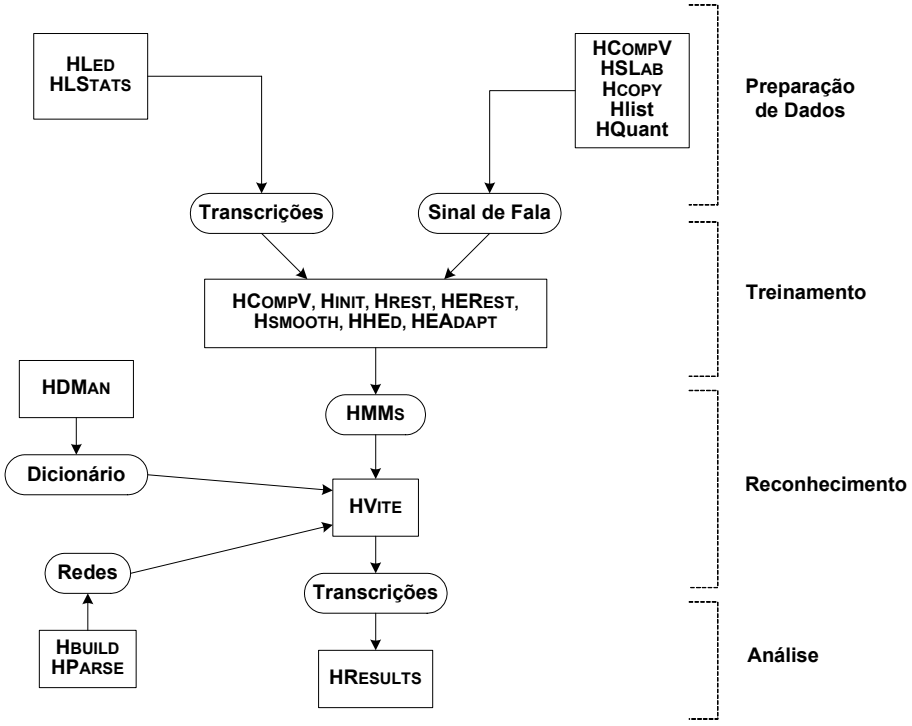

Figura 4.2: Estágios de Processamento do HTK. (Adaptado de [YOUN 01]).

um ou mais arquivos de dados da fala, e suas transcrições associadas, para um arquivo de saída específico. Em geral, essa ferramenta faz uma cópia inteira do arquivo, mas é possível extrair também apenas partes dele para depois concatená-las dependendo do procedimento adotado. Através desta ferramenta, os arquivos de entrada são convertidos para uma forma parametrizada de maneira que possam ser tratados. Nesta fase, a ferramenta **HList** é usada para inspecionar as conversões realizadas no conteúdo dos arquivos de áudio. Esses arquivos são então convertidos em arquivos MLF (*Master Label Files*), um formato usado pelo HTK, através da ferramenta **HLEd**. Para finalizar a preparação dos dados, a ferramenta **HLStats** pode ser usada para extrair estatísticas dos arquivos MLF gerados. Toda essa preparação tem como finalidade a construção do conjunto de HMMs da etapa seguinte. No caso de HMMs baseados em probabilidades discretas, utiliza-se a ferramenta **HQuant** na geração dos *codebooks*.

4.3.2 Treinamento

Nesta segunda fase do processamento é necessária a definição de uma topologia para cada HMM. Esses modelos são gerados em arquivos de texto que permitem sua livre manipulação. As definições desses protótipos têm o objetivo apenas de fornecer uma visão geral das características e topologia do HMM que serão utilizadas pelas ferramentas de treinamento. São essas ferramentas de treinamento que irão calcular realmente todos os parâmetros do modelo.

Na produção dos modelos de HMMs baseados em fones, um conjunto inicial de modelos deve ser gerado. O HTK utiliza duas formas de iniciação desses modelos:

- a primeira é através de dados de *bootstrap*, que são arquivos de fala segmentados em fones, utilizados pela ferramenta **HInit**, que gera um conjunto inicial de parâmetros usando a técnica *segmental k-means* na primeira iteração. As médias e variâncias são então estimadas. Da segunda iteração em diante a segmentação uniforme é substituída por um alinhamento de Viterbi. Esses parâmetros são então re-estimados através da ferramenta **HRest**, que utiliza o algoritmo de Baum-Welch;

- a ferramenta **HCompV** é responsável pela segunda forma, quando não há arquivos segmentados. Neste caso todos os modelos são iniciados da mesma maneira e as médias e variâncias em cada estado são iguais às médias e variâncias globais da fala.

A próxima etapa do treinamento é realizar uma única reestimação de todos os HMMs simultaneamente através da principal ferramenta do treinamento: **HERest**. Para cada sentença de treinamento os modelos de fones são então concatenados e, por meio do algoritmo *Forward-Backward*, estatísticas de ocupação de estado, médias, variâncias e outros são armazenadas para uma posterior reestimação dos parâmetros do HMM.

O HTK possibilita que os HMMs sejam refinados a cada iteração. Uma forma simples de fazer isso é usando a ferramenta **HHEd** que tem como meta modificar HMMs independentes do contexto e com uma única Gaussiana, e iterativamente refiná-los com o acréscimo de dependência de contexto e de distribuições Gaussianas com múltiplos componentes de mistura. A ferramenta **HERest** é então usada após cada iteração para reestimação dos novos parâmetros. Em sistemas dependentes de locutor, as ferramentas **HEAdapt** e **HVite** são usadas para melhor adaptar os modelos às características de locutores específicos.

O maior problema do treinamento é a escassez de dados, pois quanto mais complexo o modelo mais dados são necessários para garantir uma certa robustez na estimação dos parâmetros. O HTK também trata sistemas discretos e sistemas FTM (*Fully Tied Mixture*). A ferramenta **HSmooth** é usada então para suavizar distribuições em casos de insuficiência de dados nesses sistemas.

4.3.3 Reconhecimento

Baseada no algoritmo de Viterbi, a ferramenta que executa o reconhecimento, como seu próprio nome indica, é o **HVite**. Seus parâmetros de entrada são: uma rede que descreve as sentenças de palavras disponíveis a serem reconhecidas, um dicionário com todas as palavras pronunciadas e o conjunto de HMMs já treinados no estágio anterior. Neste processo, cada HMM será associado a cada fone da rede gerada após a conversão da rede de sentenças de palavras em redes de unidades fonéticas.

As redes de palavras são parâmetros de entrada necessárias ao **HVite** e podem ser geradas através da ferramenta **HBuild**. Uma outra ferramenta que pode ser utilizada também é o **HParse**, pois uma alternativa na especificação direta de redes de palavras é o uso da notação de uma gramática em linguagem de alto nível baseada no EBNF (*Extended Backus Naur Form*) que é usado na especificação do compilador. O **HParse** é uma ferramenta usada na conversão desta notação de alto nível em uma rede de palavras equivalente. Independentemente da ferramenta utilizada na geração das redes de palavras derivadas da base de treinamento, importante também é analisar o desempenho do sistema para a base de teste. Caso esta não exista, é possível gerá-la usando as possíveis sentenças definidas por esta mesma rede. A ferramenta **HSGen** pode ser utilizada para gerar as sentenças de teste que devem ser gravadas posteriormente. A ferramenta **HDMan** possibilita enfim, o gerenciamento dos grandes dicionários envolvidos em todo o processamento.

4.3.4 Análise

Uma vez concluído o reconhecimento das palavras e sentenças, faz-se necessária uma avaliação do desempenho do sistema projetado. Isto é possível através da ferramenta **HResults** que, a partir da própria base de testes utilizada na etapa de reconhecimento e de suas res-pectivas transcrições, realiza uma comparação entre as transcrições obtidas após o estágio de re-

conhecimento e as transcrições de referência da citada base. Para isso, utiliza técnicas de programação dinâmica para alinhar as duas transcrições e, assim, especificar o número de erros de substituições, inserções e de palavras suprimidas.

4.4 Experimentos Realizados com o HTK

Alguns experimentos de reconhecimento da base de teste TIDIGITS foram realizados utilizando o *software* HTK seguindo o mesmo procedimento experimental usado no sistema de reconhecimento de dígitos conectados do Laboratório de Processamento Digital de Sinais.de Multimídia em Tempo Real da Faculdade de Engenharia Elétrica da Unicamp. O intuito era avaliar o desempenho do sistema desenvolvido em relação ao HTK. Algumas fontes de referências bibliográficas utilizadas na elaboração dos procedimentos experimentais foram buscadas no tutorial do *software* HTK [YOUN 01], na lista de discussões disponível na *internet* e na literatura [HIRS 00, SAMU 03]. As principais etapas do experimento são descritas a seguir.

4.4.1 Procedimento Experimental

Tendo como base o método empregado na obtenção dos resultados do sistema desenvolvido, os experimentos utilizaram HMMs contínuos baseados em modelos de palavras com *dois* ou mais componentes de mistura Gaussiana para construção de um sistema de reconhecimento de dígitos conectados independente de locutor.

Nos passos seguintes todo o procedimento utilizado na preparação, treinamento e reconhecimento dos dados são relatados. Para uma melhor compreensão de cada etapa do processamento, o texto destaca algumas palavras que indicam os nomes dos arquivos que são feitos manualmente (arquivos *txt*) bem como os gerados pelo HTK. Para cada arquivo há uma explicação sobre suas características e o seu formato. As palavras que estão em **negrito** são as linhas de comando que devem ser digitadas no *prompt* do MS-DOS após instalação do *software* HTK.

Passo 1 - Gerando a Gramática

Na fase inicial de preparação dos dados, um único dicionário foi projetado para ambas as bases de treinamento e teste, além de uma gramática de suporte. O HTK possui uma linguagem própria para definição desta gramática que consiste de variáveis e expressões que descrevem todas as palavras a serem reconhecidas que, neste experimento, compreenderam os dígitos: "*zero*", "*one*", "*two*", "*three*", "*four*", "*five*", "*six*", "*seven*", "*eight*", "*nine*" e "*oh*". A gramática, escrita manualmente e em linguagem de alto nível, foi usada para a geração posterior de uma rede de palavras no formato SLF (*Standard Lattice Format*), próprio do HTK. A ferramenta **HParse** além de permitir a conversão desta linguagem de alto nível para a de baixo nível, utilizada pelo HTK (formato SLF), cria também a rede de palavras. Todo o procedimento do Passo 1 é resumido na Figura 4.3.

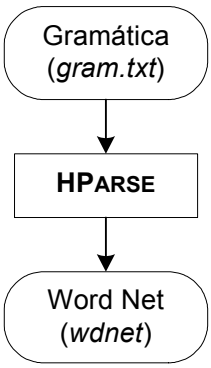

Figura 4.3: Rede de Palavras (Adaptado de [YOUN 01]).

O primeiro arquivo a ser criado manualmente é o chamado *gram.txt*, uma gramática que define a linguagem usada no experimento e contém as palavras que deverão ser reconhecidas, bem como o tipo de expressões nas quais elas estarão inseridas. Este arquivo deve assemelhar-se a Figura 4.4. As barras verticais significam alternativas (ou) e os sinais de menor que (<) e de maior que (>) definem que o que estiver entre esses sinais terá uma ou mais repetições. As palavras SENT-START e SENT-END indicam, respectivamente, o início e fim de cada sentença.

Após a criação do arquivo *gram.txt* a linha de comando seguinte deverá ser digitada no *prompt* do DOS.

HParse gram.txt wdnet

```
$digit = ONE | TWO | THREE | FOUR | FIVE | SIX | SEVEN | EIGHT | NINE | OH | ZERO;
(SENT-START(<$digit>)SENT-END)
```

Figura 4.4: Arquivo *gram.txt*.

Após a execução da ferramenta **HParse** o arquivo *wdnet* é gerado. Este arquivo contém a rede de palavras e assemelha-se a Figura 4.5.

```
VERSION=1.0
N=16   L=36
I=0    W=SENT-END
I=1    W=ZERO
I=2    W=!NULL
I=3    W=OH
I=4    W=NINE
I=5    W=EIGHT
I=6    W=SEVEN
I=7    W=SIX
I=8    W=FIVE
I=9    W=FOUR
I=10   W=THREE
I=11   W=TWO
I=12   W=ONE
I=13   W=SENT-START
I=14   W=!NULL
I=15   W=!NULL
J=0     S=2    E=0
J=1     S=2    E=1
...
```

Figura 4.5: Arquivo *wdnet*.

Passo 2 - Gerando o Dicionário

O passo seguinte consistiu em gerar um dicionário e para tal fez-se uso de uma lista feita manualmente contendo todas as palavras necessárias em ordem alfabética e uma segunda lista contendo as pronúncias das palavras usadas na gramática que indicará quais modelos serão utilizados no treinamento de cada palavra. A ferramenta **HDMan** é usada então para a construção do dicionário que consiste de todas as palavras, em ordem alfabética, existentes na primeira lista com suas respectivas pronúncias (Figura 4.6).

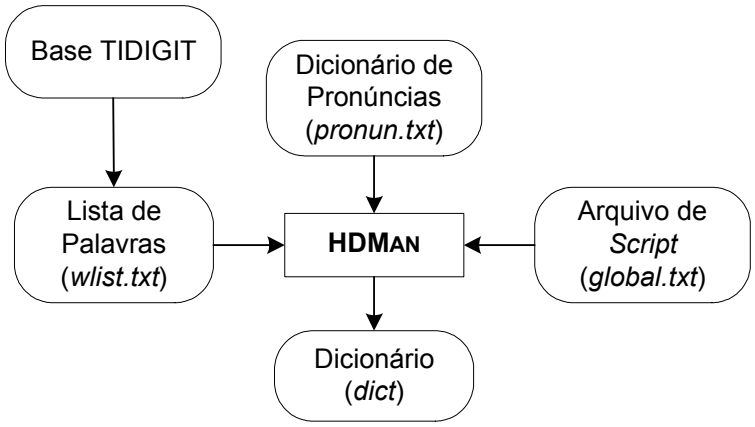

Figura 4.6: Geração de Dicionário (Adaptado de [YOUN 01]).

Primeiramente o arquivo *wlist.txt* é feito manualmente e contém uma lista de todas as palavras da gramática a serem reconhecidas em ordem alfabética como mostra a Figura 4.7.

```
EIGHT
FIVE
FOUR
NINE
OH
ONE
SENT-END
SENT-START
SEVEN
SIX
THREE
TWO
ZERO
```

Figura 4.7: Arquivo *wlist.txt*.

O segundo arquivo feito manualmente é o *pronun.txt* que deve conter as palavras a serem reconhecidas e suas correspondentes traduções a nível de palavras como apresentado na Figura 4.8. Os colchetes indicarão ao HTK que nada deverá ser escrito quando as palavras SENT-END e SENT-START forem reconhecidas. Se os colchetes forem omitidos significará que a própria palavra será escrita quando reconhecida. Portanto, para as entradas SENT-END e SENT-START o modelo de silêncio (*sil*) representará a pronúncia dos mesmos e não serão representados por nenhum símbolo na saída.

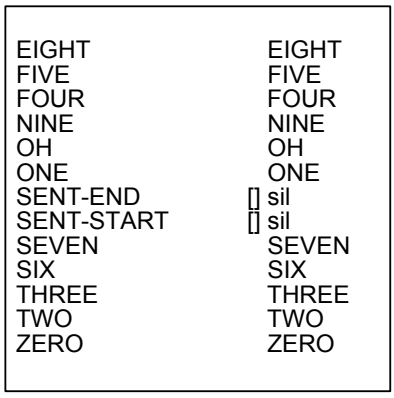

Figura 4.8: Arquivo *pronun.txt*.

O arquivo *global.txt*, também feito manualmente, é um arquivo de *script* que substitui *sil* e/ou *sp* (pausa curta) por *sil*. O conteúdo deste arquivo está na Figura 4.9.

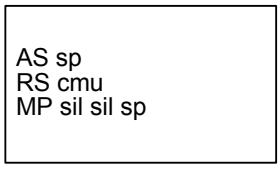

Figura 4.9: Arquivo *global.txt*.

A linha de comando a seguir é então digitada para executar a ferramenta **HDMan**.

HDMan −m −g global.txt −w wlist.txt −n words1 −l dlog dict pronun.txt

O argumento opcional **−m** vai agrupar o conteúdo das duas listas *wlist.txt* e *pronun.txt*, o argumento **−g** deve preceder o arquivo de *script global.txt*, o argumento **−w** carrega a lista armazenada no arquivo *wlist.txt*, o argumento **−n** escreve num arquivo de saída todas as palavras do vocabulário, inclusive silêncio e pausas num arquivo de saída *words*1 e a opção **−l** instrui a ferramenta **HDMan** a escrever num arquivo de saída *dlog* todas as estatísticas sobre a construção do dicionário indicando se há alguma palavra faltando ou não.

O arquivo *dict* é gerado pelo HTK. Ele é semelhante ao arquivo *pronun.txt* com o acréscimo de *sp* (pausas) no fim de cada palavra como mostra a Figura 4.10.

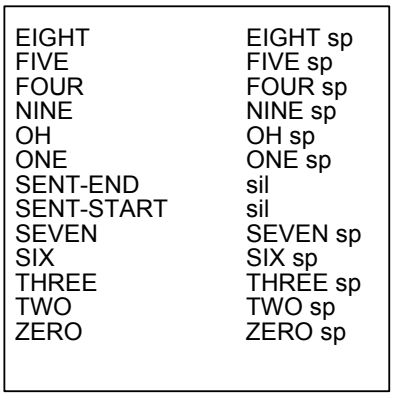

Figura 4.10: Arquivo *dict*.

O arquivo *dlog*, gerado pelo HTK, relata as estatísticas do dicionário gerado. A Figura 4.11 apresenta o conteúdo deste arquivo.

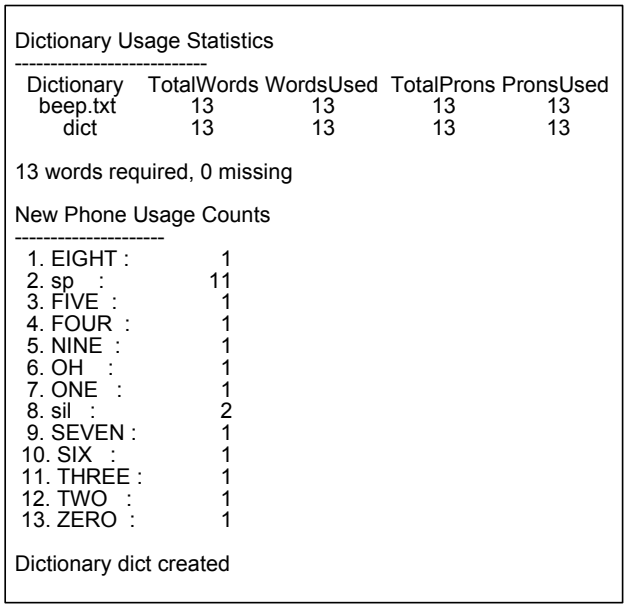

Figura 4.11: Arquivo *dlog*.

O arquivo *words*1 também é gerado pelo HTK e contém a lista de palavras usadas na contrução do dicionário inclusive pausas e silêncio como mostra a Figura 4.12.

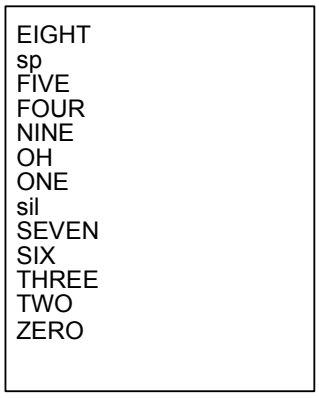

Figura 4.12: Arquivo *words*1.

Passo 3 - Gerando Arquivos de Transcrição MLF

A terceira etapa consistiu em gerar os arquivos MLF de transcrições de palavras, associados a cada arquivo da base de treinamento, necessários na fase de treinamento dos HMMs. O arquivo MLF é gerado através de um comando **Perl** que executa um arquivo de *script* (**prompts2mlf.2**) encontrado no diretório do Tutorial do HTK. A sigla **Perl** significa "*Practical Extraction And Report Language*". Essa linguagem é usada para criar programas em ambientes Unix, MS-DOS, Windows, Macintosh, OS/2 e em outros sistemas operacionais. É uma linguagem que tem funções muito eficientes para manipular textos, tornando-a muito popular para a programação de formulários WWW [PERL 05].

Feito isso, por meio da ferramenta **HLEd**, o arquivo MLF de transcrição em nível de palavras é gerado novamente com o acréscimo do silêncio no início e fim de cada sentença como mostra a Figura 4.13.

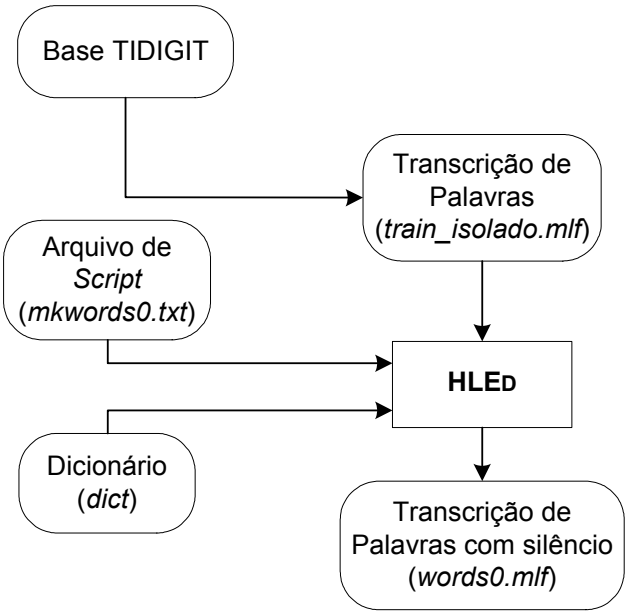

Figura 4.13: Transcrição de Palavras com Silêncio. (Adaptado de [YOUN 01]).

Antes de gerar os arquivos MLF faz-se necessária a construção de um arquivo *txt* contendo toda a base de treinamento (no caso de dígitos isolados) escrita por extenso, como mostra o arquivo *train_isolado.txt* a seguir, que é feito pelo usuário e apresenta as palavras pronunciadas de toda a base de treinamento como na Figura 4.14. Neste arquivo, a primeira linha contém um asterisco, seguido de uma barra e o nome do primeiro arquivo *wav* da base sem a extensão (*/1A_01f) e logo depois um espaço e a palavra pronunciada pelo locutor (ONE).

```
*/1A_01f ONE
*/1A_01m ONE
*/1A_02f ONE
*/1A_02m ONE

...

*/ZB_55f ZERO
*/ZB_55m ZERO
*/ZB_56f ZERO
*/ZB_57f ZERO
```

Figura 4.14: Arquivo *train_isolado.txt*.

Após a criação do arquivo contendo a base de treinamento por extenso, o comando **Perl** é executado.

Perl prompts2mlf.2 train_isolado.mlf train_isolado.txt

O arquivo *train_isolado.mlf* é gerado pelo HTK e contém os arquivos MLF ou *Master Label File* conforme apresentado na Figura 4.15. A primeira linha do arquivo identifica o próprio como um arquivo MLF. Quando o HTK for processar os arquivos de fala vai procurar por uma transcrição, ou arquivo MLF, com o mesmo nome do arquivo *wav*, mas com uma extensão diferente (*lab*). O asterisco e a barra indicam ao HTK que o diretório é indiferente. No arquivo MLF, após o nome do arquivo com a extensão *lab* segue a palavra pronunciada por extenso escrita em uma única linha e a sentença é finalizada com um ponto.

```
#!MLF!#
"*/1A_01f.lab"
ONE
.
"*/1A_01m.lab"
ONE
.


...

"*/ZB_56f.lab"
ZERO
.
"*/ZB_57f.lab"
ZERO
.
```

Figura 4.15: Arquivo *train_isolado.mlf*.

O arquivo de *script mkwords*0.*txt* também precisa ser criado, pois é usado pelo **HLEd** para inserir o silêncio no início e fim das sentenças. A Figura 4.16 apresenta o conteúdo deste arquivo onde o comando EX substitui cada palavra do arquivo *train_isolado.mlf* pela pronúncia correspondente no dicionário *dict*. O comando IS insere um modelo de silêncio no início e fim de cada sentença e o comando DE elimina todas as pausas curtas (*sp*).

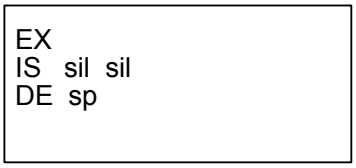

Figura 4.16: Arquivo $mkwords0.txt$.

A linha de comando a seguir é executada.

HLEd −l ∗ −d dict −i words0.mlf mkwords0.txt train_isolado.mlf

A opção **−l** é necessária para gerar o caminho '*' para armazenamento dos arquivos MLF, a opção **−d** carrega o dicionário do arquivo *dict*, o argumento opcional **−i** gera os arquivos de saída no arquivo MLF $words0.mlf$, que contém a transcrição de cada arquivo de saída com o acréscimo do silêncio como mostra a Figura 4.17.

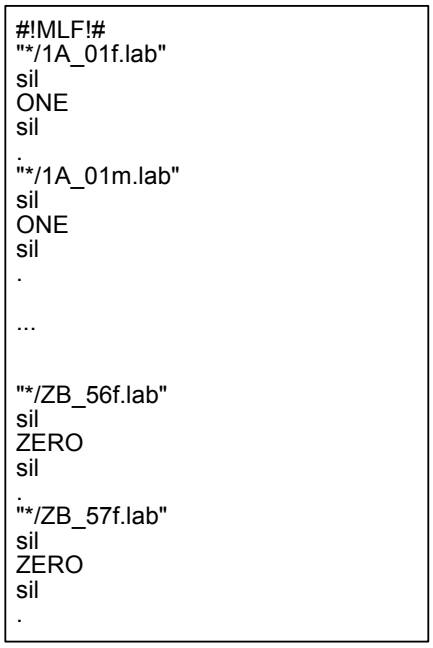

Figura 4.17: Arquivo $words0.mlf$.

Passo 4 - Codificando os Dados

A fase final de preparação dos dados consistiu em extrair os parâmetros do sinal de fala da base de treinamento. Para isso, utilizou-se a ferramenta **HCopy** para gerar os vetores

de coeficientes mel cepstrais (*Mel Frequency Cepstral Coefficients* - MFCCs) e energia (Figura 4.18). O mesmo procedimento foi utilizado para a extração dos parâmetros da base de teste. A análise espectral dos experimentos teve a seguinte configuração:

- Extração de 12 MFCCs mais o componente de energia, além de suas derivadas primeira (coeficientes delta) e segunda (coeficientes de aceleração), resultando num vetor de tamanho *trinta e nove*;
- Frames de $10ms$;
- Janelas de Hamming de $20ms$ e superposição de 50% e
- Fator de pré-ênfase $0,95$.

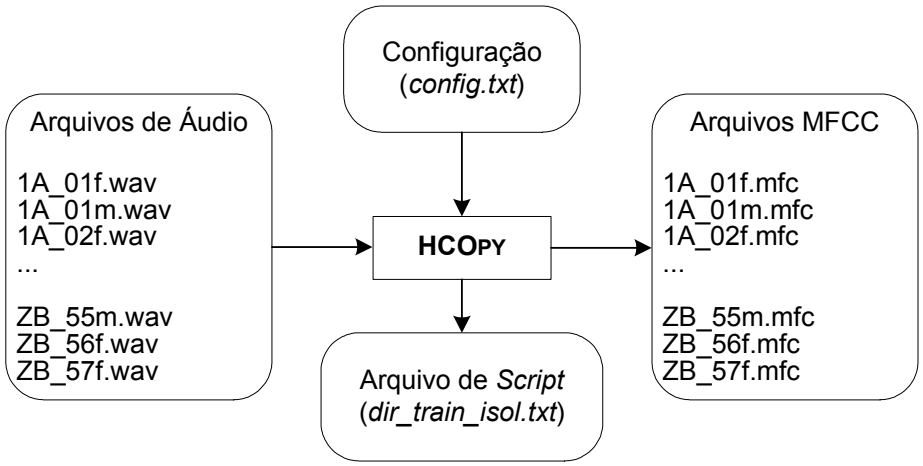

Figura 4.18: Geração de arquivos MFC. (Adaptado de [YOUN 01]).

Antes de executar a ferramenta **HCopy**, o arquivo *config.txt* deve ser feito manualmente, pois é um arquivo usado por esta ferramenta para especificar os parâmetros de conversão (Figura 4.19). O tipo de parâmetro é especificado por TARGETKIND (parâmetros mel ceptrais mais energia e derivadas primeira e segunda. O período do frame é de $10ms$, a saída é armazenada num formato comprimido. O modelo usa janelas de Hamming de $20ms$ e coeficiente de pré-ênfase de $0,95$, o banco de filtros tem 20 canais e *doze* coeficientes MFCC devem ser gerados. O tipo de fonte é um sinal em forma de onda de formato *wav*.

O arquivo *dir_train_isol.txt* contém o caminho ou diretório dos arquivos *wav* da base de treinamento e o caminho dos arquivos que serão criados pelo HTK, ou seja, os arquivos

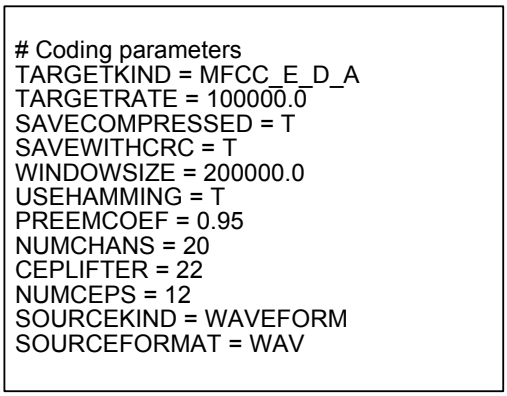

Figura 4.19: Arquivo *config.txt*.

MFC. A Figura 4.20 mostra o conteúdo deste arquivo. Ressalta-se que o caminho ou diretório onde os dois tipos de arquivos (*wav* e *mfc*) da base de treinamento de dígitos isolados se encontram devem ser indicados por "*dir_path_wav*" e "*dir_path_mfc*", respectivamente.

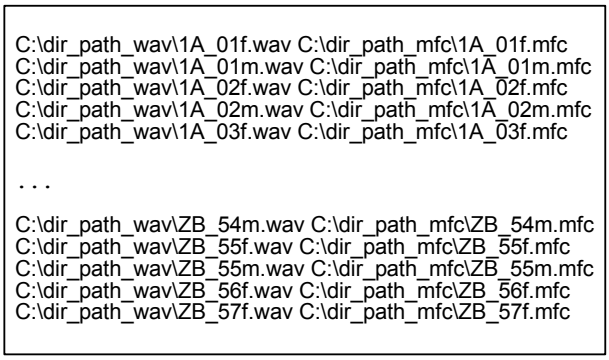

Figura 4.20: Arquivo *dir_train_isol.txt*.

A ferramenta **HCopy** é então executada pela seguinte linha de comando

HCopy −T 1 −C config.txt −S dir_train_isol.txt

A partir deste ponto todos os arquivos da base de treinamento de dígitos isolados foram transformados em arquivos *mfc* próprios do HTK no diretório indicado pelo arquivo *dir_train_isol.txt*. Na linha de comando acima, a opção **−T** configura os *flags* e a opção **−C** indica o arquivo de configuração.

Passo 5 - Treinando os Dados

O próximo estágio foi a criação de um conjunto bem treinado de modelos de palavras com 2 misturas de funções densidade de probabilidade Gaussiana. Neste exemplo, a topologia usada consiste em HMMs de *cinco* estados mais *dois* estados emissores e *dois* saltos na configuração *left-right* (Figura 4.21).

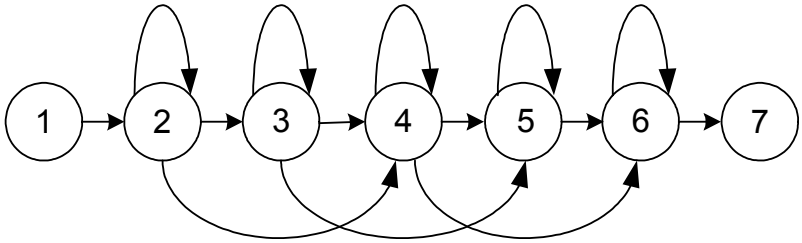

Figura 4.21: Estrutura *left-right* Utilizada no Exemplo com o HTK.

Cada estado possui inicialmente um vetor de médias nulas e uma matriz diagonal de variâncias unitárias. Com a ferramenta **HCompV** foram calculadas as médias e variâncias globais dos arquivos MFCCs da base de treinamento fazendo com que todos os HMMs ficassem com a mesma média e variância. Os HMMs de palavras foram então iniciados com a ferramenta **HInit**, re-estimados *uma* vez com a ferramenta **HRest** e *quinze* vezes com a ferramenta **HERest**.

A ferramenta **HCompV** necessita da construção do arquivo *proto.txt*.Este arquivo, feito manualmente, é um arquivo de protótipo importante na definição da topologia do modelo, neste caso *cinco* estados mais *dois* estados emissores (Figura 4.22). Este arquivo caracteriza o modelo "proto" que tem *sete* estados e *duas* misturas por estado. Como o primeiro e o último estados são emissores, o modelo especifica apenas os *cinco* estados internos, ou seja, do estado 2 ao 6. No estado 2 há *duas* misturas: a primeira tem coeficiente de peso igual a $0,5$; vetor média de tamanho *trinta e nove* e média *zero* e uma matriz de covariância diagonal unitária de tamanho *trinta e nove*. A segunda mistura também tem coeficiente de peso igual a $0,5$ e possui as mesmas características da primeira. O modelo segue com os parâmetros para todos os estados. A matriz de transição de estados é identificada por TransP.

```
~o <VecSize> 39 <MFCC_E_D_A>
~h "proto"
<BEGINHMM>
        <NumStates> 7
        <State> 2 <NumMixes> 2
                <Mixture> 1 0.5
                        <Mean> 39
                          0.0 0.0 0.0 0.0 0.0 0.0 0.0 ...
                        <Variance> 39
                          1.0 1.0 1.0 1.0 1.0 1.0 1.0 ...
                <Mixture> 2 0.5
                        <Mean> 39
                          0.0 0.0 0.0 0.0 0.0 0.0 0.0 ...
                        <Variance> 39
                          1.0 1.0 1.0 1.0 1.0 1.0 1.0 ...
        <State> 3 <NumMixes> 2
                <Mixture> 1 0.5
                        <Mean> 39
                          0.0 0.0 0.0 0.0 0.0 0.0 0.0 ...
                        <Variance> 39
                          1.0 1.0 1.0 1.0 1.0 1.0 1.0 ...
                <Mixture> 2 0.5
                        <Mean> 39
                          0.0 0.0 0.0 0.0 0.0 0.0 0.0 ...
                        <Variance> 39
                          1.0 1.0 1.0 1.0 1.0 1.0 1.0 ...
        ...

        <TransP> 7
          0.0 1.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.4 0.3 0.3 0.0 0.0 0.0
          0.0 0.0 0.4 0.3 0.3 0.0 0.0
          0.0 0.0 0.0 0.4 0.3 0.3 0.0
          0.0 0.0 0.0 0.0 0.4 0.3 0.3
          0.0 0.0 0.0 0.0 0.0 0.5 0.5
          0.0 0.0 0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Figura 4.22: Arquivo *proto.txt*.

O arquivo $dir\_train\_isol\_mfc.txt$ também é feito manualmente e contém o caminho de toda a base de treinamento de dígitos isolados que foi gerada pelo HTK (mfc). A Figura 4.23 mostra o arquivo $dir\_train\_isol\_mfc.txt$, onde "*dir_path_mfc*" deve indicar o diretório onde se encontram os arquivos mfc da base de treinamento de dígitos isolados.

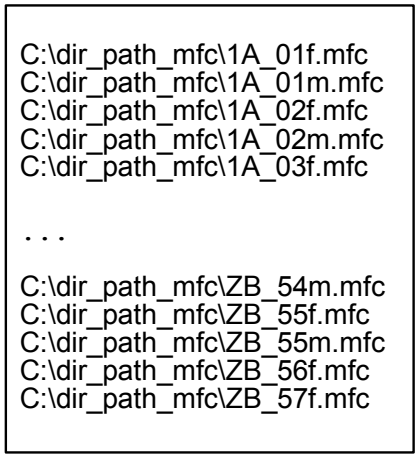

Figura 4.23: Arquivo *dir_train_isol_mfc.txt*.

O arquivo *config2.txt* é igual ao *config.txt*, descrito anteriormente, com as seguintes alterações: retirar as especificações SOURCEKIND e SOURCEFORMAT (Figura 4.24).

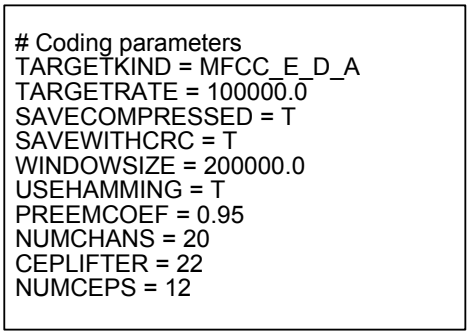

Figura 4.24: Arquivo *config2.txt*.

Antes de executar a ferramenta **HCompV** foi necessário ainda criar 16 pastas chamadas hmm0, hmm1, ..., hmm15 no diretório onde está sendo executado o programa.

A linha de comando correspondente a ferramenta **HCompV** fica

HCompV −C config2.txt −f 0.01 −m −S dir_train_isol_mfc.txt −M hmm0 proto.txt

O argumento opcional **−C** habilita o arquivo de configuração *config2.txt*, a opção **−f** possibilita gerar o arquivo *vFloors* com a variância global multiplicada por $0,01$. A opção **−m** atualiza as médias, a opção **−S** habilita o arquivo de *script dir_train_isol_mfc.txt* e a opção **−M** indica o diretório hmm0 onde os modelos devem ser armazenados.

Após a execução da ferramenta **HCompV**, o arquivo *proto.txt* é modificado pelo HTK, ou seja, as médias zero e as variâncias unitárias são substituídas pelas médias e variâncias globais da fala. Este arquivo é armazenado automaticamente na pasta hmm0, mas sem a extensão *.txt*. A Figura 4.25 mostra o arquivo *proto* modificado.

```
~o
<STREAMINFO> 1 39
<VecSize> 39  <NULLD< <MFCC_E_D_A>
~h "proto"
<BEGINHMM>
            <NumStates> 7
            <State> 2 <NumMixes> 2
                        <Mixture> 1 5.000000e-001
                                    <Mean> 39
                                     -4.410754e+000 -2.729313e+000    ...
                                    <Variance> 39
                                     3.577994e+001 4.662587e+001    ...
                                    <GCONST> 9.409501e+001
                        <Mixture> 2 0.5
                                    <Mean> 39
                                     -4.410754e+000 -2.729313e+000    ...
                                    <Variance> 39
                                     3.577994e+001 4.662587e+001    ...
                                    <GCONST> 9.409501e+001
            <State> 3 <NumMixes> 2
                        <Mixture> 1 5.000000e-001
                                    <Mean> 39
                                     -4.410754e+000 -2.729313e+000    ...
                                    <Variance> 39
                                     3.577994e+001 4.662587e+001    ...
                                    <GCONST> 9.409501e+001
                        <Mixture> 2 0.5
                                    <Mean> 39
                                     -4.410754e+000 -2.729313e+000    ...
                                    <Variance> 39
                                     3.577994e+001 4.662587e+001    ...
                                    <GCONST> 9.409501e+001
            ...

            <TransP> 7
             0.0 1.0 0.0 0.0 0.0 0.0 0.0
             0.0 0.4 0.3 0.3 0.0 0.0 0.0
             0.0 0.0 0.4 0.3 0.3 0.0 0.0
             0.0 0.0 0.0 0.4 0.3 0.3 0.0
             0.0 0.0 0.0 0.0 0.4 0.3 0.3
             0.0 0.0 0.0 0.0 0.0 0.5 0.5
             0.0 0.0 0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Figura 4.25: Arquivo *proto* gerado pelo HTK.

O arquivo $vFloors$ também é gerado pelo HTK após a execução da ferramenta **HCompV**. Neste arquivo, a variância fica igual a $0,01$ vezes a variância global. A Figura 4.26 apresenta o conteúdo do arquivo $vFloors$.

```
~v varFloor1
   <Variance> 39
      3.577994e-001 4.662587e-001 3.613866e-001 6.285893e-001 ...
```

Figura 4.26: Arquivo $vFloors$.

Neste ponto é preciso se fazer uma cópia do arquivo *proto* gerado pelo HTK e localizado na pasta hmm0 para o diretório onde está sendo executado o HTK.

Executa-se então a próxima linha de comando, **HInit**, que vai gerar um conjunto inicial de parâmetros usando a técnica *segmental k-means*.

HInit −e 0.001 −i 80 −T 1 −C config2.txt −S dir_train_isol_mfc.txt −M hmm1 proto

Na linha de comando acima, a opção **−e** determina o fator epsilon de convergência do modelo para $0,001$, a opção **−i** determina o máximo de iterações para 80, a opção **−T** habilita flags de traço para 1, a opção **−C** determina o arquivo de configuração como sendo $config2.txt$, a opção **−S** determina o arquivo de *script* como sendo $dir\_train\_isol\_mfc.txt$ e finalmente o argumento opcional **−M** indica o diretório hmm1 onde os modelos devem ser armazenados.

O arquivo *proto* é mais uma vez modificado pelo HTK. Ele é armazenado automaticamente na pasta hmm1. Deve-se copiá-lo para o diretório de execução do programa, substituindo o anterior.

Os parâmetros são então re-estimados através da ferramenta **HRest**, que utiliza o algoritmo de Baum-Welch. Para tal, executa-se a linha de comando.

HRest −e 0.001 −i 80 −T 1 −C config2.txt −S dir_train_isol_mfc.txt −M hmm2 proto

Os argumentos opcionais para a linha de comando anterior possuem as mesmas características daqueles especificados para a ferramenta **HInit**.

Da mesma forma, o arquivo *proto* é modificado pelo HTK e é armazenado automaticamente na pasta hmm2.

A próxima etapa é definir as características de cada modelo de palavra. Para isso, o arquivo $hmmdefs.txt$ é feito manualmente para ser usado pelo **HERest** que contém uma cópia do último protótipo *proto* (na pasta hmm2) para cada palavra requerida, que terá *cinco* estados, mais *dois* estados emissores totalizando *sete* estados, com exceção do modelo *sil* que terá *dois* estados mais *dois* emissores, totalizando *quatro*. Esse formato MMF (*Master Macro File*) é similar ao MLF e é construído manualmente copiando o protótipo (arquivo

proto mais recente localizado na pasta hmm2) e renomeando-o por cada palavra retirada de *words*1. Este arquivo *hmmdefs.txt* deve ser armazenado na pasta hmm0. A Figura 4.27 mostra como cada modelo de dígito é caracterizado por um *template*, ou modelo, retirado do último protótipo gerado. Observe que para especificar o modelo *EIGHT* por exemplo, bastou substituir o nome ~h "proto" por ~h "EIGHT" e copiar o resto das especificações do modelo. Isso foi feito para todos os dígitos.

```
~o
<STREAMINFO> 1 39
<VecSize> 39  <NULLD< <MFCC_E_D_A>
~h "EIGHT"
<BEGINHMM>
            <NUMSTATES> 7
            <STATE> 2 <NUMMIXES> 2
                        <MIXTURE> 1 5.464237e-001
                                    <MEAN> 39
            ...

            <TRANSP> 7
             0.000000e+000 1.000000e+000 0.000000e+000 ...
             0.000000e+000 9.590878e-001 4.091215e-002 ...
             0.000000e+000 0.000000e+000 9.218921e-001 ...
            ...

<ENDHMM>

~h "FIVE"
<BEGINHMM>
            <NUMSTATES> 7
            <STATE> 2 <NUMMIXES> 2
                        <MIXTURE> 1 5.464237e-001
                                    <MEAN> 39
            ...

            <TRANSP> 7
             0.000000e+000 1.000000e+000 0.000000e+000 ...
             0.000000e+000 9.590878e-001 4.091215e-002 ...
             0.000000e+000 0.000000e+000 9.218921e-001 ...
            ...

<ENDHMM>

...

~h "sil"
<BEGINHMM>
            <NUMSTATES> 4
            <STATE> 2 <NUMMIXES> 2
                        <MIXTURE> 1 5.464237e-001
                                    <MEAN> 39
            ...

            <TRANSP> 4
             0.000000e+000 1.000000e+000 0.000000e+000 ...
             0.000000e+000 9.590878e-001 4.091215e-002 ...
             0.000000e+000 0.000000e+000 9.218921e-001 ...
             0.000000e+000 0.000000e+000 0.000000e+000 ...

<ENDHMM>

...
```

Figura 4.27: Arquivo *hmmdefs.txt* (Adaptado de [YOUN 01]).

O arquivo *macros.txt* é idêntico ao *vFloors* da pasta hmm0 acrescentado de novas linhas de comando como mostrado na Figura 4.28. Este arquivo é usado pelo **HERest**. Ele deve ser armazenado na pasta hmm0.

```
~o
   <VecSize> 39
   <MFCC_0_D_A>
~v varFloor1
    <Variance> 39
       3.577994e-001 4.662587e-001 3.613866e-001 6.285893e-001 ...
```

Figura 4.28: Arquivo *macros*.

O arquivo *words*0.*txt*. Também é similar ao arquivo *words*1 menos o modelo de pausa curta *sp*. A Figura 4.29 apresenta este arquivo. Ele é usado pela ferramenta **HERest**.

```
EIGHT
FIVE
FOUR
NINE
OH
ONE
sil
SEVEN
SIX
THREE
TWO
ZERO
```

Figura 4.29: Arquivo *words*0.*txt*.

A partir deste momento, várias execuções da ferramenta **HERest** serão necessárias para a modelagem dos dígitos isolados. Este procedimento em detalhes é explanado no Apêndice A. Terminado o treinamento dos dígitos isolados, inicia-se o treinamento dos dígitos conectados.

O processo de treinamento dos dígitos conectados segue o mesmo raciocínio do treinamento dos dígitos isolados. Utilizando agora a base de treinamento de dígitos conectados, realiza-se os passos 3 e 4 para gerar os arquivos de transcrição MLF e codificar os dados. No passo 5 (treinamento) não é mais necessária a criação de arquivos protótipos, pois o arquivo *hmmdefs.txt* utilizado será aquele obtido no treinamento dos dígitos isolados assim como o arquivo *macros.txt*, ambos na pasta hmm15. A partir daí executa-se apenas a ferramenta **HERest** *sete* vezes, a ferramenta **HVite** *uma* vez e mais *oito* vezes a ferramenta **HERest** como no procedimento de treinamento dos dígitos isolados indicado no Apêndice A.

Passo 6 - Reconhecimento

Para o reconhecimento da base de teste de dígitos conectados é preciso gerar os arquivos de transcrição MLF através das ferramentas **Perl** e **HLEd** bem como codificar os dados através da ferramenta **HCopy** (passos 3 e 4). O procedimento é o mesmo, mudando apenas a base (teste) e os nomes dos arquivos feitos manualmente.

O arquivo *dir_teste_con_mfc.txt* é também feito manualmente e é usado pelo **HVite** para a base de teste. Similar ao *dir_train_isol_mfc.txt*, contém os caminhos de cada arquivo mfc criado anteriormente pelo HTK na fase de preparação dos dados (Figura 4.30). O termo "*dir_path_Test_mfc*" deve indicar o caminho ou diretório onde se encontram os arquivos mfc da base de teste de dígitos conectados.

```
C:\dir_path_Test_mfc\1115ZA_3311.mfc
C:\dir_path_Test_mfc\1119673A_4928.mfc
C:\dir_path_Test_mfc\111A_1.mfc
C:\dir_path_Test_mfc\111A_7854.mfc
C:\dir_path_Test_mfc\1121851A_4466.mfc

. . .

C:\dir_path_Test_mfc\ZZZ9139A_923.mfc
C:\dir_path_Test_mfc\ZZZ96A_4542.mfc
C:\dir_path_Test_mfc\ZZZ9A_2001.mfc
C:\dir_path_Test_mfc\ZZZ9Z52A_6544.mfc
C:\dir_path_Test_mfc\ZZZA_6159.mfc
```

Figura 4.30: Arquivo *dir_teste_con_mfc_txt*.

De posse dos arquivos de transcrição MLF da base de teste, do arquivo contendo o caminho de cada arquivo mfc chamado de *dir_teste_con_mfc.txt*, executa-se a ferramenta **HVite**, onde a opção **−H** carrega o arquivo *macro* do HMM, a opção **−l** indica o diretório onde armazenar os arquivos MLF, a opção **−i** determina a saída das transcrições para o arquivo MLF $recout.mlf$, a opção **−w** permite reconhecer a partir da rede de palavras *wdnet*, a opção **−p** habilita um valor para insersões de palavras e a opção **−s** indica que o fator de escala da gramática é igual a $5,0$.

HVite −H hmm15/macros.txt −H hmm15/hmmdefs.txt −S dir_teste_con_mfc .txt −l ∗ −i recout.mlf −w wdnet −p 0.0 −s 5.0 dict1.txt words0.txt

O arquivo $recout.mlf$ é então gerado pelo HTK e contém todos arquivos de teste reconhecidos e suas transcrições. Assemelha-se a Figura 4.31.

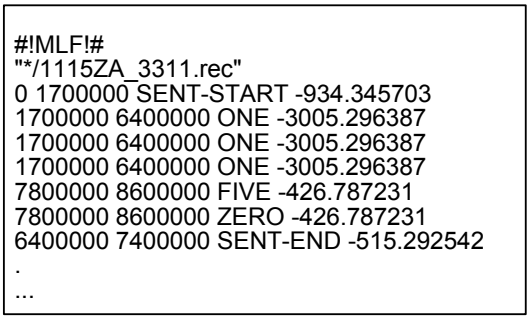

Figura 4.31: Arquivo *recout.mlf*.

Em seguida, o arquivo *trans_test.txt* é criado, devendo conter todas as sentenças da base de teste de acordo com o formato exigido pelo HTK. A Figura 4.32 mostra como é esse arquivo.

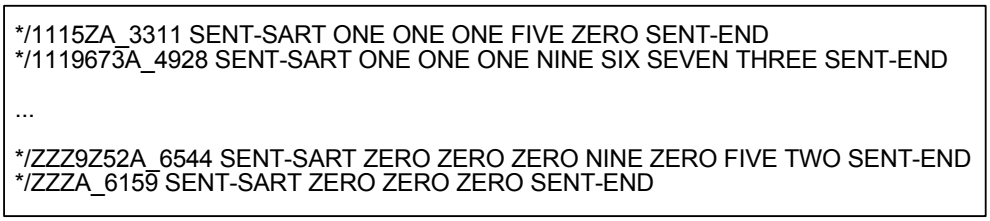

Figura 4.32: Arquivo *trans_test.txt*.

De posse do arquivo *trans_test.txt*, a seguinte linha de comando é executada.

Perl prompts2mlf.2 testref.mlf trans_test.txt

O arquivo *testref.mlf* é gerado pelo HTK e contém as sentenças exatas da base de teste que serão usadas como referência na avaliação do desempenho do reconhecimento. A Figura 4.33 mostra esse tipo de arquivo. Neste caso, a sentença é iniciada por SENT-START e terminada por SENT-END, e cada palavra de uma sentença é colocada em uma única linha até o fim da mesma.

Finalmente, realiza-se uma análise dos resultados do reconhecimento armazenados no arquivo *recout.mlf*.

HResults −I testref.mlf words0.txt recout.mlf > resultado.txt

As taxas de reconhecimento e erro são obtidas diretamente no arquivo *resultado.txt*

```
#!MLF!#
"*/1115ZA_3311.lab"
SENT-START
ONE
ONE
ONE
FIVE
ZERO
SENT-END
.
...

"*/ZZZA_6159.lab"
SENT-START
ZERO
ZERO
ZERO
SENT-END
.
```

Figura 4.33: Arquivo $testref.mlf$.

(Figura 4.34) ou no *prompt* do MS-DOS se omitido "> **resultado.txt"** da linha de comando acima. A opção $-\mathbf{I}$ carrega o arquivo MLF $testref.mlf$.

O resultado da análise indicado na Figura 4.34 apresenta data e hora de quando foi feita a análise e especifica os nomes dos arquivos de referência e os utilizados no reconhecimento ($testref.mlf$ e $recout.mlf$ respectivamente). A linha que começa por SENT mostra as estatísticas do desempenho obtido para as sentenças. Neste caso, de um total de 8700 sentenças de teste, foram reconhecidas $69,15\%$ das mesmas, ou seja, 6016 sentenças.

```
====================== HTK Results Analysis =======================
  Date: Tue Jan 15 19:04:15 2005
  Ref : testref.mlf
  Rec : recout.mlf
------------------------------------ Overall Results ----------------------------------------------
SENT: %Correct=69.15 [H=6016, S=2684, N=8700]
WORD: %Corr=99.16, Acc=91.63 [H=45598, D=32, S=353, I=3466, N=45983]
============================================================
```

Figura 4.34: Arquivo $resultado.txt$.

Na linha que começa por WORD indica as estatísticas para o reconhecimento de palavras. De 45983 palavras no total, $99,16\%$ (ou 45598 palavras) foram reconhecidas corretamente. Durante o reconhecimento 32 palavras foram suprimidas (D), 353 foram substituídas (S) e 3466 foram inseridas (I). Esses erros deveram-se principalmente ao fato do tamanho de cada sentença ser desconhecido para o sistema e talvez pelo número reduzido de épocas de treinamento dos dígitos conectados (apenas 15 iterações). O valor indicado por Acc ($91,63\%$) indica a porcentagem total de acerto de palavras levando em consideração os erros

de inserção de palavras, visto que a primeira estatística de $99,16\%$ não considera este tipo de erro.

Capítulo 5

Resultados Experimentais

5.1 Introdução

Diversos experimentos foram realizados com o novo sistema empregando HMMs contínuos a fim de se observar o desempenho deste em relação ao sistema que emprega HMMs discretos [ANDR 01], como também em relação ao HTK [HTK 93].

Duas bases de dados foram utilizadas para a realização dos experimentos: uma pequena base de dados em português brasileiro, gravada no próprio laboratório, chamada de LPDF Dígitos e uma base de dígitos conectados do inglês americano, chamada de TIDIGITS.

5.2 Bases de Dados

As bases de dados empregadas em sistemas de reconhecimento de fala possuem sua parcela de importância no desempenho do mesmo além dos algoritmos utilizados no decorrer do processo, pois de acordo com as características peculiares de cada base, principalmente em termos de tamanho, diversidade regional e qualidade de produção, os resultados finais podem mudar consideravelmente.

Nos tópicos a seguir, as duas bases de dados empregadas são detalhadas, destacando-se os números de locutores, tipos de sentenças produzidas e quantidade de elocuções gravadas por cada locutor.

5.2.1 LPDF Dígitos

A base de dados em português brasileiro, LPDF Dígitos, é dividida em três partes: uma base para treinamento de dígitos isolados, uma base para treinamento de dígitos conectados e outra para testes de dígitos conectados. A base de treinamento de dígitos isolados é composta por *quatro* locutores femininos e *cinco* locutores masculinos. A base de treinamento de dígitos conectados é composta por outros diferentes locutores, *treze* femininos e *dezoito* masculinos. Já a base de teste de dígitos conectados é formada por *quatro* locutores femininos e *cinco* masculinos diferentes daqueles. As bases foram gravadas em ambiente de escritório com locutores diferentes para todas as bases. A base LPDF Dígitos é composta por locutores da maioria das regiões do Brasil: Norte (Pará), Nordeste (Ceará), Sudeste (São Paulo) e Sul (Rio Grande do Sul).

O dicionário é composto por 11 palavras: "*zero*", "*um*", "*dois*", "*três*", "*quatro*", "*cinco*", "*seis*", "*sete*", "*oito*", "*nove*" e o "*meia*". A base de treinamento composta por dígitos isolados totaliza 990 elocuções. Cada um dos *nove* locutores realizou *dez* repetições de cada dígito. Nas bases de dígitos conectados todas as frases pronunciadas contêm uma seqüência de *oito* dígitos aleatórios, alguns deles repetidos ou não. Cada locutor pronunciou *onze* frases, que totalizou um conjunto de treinamento com 341 elocuções (2728 palavras) e um conjunto de 99 elocuções (792 palavras) para a base de teste. Em todas as bases os dígitos isolados e conectados foram lidos.

5.2.2 TIDIGITS

A segunda base utilizada foi a TIDIGITS que consiste de uma base de dígitos conectados do inglês americano independente de locutor. É uma base vasta contendo os principais dialetos dos Estados Unidos [LEON 84] e composta por um total de 326 locutores entre homens, mulheres e crianças.

As frases pronunciadas por cada locutor consistem em sentenças de dígitos. O dicionário é formado por *onze* dígitos: "*zero*", "*one*", "*two*", "*three*", "*four*", "*five*", "*six*", "*seven*", "*eight*", "*nine*" e "*oh*". Cada locutor gravou aleatoriamente 77 seqüências lidas que foram divididas em um conjunto de teste e treinamento. As seqüências são dispostas da seguinte forma:

- 22 sentenças de um único dígito (2 repetições para cada um dos 11 dígitos);
- 11 seqüências de dois dígitos;
- 11 seqüências de três dígitos;
- 11 seqüências de quatro dígitos;
- 11 seqüências de cinco dígitos e
- 11 seqüências de sete dígitos.

Para estes experimentos apenas o corpus da base constituída por adultos (homens e mulheres) foi utilizado. O conjunto de treinamento, composto por 55 homens e 57 mulheres, soma um total de 8.623 sentenças (28336 palavras) e o conjunto de teste, composto por 56 homens e 57 mulheres, um total de 8.700 sentenças (28589 palavras).

5.3 Experimentos e Resultados

Os experimentos realizados com as duas bases foram comparados com os resultados obtidos em [ANDR 01] para verificar que tipo de modelo de palavra, contínuo ou discreto, seria mais adequado para o reconhecimento de dígitos conectados. Em todos os experimentos, os modelos de dígitos conectados foram treinados com a base de treinamento através do algoritmo *segmental k-means* [RABI 93] utilizando a segunda forma de treinamento, ou seja, sem utilizar a informação dos templates de dígitos isolados durante o treinamento dos dígitos conectados, como explanado no capítulo 3. Os experimentos seguintes consistiram em avaliar o desempenho dos modelos contínuos ao se variar a quantidade de estados por modelo e também o número de misturas. As duas formas de treinamento dos dígitos conectados também foram comparadas.

Durante o treinamento das sentenças, o modelo de silêncio sempre foi considerado no início e fim de cada elocução não levando-se em consideração pausas entre os dígitos. Os modelos de silêncio utilizados em todos os experimentos possuem *dois* estados e *dois* saltos.

Na comparação de desempenho de todos os modelos calculou-se também o intervalo de confiança dos resultados. O intervalo de confiança é de 95% em torno da média, parametriza-

do por uma PDF gaussiana dado por

$$\pm 1,96\sqrt{T_a\frac{(100-T_a)}{N_t}},$$

onde T_a é a taxa de acerto em % e N_t é o número total de exemplos. Essa equação vale para uma série binomial.

No reconhecimento realizado com as duas bases utilizou-se o algoritmo decodificador *Level Building* que, em geral, apresentou um desempenho melhor que o algoritmo *Frame Synchronous*, embora o tempo de processamento de cada frase fosse maior. Com exceção dos experimentos realizados com o HTK, todos os outros resultados levaram em conta que o tamanho das sentenças (número de dígitos) era conhecido.

5.3.1 Experimentos com a Base LPDF Dígitos

Nos experimentos a seguir os erros de reconhecimento deveram-se unicamente a substituições de palavras. Não foram identificados acréscimos ou supressões de palavras em nenhuma sentença. Os erros mais freqüentes foram substituições da palavra "*três*" pela palavra "*seis*" e vice-versa, como também da palavra "*dois*" pela palavra "*oito*". Nas tabelas e gráficos apresentados, os resultados obtidos com a base LPDF Dígitos mostram taxas de acerto de palavras ou sentenças.

Primeiro Experimento

O melhor desempenho de reconhecimento de palavras e sentenças da base de treinamento em português usando HMMs contínuos (HMMC) foi obtido com o treinamento de modelos com *dez* estados e *oito* misturas. Como mostra a Tabela 5.1, considerando um intervalo de confiança de 95%, observa-se que o desempenho entre HMMs contínuos e discretos [ANDR 01] é muito próximo quando utilizada a própria base de treinamento de dígitos conectados no processo de reconhecimento de palavras e sentenças. A diferença entre os dois modelos é muito pequena e não se pode afirmar qual sistema tem melhor desempenho.

Tabela 5.1: Desempenho entre HMMC e HMMD para a Base de Treinamento LPDF Dígitos.

LPDF Dígitos - Base de Treinamento				
Modelo	Acerto de Palavras		Acerto de Sentenças	
HMM Cont. (10 est. e 8 mist.)	$99,74\%$	$\pm 0,19\%$	$97,36\%$	$\pm 1,70\%$
HMM Discreto	$99,89\%$	$\pm 0,12\%$	$99,12\%$	$\pm 0,99\%$

Segundo Experimento

Para a base de teste, a melhor taxa de reconhecimento foi obtida com modelos contínuos de *quatro* estados por fonema e *duas* misturas. A Tabela 5.2 mostra uma diferença entre os modelos contínuos e discretos de cerca de $1,09\%$ no reconhecimento de palavras e da ordem de 10% no reconhecimento de sentenças da base de teste, mas devido ao intervalo de confiança de 95% não há como afirmar qual sistema apresenta o melhor desempenho pois há uma região em que os valores têm a probabilidade de coincidirem.

Tabela 5.2: Desempenho entre HMMC e HMMD para a Base de Teste LPDF Dígitos.

LPDF Dígitos - Base de Teste				
Modelo	Acerto de Palavras		Acerto de Sentenças	
HMM Cont. (4 est./fon. e 2 mist.)	$98,28\%$	$\pm 0,91\%$	$84,85\%$	$\pm 7,06\%$
HMM Discreto	$99,37\%$	$\pm 0,55\%$	$94,95\%$	$\pm 4,31\%$

Terceiro Experimento

No terceiro experimento com a base LPDF Dígitos, fixou-se o número de estados de cada modelo contínuo de dígitos conectados em *dez* e variou-se apenas o número de misturas. A Tabela 5.3 mostra a taxa de reconhecimento de palavras e sentenças da base de treinamento e a Tabela 5.4 apresenta as taxas obtidas no reconhecimento de palavras e sentenças da base de teste. Observa-se que, para as duas bases, a taxa de reconhecimento tende a aumentar a medida que aumentamos o número de misturas dentro de cada modelo. Considerando o intervalo de confiança, pode-se observar que o modelo com *oito* misturas apresenta um desempenho melhor que o modelo com *duas* misturas quando utilizada a base de treinamento. O mesmo não se pode afirmar ao se comparar os modelos com *quatro* e *oito* misturas

por possuírem valores que podem coincidir dentro do mesmo intervalo de confiança. Esse comportamento é observado tanto no reconhecimento de palavras como no de sentenças.

Já nos experimentos com a base de teste, tanto para o reconhecimento de palavras como para o de sentenças, não há grande diferença de desempenho entre as três topologias de modelo, haja vista a probabilidade de coincidência desses valores nos intervalos de confiança.

Tabela 5.3: Resultado da Base de Treinamento LPDF Dígitos para Modelos com 10 Estados.

LPDF Dígitos - Base de Treinamento (10 estados)				
Número de Misturas	Acerto de Palavras		Acerto de Sentenças	
2	$98,97\%$	$\pm 0,38\%$	$90,91\%$	$\pm 3,05\%$
4	$99,47\%$	$\pm 0,27\%$	$94,72\%$	$\pm 2,37\%$
8	$99,74\%$	$\pm 0,19\%$	$97,36\%$	$\pm 1,70\%$

Tabela 5.4: Resultado da Base de Teste LPDF Dígitos para Modelos com 10 Estados.

LPDF Dígitos - Base de Teste (10 estados)				
Número de Misturas	Acerto de Palavras		Acerto de Sentenças	
2	$96,16\%$	$\pm 1,34\%$	$67,68\%$	$\pm 9,21\%$
4	$97,17\%$	$\pm 1,15\%$	$77,78\%$	$\pm 8,19\%$
8	$97,68\%$	$\pm 1,05\%$	$79,80\%$	$\pm 7,91\%$

Quarto Experimento

O experimento seguinte consistiu em fixar o número de misturas ($m = 2$) e variar o número de estados de acordo com a quantidade de fonemas de cada palavra do vocabulário, ou seja, de cada dígito. O modelo do dígito 9 com *dois* estados por fonema, por exemplo, teve *oito* estados. A Tabela 5.5 mostra um crescimento da taxa de reconhecimento de palavras e sentenças para a base de treinamento e a Tabela 5.6 apresenta o mesmo comportamento para a base de teste à medida em que se aumenta o número de estados por fonema de cada modelo. Para as duas bases, avaliando-se o intervalo de confiança, observa-se que o modelo com *quatro* estados por fonema e *duas* misturas apresenta desempenho superior no reconhecimento de palavras apenas em relação ao modelo com *um* estado por fonema e *duas* misturas, não

sendo possível identificar o mesmo comportamento de superioridade para o reconhecimento de sentenças onde todos os modelos apresentam valores de desempenho dentro da mesma faixa de confiança.

Tabela 5.5: Resultados da Base de Treinamento LPDF Dígitos para Modelos com 2 Misturas.

LPDF Dígitos - Base de Treinamento (2 misturas)				
Número de Estados por Fonema	Acerto de Palavras		Acerto de Sentenças	
1	$98,06\%$	$\pm 0,52\%$	$86,51\%$	$\pm 3,63\%$
2	$98,91\%$	$\pm 0,39\%$	$89,15\%$	$\pm 3,30\%$
3	$99,00\%$	$\pm 0,37\%$	$90,03\%$	$\pm 3,18\%$
4	$99,03\%$	$\pm 0,37\%$	$90,32\%$	$\pm 3,14\%$

Tabela 5.6: Resultados da Base de Teste LPDF Dígitos para Modelos com 2 Misturas.

LPDF Dígitos - Base de Teste (2 misturas)				
Número de Estados por Fonema	Acerto de Palavras		Acerto de Sentenças	
1	$95,96\%$	$\pm 1,37\%$	$68,69\%$	$\pm 9,14\%$
2	$96,46\%$	$\pm 1,29\%$	$68,69\%$	$\pm 9,14\%$
3	$97,58\%$	$\pm 1,07\%$	$77,78\%$	$\pm 8,19\%$
4	$98,28\%$	$\pm 0,91\%$	$84,85\%$	$\pm 7,06\%$

Quinto Experimento

No processo de treinamento de dígitos conectados foram destacadas *duas* formas de treinamento através do algoritmo *segmental k-means* [RABI 93]. A primeira consistia em determinar os vetores média, matriz de covariância e coeficientes de peso dos modelos de cada conjunto de *tokens* (obtidos após segmentação automática das frases), utilizando as informações dos *templates* da base de dígitos isolados usados na segmentação das sentenças de dígitos conectados. Na segunda forma de treinamento esses dados eram obtidos diretamente dos *tokens* através de uma segmentação uniforme em estados das seqüências de observação.

As tabelas seguintes apresentam as diferenças de desempenho do sistema utilizando essas duas formas de treinamento. A Tabela 5.7 apresenta os resultados para a base de treinamento

onde se verifica que os modelos de *cinco* estados e *quatro* misturas treinados a partir dos *templates* de dígitos isolados apresentam uma superioridade de desempenho em relação ao outro tipo de treinamento de $0,56\%$ para o reconhecimento de palavras e de $2,35\%$ para o de sentenças. A Tabela 5.8 mostra os resultados com a base de teste utilizando *cinco* estados e *quatro* misturas em cada modelo, onde observou -se o mesmo desempenho no reconhecimento de palavras e sentenças. Observando-se o intervalo de confiança calculado para os dois modelos com diferentes tipos de treinamento, pode-se inferir que apresentam valores de desempenho dentro da mesma faixa de confiança e, portanto, não há como afirmar qual tipo de treinamento é melhor.

Tabela 5.7: Resultados da Base de Treinamento com Modelos de 5 Estados e 4 Misturas.

LPDF Dígitos - Base de Treinamento (5 estados e 4 misturas)				
Tipo de Treinamento	Acerto de Palavras		Acerto de Sentenças	
Com *Templates* de Dígitos Isolados	$99,18\%$	$\pm 0,34\%$	$92,67\%$	$\pm 2,77\%$
Sem *Templates* de Dígitos Isolados	$98,62\%$	$\pm 0,44\%$	$90,32\%$	$\pm 3,14\%$

Tabela 5.8: Resultados da Base de Teste com Modelos de 5 Estados e 4 Misturas.

LPDF Dígitos - Base de Teste (5 estados e 4 misturas)				
Tipo de Treinamento	Acerto de Palavras		Acerto de Sentenças	
Com *Templates* de Dígitos Isolados	$96,16\%$	$\pm 1,34\%$	$71,72\%$	$\pm 8,87\%$
Sem *Templates* de Dígitos Isolados	$96,16\%$	$\pm 1,34\%$	$71,72\%$	$\pm 8,87\%$

Sexto Experimento

O experimento anterior foi repetido para modelos com um número maior de estados (*dez* estados) e mantendo o mesmo número de *quatro* misturas. A Tabela 5.9, contendo os resultados para a base de treinamento, apresenta as taxas de acerto obtidas com modelos treinados pelo primeiro método, ou seja, usando as informações de vetores média, matriz de covariância e coeficientes de peso diretamente dos *templates* dos dígitos isolados, que são significativamente menores que os resultados apresentados na Tabela 5.7, principalmente quando avaliadas as sentenças (taxa de acerto $36,66\%$ menor). A Tabela 5.9 também mostra que os modelos treinados da segunda maneira apresentaram um desempenho melhor que os modelos com

cinco estados mostrados na Tabela 5.7. O mesmo ocorre quando se usa a base de teste para avaliação de desempenho de reconhecimento, como mostrado na Tabela 5.10, onde a taxa de acerto de sentenças para modelos treinados com os *templates* dos dígitos isolados é $40,41\%$ menor em relação a taxa obtida na Tabela 5.8.

Calculando-se o intervalo de confiança de 95% para os resultados obtidos com os modelos de *dez* estados e *quatro* misturas da base de treinamento e teste (tabelas 5.9 e 5.10), pode-se afirmar que o modelo treinado sem as informações de dígitos isolados apresenta desempenho significativamente superior em relação ao modelo treinado pelo primeiro método.

Tabela 5.9: Resultados da Base de Treinamento com Modelos de 10 Estados e 4 Misturas.

LPDF Dígitos - Base de Treinamento (10 estados e 4 misturas)				
Tipo de Treinamento	Acerto de Palavras		Acerto de Sentenças	
Com *Templates* de Dígitos Isolados	$83,46\%$	$\pm 1,39\%$	$56,01\%$	$\pm 5,27\%$
Sem *Templates* de Dígitos Isolados	$99,47\%$	$\pm 0,27\%$	$94,72\%$	$\pm 2,37\%$

Tabela 5.10: Resultados da Base de Teste com Modelos de 10 Estados e 4 Misturas.

LPDF Dígitos - Base de Teste (10 estados e 4 misturas)				
Tipo de Treinamento	Acerto de Palavras		Acerto de Sentenças	
Com *Templates* de Dígitos Isolados	$69,19\%$	$\pm 3,22\%$	$31,31\%$	$\pm 9,14\%$
Sem *Templates* de Dígitos Isolados	$97,17\%$	$\pm 1,15\%$	$77,78\%$	$\pm 8,19\%$

5.3.2 Experimentos com a Base TIDIGITS

Com exceção dos experimentos realizados na comparação com o sistema HTK, todos os erros de reconhecimento foram devido a substituições de palavras, não sendo identificados acréscimos de palavras em nenhuma sentença tampouco palavras suprimidas. Os erros mais freqüentes eram substituições da palavra "*five*" pela palavra "*nine*" e vice-versa e da palavra "*zero*" pela palavra "*oh*" no início das frases. Nas tabelas e gráficos abaixo os resultados obtidos com a base TIDIGITS apresentam taxas de acerto de palavras ou sentenças.

Primeiro Experimento

Com a base TIDIGITS do inglês americano, a melhor taxa de reconhecimento de palavras e sentenças usando HMMs contínuos (HMMC) foi obtida com o treinamento de modelos de *cinco* estados e *oito* misturas. A Tabela 5.11 mostra uma comparação com os modelos discretos obtidos em [ANDR 01]. Considerando o intervalo de confiança de 95% calculado, observa-se que o desempenho no reconhecimento de dígitos conectados com HMMs contínuos usando a base de treinamento foi inferior ao obtido com HMMs discretos. Embora a diferença de desempenho entre os dois tipos de modelos no reconhecimento de palavras tenha sido apenas da ordem de $0,23\%$, a diferença no reconhecimento de sentenças foi um pouco mais acentuada ($1,6\%$).

Tabela 5.11: Desempenho entre HMMC e HMMD para a Base de Treinamento - TIDIGITS.

TIDIGITS - Base de Treinamento				
Modelo	Acerto de Palavras		Acerto de Sentenças	
HMM Cont. (5 est. e 8 mist.)	$99,35\%$	$\pm 0,09\%$	$97,14\%$	$\pm 0,35\%$
HMM Discreto	$99,58\%$	$\pm 0,07\%$	$98,74\%$	$\pm 0,24\%$

Segundo Experimento

Usando a base de teste, o melhor desempenho no reconhecimento de dígitos conectados também foi obtido com modelos contínuos de *cinco* estados e *oito* misturas. Na Tabela 5.12 verifica-se uma diferença da ordem de $0,31\%$ entre os modelos contínuos e discretos no reconhecimento de palavras e cerca de $2,39\%$ no reconhecimento de sentenças. Embora o desempenho com modelos discretos tenha sido melhor que os modelos contínuos, mesmo considerando o intervalo de confiança de 95%, não se verifica uma diferença muito acentuada entre os mesmos.

Tabela 5.12: Desempenho entre HMMC e HMMD para a Base de Teste - TIDIGITS.

TIDIGITS - Base de Teste				
Modelo	Acerto de Palavras		Acerto de Sentenças	
HMM Cont. (5 est. e 8 mist.)	$99,00\%$	$\pm 0,12\%$	$95,54\%$	$\pm 0,43\%$
HMM Discreto	$99,31\%$	$\pm 0,10\%$	$97,93\%$	$\pm 0,30\%$

Terceiro Experimento

A Tabela 5.13 mostra como evolui a taxa de reconhecimento de palavras e sentenças da base de treinamento quando o número de estados de cada modelo é fixado em *cinco* e variamos apenas o número de misturas. Verificou-se que a taxa de reconhecimento tende a um desempenho melhor a medida que aumentamos o número de misturas dentro de cada modelo mesmo considerando o intervalo de confiança. O mesmo comportamento é observado para o reconhecimento de palavras e sentenças da base de teste (Tabela 5.14).

Tabela 5.13: Resultados da Base de Treinamento TIDIGITS para Modelos com 5 Estados.

TIDIGITS - Base de Treinamento (5 estados)				
Número de Misturas	Acerto de Palavras		Acerto de Sentenças	
2	$98,59\%$	$\pm 0,14\%$	$94,18\%$	$\pm 0,49\%$
4	$99,06\%$	$\pm 0,11\%$	$96,05\%$	$\pm 0,41\%$
8	$99,35\%$	$\pm 0,09\%$	$97,14\%$	$\pm 0,35\%$

Tabela 5.14: Resultados da Base de Teste TIDIGITS para Modelos com 5 Estados.

TIDIGITS - Base de Teste (5 estados)				
Número de Misturas	Acerto de Palavras		Acerto de Sentenças	
2	$98,04\%$	$\pm 0,16\%$	$91,60\%$	$\pm 0,58\%$
4	$98,65\%$	$\pm 0,13\%$	$94,15\%$	$\pm 0,49\%$
8	$99,00\%$	$\pm 0,12\%$	$95,54\%$	$\pm 0,43\%$

Quarto Experimento

Neste experimento, o número de misturas ($m = 2$) foi fixado e variou-se o número de estados de acordo com a quantidade de fonemas de cada palavra do vocabulário. A Tabela 5.15 mostra que a taxa de reconhecimento de palavras e sentenças para a base de treinamento aumenta a medida que o número de estados por fonema aumenta. A Tabela 5.16 também apresenta o mesmo comportamento para o reconhecimento de palavras e sentenças da base de teste. Considerando o intervalo de confiança pode-se inferir que os modelos com *dois*, *três* e *quatro* estados por fonema apresentam valores de desempenho dentro da mesma faixa de precisão. Comparando o modelo com *quatro* estados por fonema com os outros modelos observa-se que há um ganho de desempenho significativo apenas em relação ao modelo com *um* estado por fonema (cerca de $1,28\%$ no reconhecimento de palavras e $4,61\%$ no reconhecimento de sentenças da base treinamento e, respectivamente, $1,54\%$ e $5,75\%$ no reconhecimento de palavras e sentenças da base de teste).

Tabela 5.15: Resultados da Base de Treinamento TIDIGITS para Modelos com 2 Misturas.

TIDIGITS - Base de Treinamento (2 misturas)				
Número de Estados por Fonema	Acerto de Palavras		Acerto de Sentenças	
1	$97,90\%$	$\pm 0,17\%$	$91,71\%$	$\pm 0,58\%$
2	$98,94\%$	$\pm 0,12\%$	$95,20\%$	$\pm 0,45\%$
3	$99,10\%$	$\pm 0,11\%$	$96,03\%$	$\pm 0,41\%$
4	$99,18\%$	$\pm 0,11\%$	$96,32\%$	$\pm 0,40\%$

Tabela 5.16: Resultados da Base de Teste TIDIGITS para Modelos com 2 Misturas.

TIDIGITS - Base de Teste (2 misturas)				
Número de Estados por Fonema	Acerto de Palavras		Acerto de Sentenças	
1	$97,24\%$	$\pm 0,19\%$	$88,79\%$	$\pm 0,66\%$
2	$98,50\%$	$\pm 0,14\%$	$93,44\%$	$\pm 0,52\%$
3	$98,68\%$	$\pm 0,13\%$	$94,25\%$	$\pm 0,49\%$
4	$98,78\%$	$\pm 0,13\%$	$94,54\%$	$\pm 0,48\%$

Quinto Experimento

Neste experimento, analisou-se o desempenho do sistema utilizando as duas formas de treinamento explanadas no capítulo 3. A Tabela 5.17 apresenta os resultados para a base de treinamento de modelos com *cinco* estados e *quatro* misturas. Verifica-se que os modelos treinados a partir dos *templates* de dígitos isolados apresentam um desempenho inferior embora a diferença de desempenho entre os tipos de treinamento seja muito pequena. A Tabela 5.18 mostra os resultados com a base de teste utilizando modelos de *cinco* estados e *quatro* misturas, onde a diferença de desempenho no reconhecimento de palavras e sentenças também é pequena, menor que $0,6\%$. Portanto, ao se avaliar os resultados considerando o intervalo de confiança, verifica-se que os dois modelos apresentam valores de desempenho dentro da mesma faixa de segurança, não sendo possível afirmar qual tipo de treinamento apresenta um melhor desempenho.

Tabela 5.17: Resultados para Diferentes Formas de Treinamento de Dígitos Conectados - TIDIGITS.

TIDIGITS - Base de Treinamento (5 estados e 4 misturas)				
Tipo de Treinamento	Acerto de Palavras		Acerto de Sentenças	
Com *Templates* de Dígitos Isolados	$99,03\%$	$\pm 0,11\%$	$95,92\%$	$\pm 0,42\%$
Sem *Templates* de Dígitos Isolados	$99,06\%$	$\pm 0,11\%$	$96,05\%$	$\pm 0,41\%$

Tabela 5.18: Resultados para Diferentes Formas de Treinamento de Dígitos Conectados - TIDIGITS.

TIDIGITS - Base de Teste (5 estados e 4 misturas)				
Tipo de Treinamento	Acerto de Palavras		Acerto de Sentenças	
Com *Templates* de Dígitos Isolados	$98,71\%$	$\pm 0,13\%$	$94,71\%$	$\pm 0,47\%$
Sem *Templates* de Dígitos Isolados	$98,65\%$	$\pm 0,13\%$	$94,15\%$	$\pm 0,49\%$

5.3.3 Experimentos com o HTK

Experimentos com a base TIDIGITS foram realizados com o software HTK de acordo com os procedimentos relatados no capítulo anterior. Primeiramente os modelos contínuos de

dígitos isolados foram treinados e, a partir deles, seguiu-se o treinamento dos modelos de dígitos conectados. As tabelas abaixo apresentam os resultados para a base de teste utilizando modelos contínuos de palavras de *cinco* estados e *duas* misturas (Tabela 5.19), de *cinco* estados e *quatro* misturas (Tabela 5.20) e de *cinco* estados e *oito* misturas (Tabela 5.21) treinados a partir da base de treinamento. Apesar da taxa de acerto de palavras usando o HTK apresentar valores mais altos que os encontrados com o sistema desenvolvido no Laboratório de Processamento Digital de Sinais de Multimídia em Tempo Real (RT$\underline{\text{M}}$DSP), a diferença entre essas taxas dos dois sistemas diminui à medida que o número de misturas aumenta. Curiosamente, o desempenho do reconhecimento de sentenças obtido com o HTK diminui com o aumento da quantidade de misturas não sendo observado o mesmo para os resultados do sistema desenvolvido. Os resultados do Laboratório para o acerto de sentenças são inferiores ao desempenho atingido pelo HTK, mais de 15% de diferença, apresentando uma pequena melhora apenas com modelos de *cinco* estados e *oito* misturas. Os experimentos realizados na comparação com o HTK consideraram que o tamanho das sentenças era desconhecido, daí o baixo desempenho.

Tabela 5.19: Resultados de Modelos de 5 Estados e 2 Misturas com o HTK e o Sistema Desenvolvido.

TIDIGITS - Base de Teste (5 estados e 2 misturas)				
Sistema	Acerto de Palavras		Acerto de Sentenças	
HTK	$99,16\%$	$\pm 0,11\%$	$69,15\%$	$\pm 0,97\%$
RT$\underline{\text{M}}$DSP	$83,39\%$	$\pm 0,43\%$	$53,36\%$	$\pm 1,05\%$

Tabela 5.20: Resultados de Modelos de 5 Estados e 4 Misturas com o HTK e o Sistema Desenvolvido.

TIDIGITS - Base de Teste (5 estados e 4 misturas)				
Sistema	Acerto de Palavras		Acerto de Sentenças	
HTK	$99,47\%$	$\pm 0,08\%$	$63,57\%$	$\pm 1,01\%$
RT$\underline{\text{M}}$DSP	$86,65\%$	$\pm 0,39\%$	$54,91\%$	$\pm 1,04\%$

Tabela 5.21: Resultados de Modelos de 5 Estados e 8 Misturas com o HTK e o Sistema Desenvolvido.

TIDIGITS - Base de Teste (5 estados e 8 misturas)				
Sistema	Acerto de Palavras		Acerto de Sentenças	
HTK	$99,60\%$	$\pm 0,07\%$	$53,13\%$	$\pm 1,05\%$
RT$^{\underline{M}}$DSP	$89,10\%$	$\pm 0,36\%$	$55,02\%$	$\pm 1,03\%$

5.4 Desempenho em Relação ao Número de Iterações

Nos experimentos a seguir todos os modelos utilizados são modelados com um número menor de épocas ou iterações para se avaliar o comportamento de cada modelo em relação ao melhor limite de parada durante o treinamento dos mesmos. O limite de parada inicialmente utilizado durante o treinamento dos modelos foi o número de épocas menor ou igual a 90. Nos experimentos apresentados, o número de iterações foi fixado em 20, 15 e 10 épocas e avaliou-se o desempenho da taxa de reconhecimento da própria base de treinamento e de teste. Quando a taxa de reconhecimento ainda era muito alta com modelos de 10 iterações, observou-se também o desempenho para 5 iterações de treinamento.

5.4.1 Base LPDF Dígitos

Primeiro Experimento

Neste primeiro experimento com a base LPDF Dígitos avaliou-se modelos de *dez* estados com *duas*, *quatro* e *oito* misturas para ambas as bases de treinamento e teste. Para a maioria dos modelos o maior desempenho da taxa de reconhecimento da base de treinamento foi obtido para modelos com *dez* épocas de treinamento. As tabelas 5.22, 5.23 e 5.24 também apresentam os intervalos de confiança para os modelos, indicando que não há diferença acentuada de desempenho entre eles.

Tabela 5.22: Desempenho da Base de Treinamento LPDF Dígitos para Modelos com 10 Estados e 2 Misturas.

LPDF Dígitos - Base de Treinamento (10 est. e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	$98,80\%$	$\pm 0,41\%$	$89,74\%$	$\pm 3,22\%$
10	$98,97\%$	$\pm 0,38\%$	$90,91\%$	$\pm 3,05\%$
15	$98,80\%$	$\pm 0,41\%$	$90,62\%$	$\pm 3,09\%$
19	$98,83\%$	$\pm 0,40\%$	$90,91\%$	$\pm 3,05\%$

Tabela 5.23: Desempenho da Base de Treinamento LPDF Dígitos para Modelos com 10 Estados e 4 Misturas.

LPDF Dígitos - Base de Treinamento (10 est. e 4 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	$99,09\%$	$\pm 0,36\%$	$93,55\%$	$\pm 2,61\%$
10	$99,47\%$	$\pm 0,27\%$	$94,72\%$	$\pm 2,37\%$
15	$99,35\%$	$\pm 0,30\%$	$94,72\%$	$\pm 2,37\%$
20	$99,35\%$	$\pm 0,30\%$	$94,72\%$	$\pm 2,37\%$
57	$99,35\%$	$\pm 0,30\%$	$94,72\%$	$\pm 2,37\%$

Tabela 5.24: Desempenho da Base de Treinamento LPDF Dígitos para Modelos com 10 Estados e 8 Misturas.

LPDF Dígitos - Base de Treinamento (10 est. e 8 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$99,65\%$	$\pm 0,22\%$	$96,77\%$	$\pm 1,88\%$
15	$99,71\%$	$\pm 0,20\%$	$97,07\%$	$\pm 1,79\%$
20	$99,56\%$	$\pm 0,25\%$	$95,89\%$	$\pm 2,11\%$
90	$99,74\%$	$\pm 0,19\%$	$97,36\%$	$\pm 1,70\%$

No reconhecimento da base de teste, a maioria dos modelos obtiveram melhor desempenho treinados com até *quinze* iterações (tabelas 5.26 e 5.27). O modelo de *dez* estados e *duas* misturas (Tabela 5.25), por exemplo, apresentou melhor desempenho apenas com *cinco* épocas de treinamento. Avaliando-se o intervalo de confiança para os diversos modelos com diferentes números de iterações, observa-se que não há uma diferença significativa de desempenho no reconhecimento da base de teste, por apresentarem valores coincidentes na mesma faixa de segurança.

Tabela 5.25: Desempenho da Base de Teste LPDF Dígitos para Modelos com 10 Estados e 2 Misturas.

LPDF Dígitos - Base de Teste (10 est. e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	$96,16\%$	$\pm 1,34\%$	$67,68\%$	$\pm 9,21\%$
10	$95,96\%$	$\pm 1,37\%$	$65,66\%$	$\pm 9,35\%$
15	$96,06\%$	$\pm 1,35\%$	$65,66\%$	$\pm 9,35\%$
19	$96,06\%$	$\pm 1,35\%$	$65,66\%$	$\pm 9,35\%$

Tabela 5.26: Desempenho da Base de Teste LPDF Dígitos para Modelos com 10 Estados e 4 Misturas.

LPDF Dígitos - Base de Teste (10 est. e 4 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	96,97%	±1,19%	76,77%	±8,32%
10	97,07%	±1,17%	76,77%	±8,32%
15	97,17%	±1,15%	77,78%	±8,19%
20	96,87%	±1,21%	75,76%	±8,44%
57	96,57%	±1,27%	72,73%	±8,77%

Tabela 5.27: Desempenho da Base de Teste LPDF Dígitos para Modelos com 10 Estados e 8 Misturas.

LPDF Dígitos - Base de Teste (10 est. e 8 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	97,07%	±1,17%	75,76%	±8,44%
15	97,37%	±1,11%	79,80%	±7,91%
20	97,17%	±1,15%	76,77%	±8,32%
90	97,68%	±1,05%	79,80%	±7,91%

Segundo Experimento

Para os modelos com número de estados por fonema variáveis e número de misturas fixas para a base de treinamento, metade dos modelos apresentaram melhor desempenho apenas quando atingiram a convergência durante o treinamento (Tabela 5.28 e Tabela 5.30). A outra metade dos modelos (Tabela 5.29 e Tabela 5.31) apresentou melhor desempenho com até *quinze* iterações. Considerando o intervalo de confiança de 95%, observa-se que a diferença de desempenho não é acentuada. As taxas de acerto no reconhecimento de palavras e sentenças da base de treinamento são muito próximas.

Tabela 5.28: Desempenho da Base de Treinamento LPDF Dígitos para Modelos com 1 Estado por Fonema e 2 Misturas.

LPDF Dígitos - Base de Treinamento (1 est./fon. e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$97,65\%$	$\pm 0,57\%$	$82,40\%$	$\pm 4,04\%$
15	$97,92\%$	$\pm 0,54\%$	$82,99\%$	$\pm 3,99\%$
20	$97,83\%$	$\pm 0,55\%$	$82,40\%$	$\pm 4,04\%$
90	$98,06\%$	$\pm 0,52\%$	$86,51\%$	$\pm 3,63\%$

Tabela 5.29: Desempenho da Base de Treinamento LPDF Dígitos para Modelos com 2 Estados por Fonema e 2 Misturas.

LPDF Dígitos - Base de Treinamento (2 est./fon. e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	$98,77\%$	$\pm 0,41\%$	$87,68\%$	$\pm 3,49\%$
10	$98,91\%$	$\pm 0,39\%$	$89,15\%$	$\pm 3,30\%$
15	$98,91\%$	$\pm 0,39\%$	$89,15\%$	$\pm 3,30\%$
20	$98,91\%$	$\pm 0,39\%$	$89,15\%$	$\pm 3,30\%$
38	$98,89\%$	$\pm 0,39\%$	$88,86\%$	$\pm 3,34\%$

Tabela 5.30: Desempenho da Base de Treinamento LPDF Dígitos para Modelos com 3 Estados por Fonema e 2 Misturas.

LPDF Dígitos - Base de Treinamento (3 est./fon. e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$98,86\%$	$\pm 0,40\%$	$88,86\%$	$\pm 3,34\%$
15	$98,97\%$	$\pm 0,38\%$	$89,74\%$	$\pm 3,22\%$
20	$98,97\%$	$\pm 0,38\%$	$89,74\%$	$\pm 3,22\%$
55	$99,00\%$	$\pm 0,37\%$	$90,03\%$	$\pm 3,18\%$

Tabela 5.31: Desempenho da Base de Treinamento LPDF Dígitos para Modelos com 4 Estados por Fonema e 2 Misturas.

LPDF Dígitos - Base de Treinamento (4 est./fon. e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$98,91\%$	$\pm 0,39\%$	$89,15\%$	$\pm 3,30\%$
15	$99,03\%$	$\pm 0,37\%$	$90,32\%$	$\pm 3,14\%$
20	$98,71\%$	$\pm 0,42\%$	$87,10\%$	$\pm 3,56\%$
90	$99,00\%$	$\pm 0,37\%$	$90,03\%$	$\pm 3,18\%$

No reconhecimento de palavras e sentenças da base de teste, os modelos treinados com *um* estado por fonema e *duas* misturas (Tabela 5.32) apresentaram desempenhos melhores para *vinte* iterações de treinamento enquanto os modelos de *dois* estados por fonema e *duas* misturas (Tabela 5.33) apresentaram melhores resultados apenas com *dez* iterações. Já os modelos com *três* estados por fonema e *quatro* estados por fonema (Tabela 5.34 e Tabela 5.35 respectivamente) apresentaram melhor desempenho apenas quando atingiram a convergência no treinamento. Os modelos também apresentam desempenho semelhante considerando o intervalo de confiança de 95%.

Tabela 5.32: Desempenho da Base de Teste LPDF Dígitos para Modelos com 1 Estado por Fonema e 2 Misturas.

LPDF Dígitos - Base de Teste (1 est./fon. e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	95,15%	±1,50%	63,64%	±9,48%
15	95,05%	±1,51%	63,64%	±9,48%
20	95,96%	±1,37%	68,69%	±9,14%
90	95,76%	±1,40%	66,67%	±9,29%

Tabela 5.33: Desempenho da Base de Teste LPDF Dígitos para Modelos com 2 Estados por Fonema e 2 Misturas.

LPDF Dígitos - Base de Teste (2 est./fon. e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	95,66%	±1,42%	66,67%	±9,29%
10	96,46%	±1,29%	68,69%	±9,14%
15	96,46%	±1,29%	68,69%	±9,14%
20	96,46%	±1,29%	68,69%	±9,14%
38	96,46%	±1,29%	68,69%	±9,14%

Tabela 5.34: Desempenho da Base de Teste LPDF Dígitos para Modelos com 3 Estados por Fonema e 2 Misturas.

LPDF Dígitos - Base de Teste (3 est./fon. e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	96,87%	±1,21%	72,73%	±8,77%
15	96,97%	±1,19%	74,75%	±8,56%
20	97,37%	±1,11%	76,77%	±8,32%
55	97,58%	±1,07%	77,78%	±8,19%

Tabela 5.35: Desempenho da Base de Teste LPDF Dígitos para Modelos com 4 Estados por Fonema e 2 Misturas.

LPDF Dígitos - Base de Teste (4 est./fon. e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	98,08%	±0,96%	82,83%	±7,43%
15	98,18%	±0,93%	82,83%	±7,43%
20	98,08%	±0,96%	81,82%	±7,60%
90	98,28%	±0,91%	84,85%	±7,06%

Terceiro Experimento

Modelos com *cinco* estados e *quatro* misturas foram avaliados usando as duas formas de treinamento no reconhecimento das bases de treinamento e teste, ou seja, treinados com ou sem *templates* de dígitos isolados. Para o reconhecimento da base de treinamento (Tabela 5.36 e Tabela 5.37) ambos os modelos (com e sem *templates* de dígitos isolados) obtiveram melhor desempenho quando atingiram a convergência durante o treinamento. Já para o reconhecimento da base de teste (Tabela 5.38 e Tabela 5.39), os modelos treinados com *quinze* iterações apresentaram as melhores taxas.

Tabela 5.36: Resultados da Base de Treinamento LPDF Dígitos para Modelos de 5 Estados e 4 Misturas com Templates de Dígitos Isolados.

LPDF Dígitos - Base de Treinamento (5 est. e 4 mist.)				
Com *Templates* de Dígitos Isolados				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$98,68\%$	$\pm 0,43\%$	$91,50\%$	$\pm 2,96\%$
15	$98,71\%$	$\pm 0,42\%$	$91,20\%$	$\pm 3,01\%$
20	$98,80\%$	$\pm 0,41\%$	$90,62\%$	$\pm 3,09\%$
49	$99,18\%$	$\pm 0,34\%$	$92,67\%$	$\pm 2,77\%$

Tabela 5.37: Resultados da Base de Treinamento LPDF Dígitos para Modelos de 5 Estados e 4 Misturas sem Templates de Dígitos Isolados.

LPDF Dígitos - Base de Treinamento (5 est. e 4 mist.)				
Sem *Templates* de Dígitos Isolados				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$98,50\%$	$\pm 0,46\%$	$88,56\%$	$\pm 3,38\%$
15	$98,30\%$	$\pm 0,49\%$	$86,80\%$	$\pm 3,59\%$
20	$98,33\%$	$\pm 0,48\%$	$87,10\%$	$\pm 3,56\%$
53	$98,62\%$	$\pm 0,44\%$	$90,32\%$	$\pm 3,14\%$

Tabela 5.38: Resultados da Base de Teste LPDF Dígitos para Modelos de 5 Estados e 4 Misturas com Templates de Dígitos Isolados.

LPDF Dígitos - Base de Teste (5 est. e 4 mist.)				
Com *Templates* de Dígitos Isolados				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	95,96%	±1,37%	71,72%	±8,87%
15	96,16%	±1,34%	71,72%	±8,87%
20	95,35%	±1,47%	67,68%	±9,21%
49	95,56%	±1,43%	67,68%	±9,21%

Tabela 5.39: Resultados da Base de Teste LPDF Dígitos para Modelos de 5 Estados e 4 Misturas sem Templates de Dígitos Isolados.

LPDF Dígitos - Base de Teste (5 est. e 4 mist.)				
Sem *Templates* de Dígitos Isolados				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	95,56%	±1,43%	64,65%	±9,42%
15	96,16%	±1,34%	69,70%	±9,05%
20	96,16%	±1,34%	71,72%	±8,87%
53	94,95%	±1,53%	63,64%	±9,48%

Quarto Experimento

No quarto experimento, modelos com *dez* estados e *quatro* misturas foram avaliados para as duas bases (treinamento e teste) considerando o uso de *templates* de dígitos isolados. Usando a base de treinamento (Tabela 5.40), obteve-se um desempenho melhor no reconhecimento de palavras para modelos treinados até atingir a convergência (54 iterações), embora o acerto de sentenças tenha sido melhor para modelos treinados com *vinte* iterações. Avaliando-se o intervalo de confiança, não há grande diferença de desempenho entre esses modelos. Para a base de teste (Tabela 5.41), modelos treinados com *quinze* iterações obtiveram desempenho melhor no acerto de palavras e sentenças.

Tabela 5.40: Resultados da Base de Treinamento LPDF Dígitos com Modelos de 10 Estados e 4 Misturas com Templates de Dígitos Isolados.

LPDF Dígitos - Base de Treinamento (10 est. e 4 mist.)				
Com *Templates* de Dígitos Isolados				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	81,82%	±1,45%	54,25%	±5,29%
15	82,32%	±1,43%	55,13%	±5,28%
20	82,79%	±1,42%	56,89%	±5,25%
54	83,46%	±1,39%	56,01%	±5,27%

Tabela 5.41: Resultados da Base de Teste LPDF Dígitos com Modelos de 10 Estados e 4 Misturas com Templates de Dígitos Isolados.

LPDF Dígitos - Base de Teste (10 est. e 4 mist.)				
Com *Templates* de Dígitos Isolados				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	68,28%	±3,24%	27,27%	±8,77%
15	69,19%	±3,22%	31,31%	±9,14%
20	68,99%	±3,22%	31,31%	±9,14%
54	68,69%	±3,23%	31,31%	±9,14%

5.4.2 Base TIDIGITS

Primeiro Experimento

No primeiro experimento realizado com a base TIDIGITS de treinamento observou-se o desempenho de modelos com *cinco* estados e *duas* misturas, *cinco* estados e *quatro* misturas e *cinco* estados e *oito* misturas. O primeiro modelo de *cinco* estados e *duas* misturas (Tabela 5.42) apresentou um desempenho melhor com apenas *vinte* iterações. A segunda estrutura de modelo, com *cinco* estados e *quatro* misturas (Tabela 5.43), apresentou um desempenho melhor apenas quando atingiu a convergência, mas nada muito superior em relação aos mesmos modelos treinados com um número menor de épocas. Finalmente, os modelos com *cinco* estados e *oito* misturas (Tabela 5.44) apresentaram desempenho superior no reconhecimento de palavras com apenas *cinco* iterações de treinamento e no reconhecimento de sentenças

com apenas *quinze* iterações, mas ao se considerar o intervalo de confiança, pode-se observar que os modelos treinados com somente *cinco* épocas apresentaram resultados satisfatórios para o reconhecimento de palavras e sentenças.

Tabela 5.42: Resultados da Base de Treinamento TIDIGITS para Modelos com 5 Estados e 2 Misturas.

TIDIGITS - Base de Treinamento (5 est. e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$98,50\%$	$\pm 0,14\%$	$93,85\%$	$\pm 0,51\%$
15	$98,57\%$	$\pm 0,14\%$	$94,11\%$	$\pm 0,50\%$
20	$98,59\%$	$\pm 0,14\%$	$94,18\%$	$\pm 0,49\%$
25	$98,59\%$	$\pm 0,14\%$	$94,16\%$	$\pm 0,50\%$

Tabela 5.43: Resultados da Base de Treinamento TIDIGITS para Modelos com 5 Estados e 4 Misturas.

TIDIGITS - Base de Treinamento (5 est. e 4 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	$98,99\%$	$\pm 0,12\%$	$95,54\%$	$\pm 0,44\%$
10	$99,02\%$	$\pm 0,11\%$	$95,87\%$	$\pm 0,42\%$
15	$99,03\%$	$\pm 0,11\%$	$95,91\%$	$\pm 0,42\%$
20	$99,04\%$	$\pm 0,11\%$	$95,92\%$	$\pm 0,42\%$
55	$99,06\%$	$\pm 0,11\%$	$96,05\%$	$\pm 0,41\%$

Tabela 5.44: Resultados da Base de Treinamento TIDIGITS para Modelos com 5 Estados e 8 Misturas.

TIDIGITS - Base de Treinamento (5 est. e 8 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	$99,35\%$	$\pm 0,09\%$	$97,10\%$	$\pm 0,35\%$
10	$99,34\%$	$\pm 0,09\%$	$97,09\%$	$\pm 0,35\%$
15	$99,35\%$	$\pm 0,09\%$	$97,14\%$	$\pm 0,35\%$
20	$99,34\%$	$\pm 0,09\%$	$97,11\%$	$\pm 0,35\%$
90	$99,33\%$	$\pm 0,09\%$	$97,12\%$	$\pm 0,35\%$

Utilizando a base de teste da TIDIGITS, os modelos com *cinco* estados e *duas* misturas (Tabela 5.45) já apresentaram um desempenho satisfatório com *vinte* iterações enquanto os modelos com *cinco* estados e *quatro* misturas (Tabela 5.46) precisaram apenas de *dez* iterações para atingirem seus melhores resultados. Os modelos com *cinco* estados e *oito* misturas treinados por *quinze* épocas apresentaram os melhores desempenhos no reconhecimento de palavras e sentenças (Tabela 5.47).

Tabela 5.45: Resultados da Base de Teste TIDIGITS para Modelos com 5 Estados e 2 Misturas.

TIDIGITS - Base de Teste (5 est. e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$97,91\%$	$\pm 0,17\%$	$91,17\%$	$\pm 0,60\%$
15	$97,99\%$	$\pm 0,16\%$	$91,51\%$	$\pm 0,59\%$
20	$98,04\%$	$\pm 0,16\%$	$91,60\%$	$\pm 0,58\%$
25	$98,03\%$	$\pm 0,16\%$	$91,56\%$	$\pm 0,58\%$

Tabela 5.46: Resultados da Base de Teste TIDIGITS para Modelos com 5 Estados e 4 Misturas.

TIDIGITS - Base de Teste (5 est. e 4 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	$98,55\%$	$\pm 0,14\%$	$93,53\%$	$\pm 0,52\%$
10	$98,65\%$	$\pm 0,13\%$	$94,15\%$	$\pm 0,49\%$
15	$98,62\%$	$\pm 0,14\%$	$93,98\%$	$\pm 0,50\%$
20	$98,63\%$	$\pm 0,13\%$	$94,07\%$	$\pm 0,50\%$
55	$98,62\%$	$\pm 0,14\%$	$94,03\%$	$\pm 0,50\%$

Tabela 5.47: Resultados da Base de Teste TIDIGITS para Modelos com 5 Estados e 8 Misturas.

TIDIGITS - Base de Teste (5 est. e 8 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$99,00\%$	$\pm 0,12\%$	$95,39\%$	$\pm 0,44\%$
15	$99,00\%$	$\pm 0,12\%$	$95,54\%$	$\pm 0,43\%$
20	$98,97\%$	$\pm 0,12\%$	$95,45\%$	$\pm 0,44\%$
90	$98,87\%$	$\pm 0,12\%$	$95,11\%$	$\pm 0,45\%$

Segundo Experimento

No segundo experimento com a base TIDIGITS, estruturas de modelos com quantidade de estados por fonema variável e *duas* misturas foram observadas para diversos números de iterações. Os modelos com *um* estado por fonema e *duas* misturas (Tabela 5.48) tiveram desempenho superior ao atingirem a convergência durante o treinamento (26 iterações). Modelos com *dois* estados por fonema e *duas* misturas (Tabela 5.49) também apresentaram o mesmo comportamento, ou seja, os modelos treinados com 38 épocas obtiveram melhores resultados no reconhecimento de palavras e sentenças. Os modelos com *três* estados por fonema e *duas* misturas atingiram melhores desempenhos com apenas *dez* iterações (Tabela 5.50) e para modelos com *quatro* estados por fonema e *duas* misturas foram necessárias *quinze* iterações (Tabela 5.51).Considerando-se o intervalo de confiança de 95%, a diferença de resultados para os diversos números de iterações para um mesmo modelo não é significativa.

Tabela 5.48: Resultados da Base de Treinamento TIDIGITS para Modelos com 1 Estado por Fonema e 2 Misturas.

TIDIGITS - Base de Treinamento (1 est./fon e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$97,84\%$	$\pm 0,17\%$	$91,41\%$	$\pm 0,59\%$
15	$97,88\%$	$\pm 0,17\%$	$91,63\%$	$\pm 0,58\%$
20	$97,88\%$	$\pm 0,17\%$	$91,64\%$	$\pm 0,58\%$
26	$97,90\%$	$\pm 0,17\%$	$91,71\%$	$\pm 0,58\%$

Tabela 5.49: Resultados da Base de Treinamento TIDIGITS para Modelos com 2 Estados por Fonema e 2 Misturas.

TIDIGITS - Base de Treinamento (2 est./fon e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$98,79\%$	$\pm 0,13\%$	$94,75\%$	$\pm 0,47\%$
15	$98,60\%$	$\pm 0,14\%$	$94,31\%$	$\pm 0,49\%$
20	$98,79\%$	$\pm 0,13\%$	$94,75\%$	$\pm 0,47\%$
38	$98,94\%$	$\pm 0,12\%$	$95,20\%$	$\pm 0,45\%$

Tabela 5.50: Resultados da Base de Treinamento TIDIGITS para Modelos com 3 Estados por Fonema e 2 Misturas.

TIDIGITS - Base de Treinamento (3 est./fon e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	98,97%	±0,12%	95,47%	±0,44%
10	99,10%	±0,11%	96,03%	±0,41%
15	99,09%	±0,11%	95,99%	±0,41%
20	99,09%	±0,11%	96,00%	±0,41%
74	99,10%	±0,11%	96,02%	±0,41%

Tabela 5.51: Resultados da Base de Treinamento TIDIGITS para Modelos com 4 Estados por Fonema e 2 Misturas.

TIDIGITS - Base de Treinamento (4 est./fon e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	99,15%	±0,11%	96,20%	±0,40%
10	99,17%	±0,11%	96,29%	±0,40%
15	99,18%	±0,11%	96,32%	±0,40%
20	99,15%	±0,11%	96,24%	±0,40%
49	99,16%	±0,11%	96,32%	±0,40%

Considerando a base de teste TIDIGITS, com exceção dos modelos com *dois* estados por fonema e *duas* misturas (Tabela 5.53), que só alcançaram um melhor desempenho ao atingirem a convergência durante o treinamento com 38 iterações, todos os outros modelos avaliados apresentaram resultados melhores com menos iterações. Os modelos com *um* estado por fonema e *duas* misturas (Tabela 5.52) apresentaram taxas melhores de desempenho com apenas *quinze* iterações e os outros dois modelos, com *três* estados por fonema e *duas* misturas (Tabela 5.54) e com *quatro* estados por fonema e *duas* misturas (Tabela 5.55), com apenas *dez* iterações.

Tabela 5.52: Resultados da Base de Teste TIDIGITS para Modelos com 1 Estado por Fonema e 2 Misturas.

TIDIGITS - Base de Teste (1 est./fon e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$97,11\%$	$\pm 0,19\%$	$88,49\%$	$\pm 0,67\%$
15	$97,24\%$	$\pm 0,19\%$	$88,79\%$	$\pm 0,66\%$
20	$97,24\%$	$\pm 0,19\%$	$88,75\%$	$\pm 0,66\%$
26	$97,24\%$	$\pm 0,19\%$	$88,71\%$	$\pm 0,67\%$

Tabela 5.53: Resultados da Base de Teste TIDIGITS para Modelos com 2 Estados por Fonema e 2 Misturas.

TIDIGITS - Base de Teste (2 est./fon e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$98,36\%$	$\pm 0,15\%$	$92,84\%$	$\pm 0,54\%$
15	$98,08\%$	$\pm 0,16\%$	$92,03\%$	$\pm 0,57\%$
20	$98,36\%$	$\pm 0,15\%$	$92,84\%$	$\pm 0,54\%$
38	$98,50\%$	$\pm 0,14\%$	$93,44\%$	$\pm 0,52\%$

Tabela 5.54: Resultados da Base de Teste TIDIGITS para Modelos com 3 Estados por Fonema e 2 Misturas.

TIDIGITS - Base de Teste (3 est./fon e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	98,54%	±0,14%	93,53%	±0,52%
10	98,68%	±0,13%	94,25%	±0,49%
15	98,66%	±0,13%	94,18%	±0,49%
20	98,65%	±0,13%	94,17%	±0,49%
74	98,66%	±0,13%	94,15%	±0,49%

Tabela 5.55: Resultados da Base de Teste TIDIGITS para Modelos com 4 Estados por Fonema e 2 Misturas.

TIDIGITS - Base de Teste (4 est./fon e 2 mist.)				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
5	98,77%	±0,13%	94,52%	±0,48%
10	98,78%	±0,13%	94,54%	±0,48%
15	98,74%	±0,13%	94,41%	±0,48%
20	98,75%	±0,13%	94,38%	±0,49%
49	98,76%	±0,13%	94,45%	±0,48%

Terceiro Experimento

No terceiro e último experimento com a base TIDIGITS modelos de *cinco* estados e *quatro* misturas treinados a partir de *templates* de dígitos isolados obtiveram desempenho superior no acerto de palavras e sentenças, avaliando-se ambas as bases de treinamento (Tabela 5.56) e teste (Tabela 5.57), apenas ao atingirem o limite de 90 iterações. Considerando o intervalo de confiança de 95% para o desempenho no acerto de palavras e sentenças, os modelos com 90 iterações apresentaram resultados superiores em relação aos demais modelos com número menor de iterações e sob a mesma condição de treinamento.

Tabela 5.56: Resultados da Base de Treinamento TIDIGITS com Modelos de 5 Estados e 4 Misturas com Templates de Dígitos Isolados.

TIDIGITS - Base de Treinamento (5 est. e 4 mist.)				
Com *Templates* de Dígitos Isolados				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$98,30\%$	$\pm 0,15\%$	$94,24\%$	$\pm 0,49\%$
15	$98,50\%$	$\pm 0,14\%$	$94,65\%$	$\pm 0,48\%$
20	$98,56\%$	$\pm 0,14\%$	$94,72\%$	$\pm 0,47\%$
90	$99,03\%$	$\pm 0,11\%$	$95,92\%$	$\pm 0,42\%$

Tabela 5.57: Resultados da Base de Teste TIDIGITS com Modelos de 5 Estados e 4 Misturas com Templates de Dígitos Isolados.

TIDIGITS - Base de Teste (5 est. e 4 mist.)				
Com *Templates* de Dígitos Isolados				
Número de Iterações	Acerto de Palavras		Acerto de Sentenças	
10	$98,00\%$	$\pm 0,16\%$	$92,57\%$	$\pm 0,55\%$
15	$98,19\%$	$\pm 0,15\%$	$93,20\%$	$\pm 0,53\%$
20	$98,23\%$	$\pm 0,15\%$	$93,16\%$	$\pm 0,53\%$
90	$98,71\%$	$\pm 0,13\%$	$94,71\%$	$\pm 0,47\%$

Capítulo 6

Conclusão

Este trabalho apresentou o desenvolvimento de um sistema de reconhecimento de dígitos conectados utilizando HMMs contínuos baseados em palavras e independente de locutor. Os experimentos realizados tiveram por meta observar o comportamento do modelo contínuo implementado no sistema do laboratório RT$^{\underline{M}}$DSP em relação ao modelo discreto já existente no sistema, bem como em relação ao modelo contínuo de um *software* livre de reconhecimento de fala, chamado HTK. Para tal utilizou-se duas bases de dados, uma em português brasileiro e outra do inglês americano.

Os testes com a base LPDF Dígitos mostraram que tanto para o *corpus* de treinamento (31 locutores) quanto para o de teste (9 locutores), o desempenho no reconhecimento de dígitos conectados utilizando HMMs discretos foi um pouco melhor quando comparado aos HMMs contínuos. Ao se avaliar o intervalo de confiança observou-se que ambos os modelos tinham valores de desempenho na mesma faixa. Essas características não foram observadas com a base TIDIGITS, cuja base de treinamento foi gravada por 112 locutores e a base de teste por 113 locutores. Mesmo considerando o intervalo de confiança de 95%, o sistema discreto apresentou um desempenho melhor.

A complexidade computacional no treinamento de modelos contínuos é muito maior comparada ao dos modelos discretos, visto que HMMs contínuos têm muito mais parâmetros a serem estimados em relação aos HMMs discretos. Portanto, o tamanho da base de treinamento é um fator relevante na taxa de reconhecimento de fala quando se trata de treinamento de modelos contínuos. O fato de a base em português apresentar uma baixa taxa de reconhecimento de sentenças, menos de 85% para modelos com *quatro* estados por fonema e

duas misturas, provavelmente deve-se ao fato da base de treinamento ser muito pequena, contendo apenas 341 elocuções.

Como o tamanho da base TIDIGITS de treinamento é maior, esperava-se um desempenho melhor em relação aos modelos discretos e também devido à literatura ressaltar a superioridade dos modelos contínuos em relação aos discretos. No reconhecimento de dígitos conectados independente de locutor, apesar de a diferença entre os dois modelos não ter sido elevada, a complexidade computacional dos HMMs contínuos usando modelos de *cinco* estados e *oito* misturas foi grande. Algumas suposições podem ser feitas a respeito deste comportamento:

- Como muitos *codebooks* são utilizados nos modelos discretos (*seis codebooks* independentes, um para cada parâmetro, com *codebooks* de 256 vetores para os parâmetros mel cepstrais e de 32 vetores para as energias) [ANDR 01], os erros de quantização não foram relevantes, mas se usarmos apenas *um codebook* o desempenho dos modelos contínuos poderia ser comparável ao dos modelos discretos.

- A quantidade de misturas para um reconhecimento de fala independente de locutor de uma base extensa como a TIDIGITS (8700 sentenças) pode ser pequena, mas aumentando-se cada vez mais o número de misturas iria elevar a complexidade computacional consideravelmente não compensando a pequena melhoria que se obteria.

Este tipo de comportamento, onde os modelos discretos superaram os modelos contínuos, também foi observado em um experimento realizado com o sistema de reconhecimento de fala contínua independente de locutor para grandes vocabulários, SPHINX, desenvolvido na *Carnegie Mellon University* [HUAN 89].

Observando as Tabelas 5.3 - 5.6 para a base LPDF Dígitos e as Tabelas 5.13 - 5.16 para a base TIDIGITS, nota-se que ao se fixar o número de estados e ao se variar o número de misturas a taxa de reconhecimento de palavras e sentenças é, em geral, melhor do que a obtida com modelos de misturas fixas e estados variáveis. Embora os modelos com estados fixos apresentem um desempenho melhor, ao dobrarmos o número de misturas, a taxa de acerto não segue na mesma direção para bases pequenas como a LPDF Dígitos, ou seja, o aumento é pequeno: $7,07\%$ para reconhecimento de sentenças da base de teste quando uma grande complexidade computacional está envolvida. Apesar da taxa de reconhecimento

ser, em geral, menor para modelos com misturas fixas e estados variáveis, ao dobrarmos o número de estados por fonema o ganho é maior: $16,16\%$ para reconhecimento de sentenças da base de teste. Avaliando-se o intervalo de confiança para modelos com estados fixos e misturas variáveis e vice-versa, observa-se que os desempenhos obtidos são muito próximos e há a probabilidade de coincidirem dentro desse intervalo. Já com a base TIDIGITS os modelos com estados fixos e misturas variáveis apresentam uma taxa de acerto maior quando comparamos com os modelos de misturas fixas, mas o ganho é semelhante quando dobramos o número de misturas e mantemos os estados fixos ou dobramos o número de estados com as misturas fixas.

Pode-se inferir dos resultados que é mais interessante aumentarmos o número de estados do que o número de misturas quando dispomos de bases pequenas, por apresentarem pouca diversidade de informações acústicas necessárias a uma boa modelagem dos parâmetros dos modelos contínuos.

Foram apresentados experimentos com resultados obtidos de *dois* tipos de treinamento dos dígitos conectados: um que usava as informações dos *templates* dos dígitos isolados e outro que não usava. Observou-se que, para bases pequenas, o desempenho entre os dois tipos de treinamento é bem diferente quando dobramos o número de estados do modelo. A taxa de acerto cai consideravelmente quando utilizamos os modelos de dígitos conectados que usaram as informações dos *templates* dos dígitos isolados durante o treinamento. Já o desempenho dos modelos de dígitos conectados que não usaram essas informações apresentou melhora no reconhecimento. Para bases grandes como a TIDIGITS não houve grande diferença quando comparados os modelos com treinamento diferentes. Embora o uso das informações dos *templates* dos dígitos isolados durante o treinamento dos modelos discretos apresentados em [ANDR 01] tenha sido relevante no desempenho dos mesmos, nos modelos contínuos não se observou o mesmo, principalmente com a base em português brasileiro. Os HMMs contínuos apresentaram desempenho superior apenas ao dobrar-se o número de estados.

O comportamento diferente de desempenho ocorrido com a base LPDF Dígitos e a TIDIGITS deveu-se provavelmente à forma de gravação da base de treinamento de dígitos isolados e ao fato dessa base fazer ou não parte da base de dígitos conectados. Na base em português brasileiro, os dígitos foram gravados separadamente, ou seja, não foram gravados sequencialmente e depois segmentados manualmente como ocorreu em [ANDR 01]. Portan-

to, informações acústicas importantes foram perdidas, além do fato dessa base de dígitos isolados não fazer parte da base de dígitos conectados que contém apenas seqüências de *oito* dígitos. Já a base de dígitos isolados da TIDIGITS fez parte do corpus de dígitos conectados da mesma que possuíam sentenças de *um* até *sete* dígitos, apesar de apresentar a mesma forma de gravação da LPDF Dígitos.

Os testes com o *software* HTK serviu de comparação de desempenho entre este e o sistema desenvolvido no laboratório RT$\underline{\text{M}}$DSP utilizando modelos contínuos. Os resultados mostraram que as taxas obtidas no reconhecimento de palavras são semelhantes, embora no HTK seja superior, e, portanto, consistentes com os resultados de um sistema utilizado em muitas pesquisas no mundo inteiro. Já para o desempenho no reconhecimento de sentenças, o HTK apresentou taxas muito baixas, mas ainda melhores que as obtidas com o sistema de modelos contínuos desenvolvido no laboratório RT$\underline{\text{M}}$DSP.

Outro experimento utilizando os mesmos modelos apresentados anteriormente com as bases LPDF Dígitos e TIDIGITS consistiu em treinar cada modelo com um número menor de épocas afim de se verificar o melhor desempenho dos mesmos no reconhecimento de palavras e sentenças das bases de treinamento e teste. No reconhecimento do corpus de treinamento de ambas as bases (LPDF Dígitos e TIDIGITS) observou-se que no geral, dentre todos os modelos avaliados, os que tiveram em sua maioria os melhores desempenhos foram aqueles treinados até atingirem a convergência ou o limite imposto de 90 iterações. Já para o reconhecimento do corpus de teste os melhores modelos foram aqueles treinados com um número menor de iterações. Para a base em português a maioria dos modelos que apresentaram melhores taxas foram aqueles treinados com *quinze* iterações e para a base em inglês foram aqueles treinados com *dez* épocas. Estes resultados confirmam a influência negativa do "super-treinamento" no desempenho dos modelos para o reconhecimento de palavras e sentenças independente de locutor, ou seja, modelos treinados com até *quinze* iterações são suficientes para a obtenção de bons resultados.

Como trabalhos futuros, sugere-se a repetição dos mesmos experimentos usando modelos semi-contínuos que usam características dos modelos discretos e contínuos. Outra sugestão é inserir modelos de silêncio entre os dígitos conectados, visto que várias sentenças, principalmente da base TIDIGITS, apresentam pequenas pausas entre os dígitos.

Muitas dificuldades foram encontradas durante a programação, principalmente na fase

de treinamento dos dígitos conectados, devido a grande quantidade de cálculos tornando o processo muito lento e fazendo-se necessária uma preocupação constante com a otimização do algoritmo visando uma aplicação futura em tempo real. A implementação do sistema em uma placa de desenvolvimento de processamento digital de sinais que contenha interfaces de telefonia e Ethernet seria um trabalho interessante e com muita empregabilidade prática.

Uma outra possibilidade é o estudo e emprego de gramática em diálogos por meio de máquinas de estados finitos. Um sistema de diálogos interativos pela fala empregando mensagens gravadas como saída, possibilita o diálogo com o usuário.

Fazer com que a máquina desempenhe tão bem algumas tarefas simples que qualquer ser humano pode fazer não é um sonho recente. No mundo dos filmes de ficção científica essa possibilidade já foi prevista. Alguns exemplos clássicos como "2001: Uma Odisséia no Espaço" e "Guerra nas Estrelas", onde computadores e robôs interagiam perfeitamente com o homem, identificando sons, falas e sentimentos é um exemplo. O sistema de reconhecimento de fala ideal ainda está muito longe de ser atingido. A cada dia surgem novas pesquisas e descobertas nesta área que é tão vasta e desafiadora. Espera-se que o presente trabalho possa ser uma boa fonte de informação e consulta, e que também contribua para o estudo e pesquisas nesta área tão estimulante.

Referências Bibliográficas

[ANDR 01] R. V. Andreão, "Implementação em Tempo Real de um Sistema de Reconhecimento de Dígitos Conectados", Dissertação de Mestrado, UNICAMP, Campinas, Janeiro 2001.

[EULE 96] S. Euler, "A Time Continuous Model for Speech Recognition", ICASSP-96, págs. II-889-892, Atlanta 1996.

[GERS 92] A. Gersho e R. M. Gray, "Vector Quantization and Signal Compression", Kluwer Academic Publishers, 1992.

[HIRS 00] H. Hirsch e D. Pearce, "The Aurora Experimental Framework for the Performance Evaluation of Speech Recognition Systems under Noisy Conditions", ISCA ITRW ASR2000 Automatic Speech Recognition: Challenges for the Next Millennium; Paris, France, Setembro 18-20, 2000.

[HTK 93] http://htk.eng.cam.ac.uk/

[HUAN 88] X. D. Huang e M. A. Jack, "Semi-Continuous Hidden Markov Models in Isolated Word Recognition", 9th International Conference on Pattern Recognition, Vol.1, págs. 406-408, Novembro 1988.

[HUAN 89] X. D. Huang, "Semi-Continuous Hidden Markov Models for Speech Recognition", Doctor Thesis, University of Edinburgh, 1989.

[HUAN 01] X. D. Huang, A. Acero e H. Hon, "Spoken Language Processing - A Guide to Theory, Algorithm and System Development", Prentice Hall, 2001.

[JELI 76] F. Jelinek, "Continuous Speech Recognition by Statistical Methods", Proc IEEE, 64:532-536, Abril 1976.

[JELI 97] F. Jelinek, "Statistical Methods for Speech Recognition", MIT Press, 1997.

[LEEC 89] K. Lee, "Automatic Speech Recognition ", Kluwer Academic Publishers, 1989.

[LEON 84] R. G. Leonard, "A Database for Speaker Independent Digit Recognition", ICASSP84, Vol. 3, págs. 42.11.1-4, Março, 1984.

[MART 97] J. A. Martins, "Avaliação de Diferentes Técnicas para Reconhecimento de Fala", Tese de Doutorado, UNICAMP, Campinas, Dezembro 1997.

[MELO 00] L. G. P. Meloni, "Learning Discrete Hidden Markov Models", Computer Applications in Engineering Education, John Wiley & Sons, Vol. 8, issue 3, págs. 1-9, Outubro, 2000.

[PAUL 89] D. Paul, "The Lincoln Robust Continuous Speech Recognizer", ICASSP-89, págs. 449-452, 1989.

[PERL 05] http://www.htmlstaff.org/Perl/perl20.php

[RABI 86] L. R. Rabiner e B. H. Juang, "An Introduction to Hidden Markov Models", IEEE Transactions on Acoustics, Speech and Signal Processing, Janeiro 1986.

[RABI 89a] 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.

[RABI 89b] L. R. Rabiner, J. G. Wilpon, "High Performance Connected Digit Recognition Using Hidden Markov Models", IEEE Transactions on Acoustics Speech and Signal Processing, Vol. 47, no. 8, Agosto, 1989.

[RABI 93] L. Rabiner e B. H. Juang, "Fundamentals of Speech Recognition", Prentice Hall, 1993.

[SAMU 03] Samudravijaya K., M. Barot, "A Comparison of Public Domain Software Tools for Speech Recognition", Workshop on Spoken Language Processing, TIFR, págs. 9-11, Mumbai, Janeiro, 2003.

[YOUN 01] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev e P. Woodland, "The HTK Book - version 3.1", 2001.

[YNOG 99] C. Ynoguti, "Reconhecimento de Fala Contínua Usando Modelos Ocultos de Markov", Tese de Doutorado, UNICAMP, Campinas, Maio 1999.

Apêndice A

Treinamento dos Modelos Usando o HTK - Continuação do Passo 5

Após a iniciação dos parâmetros dos modelos de dígitos isolados através das ferramentas **HCompV**, **HInit** e **HRest** executase a linha de comando a seguir:

HERest −C config2.txt −I words0.mlf −t 250.0 150.0 1000.0 −S dir_tr_mfc.txt −H hmm0/macros.txt −H hmm0/hmmdefs.txt −M hmm1 words0.txt

A ferramenta **HERest** retorna dentro da pasta hmm1 os arquivos *macros.txt* e *hmmdefs.txt*.

Deve-se então executar **HERest** mais uma vez trocando-se o nome da pasta de entrada hmm0 por hmm1 e a de saída hmm1 por hmm2 como mostra a linha de comando seguinte:

HERest −C config2.txt −I words0.mlf −t 250.0 150.0 1000.0 −S dir_tr_mfc.txt −H hmm1/macros.txt −H hmm1/hmmdefs.txt −M hmm2 words0.txt

Novamente a ferramenta **HERest** retorna os arquivos *macros.txt* e *hmmdefs.txt* agora dentro da pasta hmm2. De forma semelhante, troca-se o nome da pasta de entrada hmm1 por hmm2 e a de saída hmm2 por hmm3. A linha de comando a seguir é executada:

HERest −C config2.txt −I words0.mlf −t 250.0 150.0 1000.0 −S dir_tr_mfc.txt −H hmm2/macros.txt −H hmm2/hmmdefs.txt −M hmm3 words0.txt

Da mesma forma os arquivos *macros.txt* e *hmmdefs.txt* são retornados dentro da pasta hmm3. Troca-se então o nome da pasta de entrada hmm2 por hmm3 e a de saída hmm3 por hmm4 e a ferramenta **HERest** é executada novamente através da linha de comando:

HERest −C config2.txt −I words0.mlf −t 250.0 150.0 1000.0 −S dir_tr_mfc.txt −H hmm3/macros.txt −H hmm3/hmmdefs.txt −M hmm4 words0.txt

O procedimento é repetido mais *três* vezes tomando-se o cuidado de trocar o nome das pastas de entrada e saída em cada execução da linha de comando como mostra as linhas de comando a seguir:

HERest −C config2.txt −I words0.mlf −t 250.0 150.0 1000.0 −S dir_tr_mfc.txt −H hmm4/macros.txt −H hmm4/hmmdefs.txt −M hmm5 words0.txt

HERest −C config2.txt −I words0.mlf −t 250.0 150.0 1000.0 −S dir_tr_mfc.txt −H hmm5/macros.txt −H hmm5/hmmdefs.txt −M hmm6 words0.txt

HERest −C config2.txt −I words0.mlf −t 250.0 150.0 1000.0 −S dir_tr_mfc.txt −H hmm6/macros.txt −H hmm6/hmmdefs.txt −M hmm7 words0.txt

Após a sétima execução da ferramenta **HERest** quando os arquivos *macros.txt* e *hmmdefs.txt* são treinados e retornados dentro da pasta hmm7, executa-se a ferramenta *HVite* que usa o algoritmo de Viterbi. Cria-se então o arquivo *dict*1.*txt* que é igual ao arquivo *dict* sem as pausas curtas *sp* e com a especificação *silence sil*.

O arquivo *dict*1.*txt* é feito manualmente e é usado por HVite. A Figura 1 mostra como deve ser esse arquivo.

A ferramenta **HVite** é então executada onde Linha de comando:

EIGHT	EIGHT
FIVE	FIVE
FOUR	FOUR
NINE	NINE
OH	OH
ONE	ONE
SENT-END	sil
SENT-START	sil
SEVEN	SEVEN
silence	sil
SIX	SIX
THREE	THREE
TWO	TWO
ZERO	ZERO

Figura 1: Arquivo *dict*1.*txt*.

HVite −l ∗ − o SWT −b silence −C config2.txt −a −H hmm7/macros.txt −H hmm7/hmmdefs.txt −i alinha.mlf −m −t 250.0 −y lab −I train_isolado.mlf −S dir_tr_mfc.txt dict1.txt words0.txt

alinha.mlf (gerado pelo HTK - contém a transcrição das palavras do arquivo *train_isolado.mlf* acrescido do silêncio no início e fim de cada palavra. Como não há pausas entre as palavras o arquivo *alinha.mlf* é idêntico ao *words*0.*mlf*.).

Nas próximas linhas de comando deve-se acrescentar a opção **−B** antes da opção **−C**, para que os modelos sejam armazenados na forma binária.

Linha de comando:

HERest −B −C config2.txt −I alinha.mlf −t 250.0 150.0 1000.0 −S dir_tr_mfc.txt −H hmm7/macros.txt −H hmm7/hmmdefs.txt −M hmm8 words0.txt

Retorna dentro de hmm8 os arquivos *macros.txt* e *hmmdefs.txt*.

Linha de comando:

HERest −B −C config2.txt −I alinha.mlf −t 250.0 150.0 1000.0 −S dir_tr_mfc.txt −H hmm8/macros.txt −H hmm8/hmmdefs.txt −M hmm9 words0.txt

Retorna dentro de hmm9 os arquivos *macros.txt* e *hmmdefs.txt*.

O processo é repetido mais *seis* vezes até que o *HERest* retorne dentro da pasta hmm15 os arquivos $macros.txt$ e $hmmdefs.txt$. Os arquivos $macros.txt$ e $hmmdefs.txt$ são os modelos treinados com a base de treinamento de dígitos isolados que serão usados no treinamento dos dígitos conectados.

Apêndice B

Primeiros Experimentos com Dados Articulatórios e sua Relação com a Segmentação Acústica

Artigo apresentado no VII Congresso Nacional - I Congresso Internacional de Fonética e Fonologia da Faculdade de Letras da UFMG, Belo Horizonte, Brasil; e publicado na Revista de Estudos da Linguagem - Faculdade de Letras da UFMG, Vol 12 - N 1, 39 - 52, 2004.

Primeiros Experimentos com Dados Articulatórios e sua Relação com a Segmentação Acústica

ABSTRACT: The main purpose of this work is to analyze the projections of maximum and minimum points of two articulatory trajectories during speech production over the acoustic signal namely, mouth aperture and the jaw opening. In order to do so, we have chosen a native French speaker who repeated 5 times the following sentence: "*C´est pas ububuz, c'est pas ybybyz, c'est pas iziziz, c'est pas azhazhaz*". It seems reasonable from the analysis shown to establish a correspondence between articulatory and acoustic events, which happens to be useful to help determine acoustic boundaries. The articulatory boundaries coincide with the middle of the vowels and the mouth aperture seems to be a more consistent parameter than jaw opening for establishing these regularities.

1. INTRODUÇÃO

Caracterizar unidades fonéticas em função de parâmetros físicos mensuráveis tem sido objeto de estudos de muitos trabalhos em fonética lingüística (Fant 1973; Halle & Stevens 1979; Jakobson, Fant & Halle 1969; Ladefoged 1971). De fato, estruturas fonéticas são caracterizadas por padrões articulatórios que estão em movimento contínuo, ao invés de configurações estáticas (Browman & Goldstein 1986; Browman & Goldstein 1988).

A utilização de informações articulatórias, conjuntamente com informações acústicas no processo de identificação de fronteiras fonéticas, pode ser de grande importância na segmentação de fones. Além disso, em aplicações que necessitem de segmentação automática de sentenças faladas como, por exemplo, em reconhecimento de fala, é esperado que o sistema se torne mais robusto com a utilização da informação articulatória (Benoît 1996) na recuperação da informação fonética, em situações nas quais o sinal acústico não fornece uma definição precisa para a determinação das fronteiras entre os segmentos.

O artigo apresenta o estudo da existência de possíveis correspondências entre eventos articulatórios, tais como picos (pontos de máximo e mínimo) nos sinais de movimentação do queixo ou de abertura da boca, e a estrutura fonética-acústica de uma sentença do francês nativo. A partir dos resultados é avaliado se as informações articulatórias podem facilitar a determinação das fronteiras entre as unidades fonéticas.

Primeiramente avaliamos aqui se existe correspondência temporal entre fones de mesma natureza ou de natureza distinta. Na seqüência, analisa-se a ocorrência dos eventos articulatórios no intuito de verificar se tal informação pode ser utilizada para marcar as fronteiras acústicas.

2. BASE DE DADOS

Nos experimentos utiliza-se uma base de dados coletada no *Advanced Telecommunications Institute International* (ATR), em Kyoto, Japão (Yehia, Rubin & Bateson 1998). Tal base é constituída de dados acústicos e articulatórios gravados para um locutor francês nativo (CB), do sexo masculino, contendo cinco repetições da sentença "*C´est pas ububuz, c'est pas ybybyz, c'est pas iziziz, c'est pas azhazhaz*", pronunciada em uma única sucessão, sendo que cada uma tem 8 segundos de duração.

As trajetórias de pontos emitidos por diodos emissores (LEDs) de luz infravermelha (*Ireds*) posicionados na face (lábio superior, lábio inferior e queixo) do locutor foram rastreadas a partir do Optotrack. Tal equipamento consiste de um conjunto de três câmeras capazes de rastrear a posição dos LEDs, e possibilita a aquisição de posição com uma taxa de amostragem de 60 Hz.

No experimento também foi coletada a atividade de músculos da face através de eletromiografia, e por esse motivo o sinal acústico também foi coletado a 5000 Hz.

2.1 COMPENSAÇÃO DO MOVIMENTO DA CABEÇA

O rastreamento das trajetórias de pontos localizados na face durante a produção da fala, realizada através do Optotrack, fornece uma medida da superposição do movimento da cabeça e o movimento de tais pontos. No entanto, deseja-se observar apenas o movimento dos pontos da face, e dessa forma deve-se subtrair do movimento total, a movimentação da cabeça. Neste sentido, determina-se um novo sistema de coordenadas cartesianas, localizado sobre a cabeça do locutor (Figura 1), de tal forma que o movimento relativo entre a cabeça do locutor e o sistema de coordenadas seja nulo, ou seja, as trajetórias rastreadas em relação ao novo sistema de coordenadas corresponderão apenas ao movimento dos pontos em questão. Os eixos $\hat{X}$ e $\hat{Y}$ pertencem ao plano frontal, e o eixo $\hat{Z}$ é perpendicular ao plano formado por $\hat{X}$ e $\hat{Y}$.

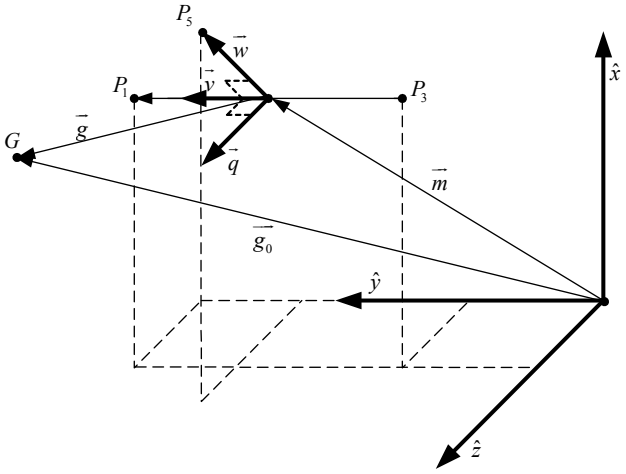

Figura 1 – Criação de um sistema de coordenadas localizado na cabeça do locutor.

Inicialmente têm-se os pontos $\mathbf{P_1}$, $\mathbf{P_3}$ e $\mathbf{P_5}$ localizados sobre a cabeça do locutor. Pode-se determinar o ponto médio do segmento $\mathbf{P_1P_3}$, que corresponderá ao vértice do novo sistema de coordenadas.

$$\vec{m} = \frac{\vec{p}_1 + \vec{p}_3}{2} \tag{1}$$

Na seqüência pode-se determinar os vetores na direção do segmento que liga o ponto médio ao ponto $\mathbf{P_1}$ e o vetor na direção do segmento que liga o ponto médio ao ponto $\mathbf{P_5}$, respectivamente.

$$\vec{v} = \vec{p}_1 - \vec{m} = \frac{\vec{p}_1 - \vec{p}_3}{2} \tag{2}$$

e

$$\vec{w} = \vec{p}_5 - \vec{m}. \tag{3}$$

Por último calcula-se o vetor ortogonal ao plano formado pelos vetores obtidos anteriormente através de um produto vetorial:

$$\vec{q} = \vec{w} \times \vec{v}. \tag{4}$$

As trajetórias dos pontos representadas no novo sistema de coordenadas podem ser obtidas pelas projeções do vetor $\vec{g}$ sobre os novos eixos definidos pelos vetores $\vec{w}$, $\vec{v}$ e $\vec{q}$, sendo $\vec{g}$ definido como:

$$\vec{g} = \vec{g}_0 - \vec{m}. \tag{5}$$

Os marcadores (LEDs) e o novo sistema de coordenadas estão ilustrados na Figura 2.

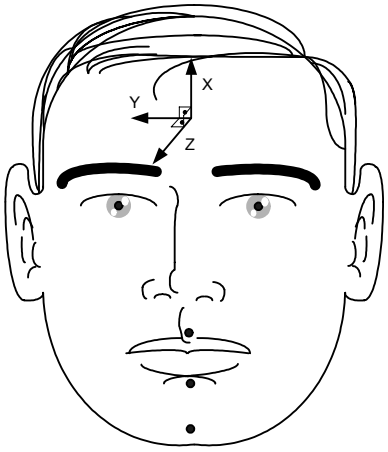

Figura 2 – Localização dos LEDs "•" (lábio superior, lábio inferior e queixo) e do novo sistema de coordenadas.

3. ANÁLISE DE FRONTEIRAS FONÉTICAS E ARTICULATÓRIAS

Os fenômenos acústicos observados durante o processo de produção da fala ocorrem como conseqüência da configuração do trato vocal e, dessa forma, estão ligados a eventos articulatórios. Deseja-se avaliar então quais eventos articulatórios se correlacionam com as unidades acústicas do sinal de fala.

Neste sentido, observam-se os instantes de ocorrência dos pontos de máximo e mínimo do sinal de trajetória do LED localizado no queixo, a fim de se analisar os instantes correspondentes no sinal acústico. O mesmo procedimento é adotado utilizando-se o sinal de abertura da boca, gerado a partir da diferença entre a trajetória do LED localizado no lábio superior e a do LED localizado no lábio inferior.

Procura-se observar a existência de regularidade na ocorrência dos eventos articulatórios representados pelos pontos de máximo e mínimo das trajetórias medidas, e se o processo de produção dos fones ocorre de forma semelhante nas vogais (/a/, /ɛ/, /i/, /y/, /u/, /ə/) e nas consoantes (/b/ e /z/) analisadas no experimento. Dessa forma, pode-se

inferir se a informação articulatória facilita a segmentação da sentença falada, nos casos em que a extração da informação acústica é mais delicada.

As frases são segmentadas manualmente em fones de acordo com os espectrogramas observados, e os instantes de ocorrência dos picos dos sinais de trajetória do queixo e de abertura da boca são projetados sobre tais sentenças. Assim, pode-se medir a posição relativa de ocorrência do pico dentro do segmento correspondente a um fone. Em algumas repetições da sentença, não se observam picos dentro do intervalo de produção de determinados fones, e em outras, ocorrem mais de um pico dentro do segmento de um mesmo fone acústico. Neste último caso, observa-se a média da ocorrência do pico nas demais repetições do fone e seleciona-se o pico mais próximo da média. Além disso, adota-se o critério de que todo pico ocorrido em uma posição menor que 1% do intervalo de duração do fone pertence ao fone anterior, isto é, alonga-se o intervalo anterior até o instante de ocorrência de tal pico. Tal critério é utilizado devido à possibilidade de existência de inconsistências na determinação precisa das fronteiras acústicas entre os fones.

Os procedimentos de análise são divididos em duas partes: projeções dos instantes de ocorrência dos picos do sinal de trajetória do LED localizado no queixo e projeções dos instantes de ocorrência dos picos do sinal de abertura da boca.

4. RESULTADOS

Os procedimentos de análise fornecem resultados sobre a organização temporal da produção fonética das vogais e consoantes analisadas no trabalho, baseados na informação articulatória. Tais resultados são observados para a trajetória do LED localizado no queixo e para a abertura da boca, como descritos na seqüência.

4.1 SINAL DE TRAJETÓRIA DA ABERTURA DA BOCA

Medem-se os instantes dos picos de máximo e mínimo do sinal de abertura da boca, relativos ao intervalo de duração do fone associado aos mesmos em termos percentuais.

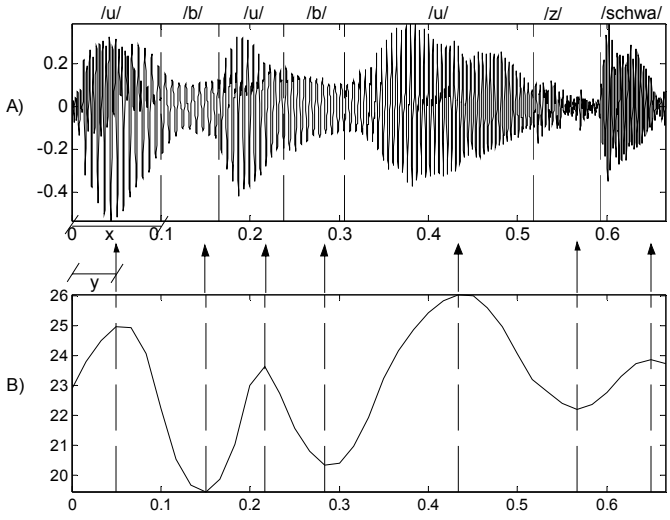

Figura 3 – Projeção dos eventos articulatórios sobre os segmentos acústicos. A) Segmentação acústica em fones do trecho "*ububuz*"; B) Picos no sinal de abertura da boca.

Dessa forma, os instantes de ocorrência dos picos (t_{picos}) são definidos em relação ao início do segmento acústico no qual se inserem, medido a partir do início do intervalo *x*, de acordo com a relação abaixo:

$$t_{pi\cos} = \frac{y}{x}, \tag{6}$$

onde *x* corresponde ao intervalo do fone obtido pela segmentação manual (Figura 3.A) e *y* ao intervalo de ocorrência do pico (Figura 3.B).

Em seguida, realiza-se o teste estatístico ANOVA sobre tais valores obtidos para as vogais (/a/, /ɛ/, /i/, /y/, /u/, /ə/) e para as consoantes (/b/ e /z/). Tal teste realiza uma comparação entre as médias dos conjuntos de dados de forma indireta, através da análise de variâncias, assumindo-se como hipótese nula que todos os dados provém da mesma população.

Os resultados do teste estatístico encontram-se nas tabelas 1, 2, 3 e 4.

Tabela 1 – Teste ANOVA para as vogais analisadas (/a/, /ɛ/, /i/, /y/, /u/, /ə/).

	/a/	/ə/	/ɛ/	/i/	/y/	/u/
p	0,128	0,572	0,981	0,203	0,242	0,109
F	2,45	0,610	0,02	2,04	1,67	2,86

Tabela 2 – Teste ANOVA para comparação entre vogais (/u/×/y/ e /a/×/y/×/u/).

	/u/×/y/	/a/×/y/×/u/
p	0,092	$5{,}06 \times 10^{-6}$
F	2,26	9,20

Tabela 3 – Teste ANOVA para as consoantes analisadas (/b/ e /z/).

	/b/	/z/
p	$7{,}24 \times 10^{-4}$	0,656
F	15,89	0,56

Tabela 4 – Teste ANOVA para comparação entre vogal e consoante (/i/ × /z/, /ə/ × /z/, /u/ × /z/).

	/i/ × /z/	/ə/ × /z/	/u/ × /z/
p	0,421	0,448	0,432
F	1,08	1,03	1,04

A Tabela 1 mostra que as vogais presentes na sentença são produzidas aproximadamente da mesma forma, quando se analisa cada uma separadamente, uma vez que o teste ANOVA fornece valores maiores que 0,05 para a variável estatística *p*. Porém, a Tabela 2 mostra que vogais distintas são produzidas de forma diferente quando comparadas entre si. Interessante notar que quando o formato labial na produção de fones é semelhante, como é o caso do /u/ e do /y/, o teste indica uma regularidade na produção dos mesmos, o que não ocorre para produção de fones de movimento articulatório muito diferentes como mostrado para os fones /a/, /y / e /u/.

No entanto, a Tabela 3 mostra que o fone /b/ não é produzido de forma independente da posição em que tal fone se insere na sentença, pois o valor de *p* está abaixo de 0,05. Dessa forma, pode-se interpretar que as vogais analisadas não se mostram sensíveis às consoantes vizinhas, enquanto que a consoante /b/ é afetada pelas vogais vizinhas, do ponto de vista articulatório. No caso do fone /z/, pode-se observar que a

informação articulatória extraída a partir do movimento de abertura da boca não é suficiente para evidenciar a influência das vogais vizinhas sobre tal fone. A tabela 4 mostra que o fone /z/ é produzido da mesma forma que as vogais vizinhas, comprovando tal fato. A informação articulatória provavelmente mais apropriada para a realização dos testes estatísticos, neste caso, deve ser a do movimento da língua.

Tabela 5 – Média e desvio padrão dos instantes de ocorrência dos picos nas vogais (/a/, /ə/, /ɛ/, /i/, /y/, /u/).

	/a/	/ə/	/ɛ/	/i/	/y/	/u/
μ_{picos}	0,545	0,552	0,648	0,610	0,729	0,697
σ_{picos}	0.115	0.227	0.319	0.212	0.108	0.09

Tabela 6 – Média e desvio padrão dos instantes de ocorrência dos picos nas consoantes (/b/ e /z/).

	/b/	/z/
μ_{picos}	0,739	0,854
σ_{picos}	0.135	0.116

Na Tabela 5 encontram-se as médias dos instantes de ocorrência dos picos nas vogais em análise. Pode-se observar que as médias estão compreendidas no intervalo medido em segundos $0{,}5 \leq \mu_{picos} \leq 0{,}80$. Tais valores sugerem que existe uma tendência da existência de eventos articulatórios em torno de 65% do intervalo de duração das vogais. Assim, pode ser interessante utilizar essa informação para a segmentação em difones, considerando-se neste caso que o início do difone não se encontra exatamente na metade do fone anterior, mas tem o início em torno de 65% do intervalo de duração do fone. Tal afirmação pode ser baseada no fato de que os eventos articulatórios são bastante relacionados com a estrutura acústica da sentença, pois a produção do sinal acústico ocorre a partir da configuração do trato vocal. A mesma conclusão pode ser estendida para as consoantes em análise, conforme mostra a Tabela 6.

4.2 SINAL DE TRAJETÓRIA DO PONTO NO QUEIXO

O mesmo procedimento é adotado na obtenção dos resultados utilizando-se o sinal de trajetória do LED localizado no queixo. Os valores obtidos através do teste estatístico encontram-se nas Tabelas 7, 8, 9 e 10.

Tabela 7 – Teste ANOVA para as vogais analisadas (/a/, /ɛ/, /i/, /y/, /u/, /ə/).

	/a/	/ə/	/ɛ/	/i/	/y/	/u/
P	0,304	0,390	0,100	0,837	0,712	0,446
F	1,32	1,31	5,45	0,05	0,15	0,66

Tabela 8 – Teste ANOVA para comparação entre vogais (/u/×/y/ e /a/×/y/×/u/).

	(/u/×/y/)	/a/×/y/×/u/
p	0,195	0,010
F	1,83	3,79

Tabela 9 – Teste ANOVA para as consoantes analisadas (/b/ e /z/).

	/b/	/z/
p	$2{,}35 \times 10^{-6}$	0,214
F	222,61	2,13

Tabela 10 – Teste ANOVA para comparação entre vogal e consoante(/i/ × /z/, /ə/ × /z/, /u/ × /z/).

	/i/ × /z/	/ə/ × /z/	/u/ × /z/
p	0,557	0,077	0,226
F	0,87	2,74	1,91

Os resultados obtidos com o sinal de abertura da boca são confirmados pelos resultados obtidos com a trajetória do marcador localizado no queixo. Assim, as conclusões dos testes descritos no item anterior podem ser estendidas para os testes com o LED localizado no queixo, observando-se também as Tabelas 11 e 12.

Tabela 11 – Média e desvio padrão dos instantes de ocorrência dos picos nas vogais (/a/, /ə/, /ɛ/, /i/, /y/, /u/).

	/a/	/ə/	/ɛ/	/i/	/y/	/u/
μ_{picos}	0,535	0,522	0,804	0,628	0,755	0,579
σ_{picos}	0.136	0.154	0.219	0.32	0.284	0.223

Tabela 12 – Média e desvio padrão dos instantes de ocorrência dos picos nas consoantes (/b/ e /z/).

	/b/	/z/
μ_{picos}	0,66	0,60
σ_{picos}	0.409	0.243

5. CONCLUSÕES

Percebe-se que a movimentação articulatória no arranjo frasal proposto para este trabalho fornece resultados que indicam a existência de correspondência entre o domínio articulatório e o domínio acústico a partir da informação extraída dos sinais de movimentação do queixo e abertura da boca, ou seja, os pontos de máximo e mínimo observados nos sinais espaciais. Tal informação pode ser utilizada para identificar fronteiras de unidades acústicas conhecidas (difones e trifones, no caso de síntese acústica da fala). Além disso, os resultados obtidos com o sinal de abertura da boca apresentam uma variabilidade menor do que os obtidos com o sinal de movimentação do queixo.

A observação de regularidade na produção de vogais do ponto de vista articulatório, tanto pela informação extraída da abertura da boca, como pela informação extraída da movimentação do queixo, revela a possibilidade da utilização deste dado para a distinção entre algumas vogais. Além disso, pode comprovar também que a consoante /b/ é sensível à presença das vogais vizinhas, enquanto que as vogais analisadas não se mostram influenciadas pelas consoantes mais próximas.

Sabe-se que a produção acústica depende diretamente da configuração do trato vocal e, portanto está fortemente ligada aos eventos articulatórios estudados no trabalho (pontos de máximo e mínimo do sinal de abertura da boca e de movimentação do queixo). No caso ideal, todos os segmentos determinados por eventos articulatórios deveriam cair sobre a posição central do fone, ou sobre o *onset* e *offset* dos mesmos, o que permitiria encontrar facilmente as fronteiras entre unidades acústicas tais como difones e trifones. Os resultados mostram que os eventos articulatórios não ocorrem exatamente no centro das vogais, mas se aproximam do mesmo. Neste sentido, pretende-se realizar estudos futuros a fim de reavaliar a definição das fronteiras de tais unidades acústicas, no intuito de verificar se a modificação na localização das fronteiras entre as unidades acústicas pode trazer ganhos para os sistemas de síntese (maior naturalidade) e reconhecimento de fala (menor taxa de erro), levando em consideração a relação

custo/benefício para a implementação do segmentador automático que também utiliza a informação articulatória. Além disso, o valor médio de ocorrência do pico em uma determinada posição dentro do fone pode ser utilizado como informação adicional em algoritmos de reconhecimento de fala.

Portanto, pode-se utilizar a informação de abertura da boca, por exemplo, de forma conjunta com a informação acústica durante o processo de determinação das fronteiras entre as unidades fonéticas, sempre que haja disponibilidade do material, a fim de se facilitar a segmentação.

6. AGRADECIMENTOS

Ao suporte financeiro dado pela FAPESP a Jaqueline Vieira Gonçalves, e a Sérgio Robertos Barros pela contribuição com o trabalho.

7. REFERÊNCIAS

Browman C. P. & Goldstein L. 1986. Towards an articulatory phonology. *Phonology Yearbook*, **3**:219-252.

Browman C. P. & Goldstein L. 1988. Some Notes on Syllable Structure in Articulatory Phonology. *Phonetica* **45**:140-155.

Fant G. 1973. Distinctive features and phonetic dimensions. In G. Fant *Speech sounds and features*. Cambridge, MA, 171-191.

Halle M. & Stevens K. N. 1979. Some Reflections on the theoretical bases of phonetics. In B. Lindblom & S. Ohman (Eds.) *Frontiers of speech communication research*. New York, Academic Press, 335-353.

Jakobson R., Fant C. G. M. & Halle M. 1969. *Preliminaries to speech analysis: The distinctive features and their correlates.* Cambridge, Mass: MIT Press. (MIT Acoustics Laboratory Technical Report 13.)

Ladefoged P. 1971. *Preliminaries to linguistic phonetics*. University of Chicago Press. 1971.

Benoît, C. 1996. *Synthesis and Automatic Recognition of Audio-Visual Speech*. The Institution of Electrical Engineers. London, UK.

Yehia H., Rubin P. & Bateson, E. V. 1998. Quantitative Association of Vocal-Tract and Facial Behaviour. *Speech Communication*, **26** (1-2): 23-43.