UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CURITIBA
CURSO SUPERIOR DE BACHARELADO EM ENGENHARIA ELÉTRICA

GUILHERME ANTONIO PAVELSKI

ANÁLISE DE RECONHECEDOR DE FALA DEPENDENTE DE LOCUTOR QUANDO USADO PARA LOCUTORES NÃO TREINADOS

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2016

GUILHERME ANTONIO PAVELSKI

ANÁLISE DE RECONHECEDOR DE FALA DEPENDENTE DE LOCUTOR QUANDO USADO PARA LOCUTORES NÃO TREINADOS

Trabalho de Conclusão de Curso de Graduação, apresentado como Trabalho de Conclusão de Curso do curso de Bacharelado em Engenharia Elétrica do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR), como requisito parcial para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Marcelo de Oliveira Rosa

CURITIBA
2016

Guilherme Antonio Pavelski

Análise de reconhecedor de fala dependente de locutor quando usado para locutores não treinados

Este Trabalho de Conclusão de Curso de Graduação foi julgado e aprovado como requisito parcial para a obtenção do Título de Engenheiro Eletricista, do curso de Engenharia Elétrica do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR).

Curitiba, 21 de novembro de 2016.

_____

Prof. Emerson Rigoni, Dr. Eng.
Coordenador de Curso
Engenharia Elétrica

_____

Profa. Annemarlen Gehrke Castagna, Ma.
Responsável pelos Trabalhos de Conclusão de Curso
de Engenharia Elétrica do DAELT

ORIENTAÇÃO

_____

Prof. Marcelo de Oliveira Rosa, Dr. Eng.
Universidade Tecnológica Federal do Paraná
Orientador

BANCA EXAMINADORA

_____

Prof. Marcelo de Oliveira Rosa, Dr. Eng.
Universidade Tecnológica Federal do Paraná

_____

Prof. Rafael Fontes Souto, Dr. Eng.
Universidade Tecnológica Federal do Paraná

_____

Prof. Antonio Carlos Pinho, Dr. Eng.
Universidade Tecnológica Federal do Paraná

_____

Prof. Gustavo Nishida, Dr.
Universidade Tecnológica Federal do Paraná

A folha de aprovação assinada encontra-se na Coordenação do Curso de Engenharia Elétrica.

AGRADECIMENTOS

Agradeço principalmente a quem acreditou que seria possível e a quem me incentivou durante todo esse tempo, assim como a quem não me abandonou.

Meu sincero obrigado aos professores que dispuseram de seu tempo para me orientar, avaliar, fazer correções e dar sugestões de como melhorar este trabalho, assim como a todos que auxiliaram quando puderam.

“E ela (a vitória) teria que vir assim, dramática.”

Galvão Bueno, 30 de julho de 2000.

RESUMO

PAVELSKI, Guilherme A. **Análise de reconhecedor de fala dependente de locutor quando usado para locutores não treinados**. 2016. 158 f. Trabalho de Conclusão de Curso (Graduação – Curso de Engenharia Elétrica). Universidade Tecnológica Federal do Paraná, 2016.

O presente documento se propõe a desenvolver uma avaliação de uma ferramenta de reconhecimento de fala para se analisar a variação da taxa de acertos em função da variação de parâmetros tanto vocais quanto inerentes à ferramenta utilizada. O enfoque foi dado ao caso em que grupos de locutores distintos são utilizados para treinar e testar, respectivamente, o sistema de reconhecimento. Discute-se a utilização de reconhecimento de fala baseado em Modelos Ocultos de Markov e algoritmos de varredura para encontrar o melhor resultado. Também foi desenvolvido um guia de utilização da ferramenta de forma a detalhar as etapas de montagem deste reconhecedor.

Palavras-chave: Reconhecimento de fala. Modelos Ocultos de Markov. HTK Toolkit. Algoritmo de Viterbi.

ABSTRACT

PAVELSKI, Guilherme A. **A speaker-dependent speech recognition tool analysis when used by untrained speakers.** 2016. 158 f. Trabalho de Conclusão de Curso (Graduação – Curso de Engenharia Elétrica). Universidade Tecnológica Federal do Paraná, 2016.

The present paper proposes to develop an evaluation of a speech recognition tool to analyze the accuracy rate variation against the parameters variations for both vocal features and peculiar parameters controlled by the respective tool. The focus in this study is the case where different speaker groups are used in order to training and testing the speech recognition system. This work discuss the use of speech recognition based on Hidden Markov Models and sweep line algorithms to search the best results. A tutorial for the tool was also developed in order to detail the steps of the recognition system preparation.

Keywords: Speech Recognition. Hidden Markov Models. HTK Toolkit. Viterbi Algorithm.

LISTA DE ABREVIAÇÕES

- *CMU Sphinx- Carnegie Mellon University Sphinx:* Sistema de reconhecimento desenvolvido pela Universidade Carnegie Mellon em Pittsburgh, Estados Unidos.
- *DNN – Deep Neural Network:* Rede neural profunda.
- *DFT – Discrete Fourier Transform.* Transformada discreta de Fourier. Equivalente da Transformada de Fourier para sinais amostrados.
- *DPCF - Difference phase-correlation-based function:* Critério que utiliza função baseada na diferença de correlação de fase.
- *FM – Frequency Modulation:* Modulação em frequência, técnica amplamente utilizada em rádio-difusão para a modulação de sinais.
- *FFT – Fast Fourier Transform*: Transformada rápida de Fourier, algoritmo computacional utilizado para se transformar a transformada de Fourier de um sinal.
- *HMM – Hidden Markov Model:* Modelo oculto de Markov.
- *HTK - Hidden Markov Model Toolkit:* Conjunto de ferramentas para Modelos Ocultos de Markov.
- *IDFT – Inverse Discrete Fourier Transform*: Inversa da Transformada Discreta de Fourier.
- *IPA - International Phonetic Alphabet*: Alfabeto fonético internacional que define a forma de representação dos fonemas em dicionários de todo o mundo.
- *LPC – Linear Predictive Coding:* codificação preditiva linear, forma de análise espectral.
- *MFC – Mel-Frequency Cepstrum:* Técnica utilizada em reconhecimento de fala que se baseia na transformada inversa de Fourier de um espectro na forma logarítmica analisando a saída em uma escala Mel de frequências.
- *MFCC – Mel-Frequency Cepstral Coefficients:* Forma de parametrização que se baseia em obter coeficientes que definam uma determinada forma de onda a partir de MFC.

LISTA DE FIGURAS

Figura 2.1 - Funcionamento básico do *software* HTK. 9
Figura 2.2 - Procedimentos utilizados durante o reconhecimento de fala. 10
Figura 2.3 – Espectrograma de frequências da nota musical Lá7 (7,04 kHz) amostrada a 8 kHz ($f_{sampling}$). Cores escuras indicam maior energia espectral. 12
Figura 2.4 - Segmentação de sinais de áudio para se criar o espectrograma. 13
Figura 2.5 - Transformação do conteúdo frequencial em níveis de Grey. 13
Figura 2.6 - Sinais de fala (esquerda) e seus respectivos espectrogramas (direita) para os dígitos zero, um e dois por um mesmo locutor. 14
Figura 2.7 - Espectrogramas para o sinal de voz de três locutores diferentes dizendo os dígitos zero (esquerda) e um (direita). 15
Figura 2.8 - Representação do aparelho fonador humano. 16
Figura 2.9 - Comparação dos sinais de fala masculino e feminino. 18
Figura 2.10 - Representação ilustrativa do ouvido humano e seus componentes. ...20
Figura 2.11 - Esquematização da cóclea e níveis de frequência perceptíveis. 21
Figura 2.12 - Faixa de amplitudes que o ouvido humano consegue suportar. 22
Figura 2.13 - Capacidade humana de reconhecer frequências. 23
Figura 2.14 - Faixa de frequências onde opera a audição humana. 23
Figura 2.15 - Correspondência entre as escalas de frequências. 25
Figura 2.16 - Procedimento para se gerar um *cepstrum*. 26
Figura 2.17 – Comparação entre espectros de frequência, sinal no tempo e seu respectivo *Cepstrum*. 28
Figura 2.18 - Função de ponderação triangular. 29
Figura 2.19 - Comparação entre os métodos de parametrização do sinal. 31
Figura 2.20 - FFT de um sinal não-periódico utilizando (esquerda) e não-utilizando (direita) enjanelamento. 32
Figura 2.21 - Comparação entre diversas janelas. 33
Figura 3.1 – Modelo discreto de Markov com 5 estados. 38
Figura 3.2 – Modelo oculto de Markov com 2 estados: os lados da moeda. 41
Figura 3.3 – Modelo oculto de Markov com 2 estados: moeda 1 e moeda 2. 41
Figura 3.4 – Modelo oculto de Markov com 3 estados: as três moedas. 42
Figura 3.5 – Explicação do método *forward-backward* no trecho *forward*. 48
Figura 3.6 – Explicação do método *forward-backward* no trecho *backward*. 49
Figura 3.7 - Descrição do método de Baum-Welch 51

Figura 3.8 – Máquina similar à descrita. 56
Figura 3.9 - Exemplo do cálculo das probabilidades. 57
Figura 3.10 - Apenas os estados de (t-1) e t interessam nessa fase. 58
Figura 3.11 - Reconstrução do caminho de sequência ótima 59
Figura 3.12 - Diagrama de blocos mostrando as etapas do reconhecimento. 60
Figura 4.1 - Parâmetros de conversão de áudio para MFCC. 65
Figura 4.2 - Comparação entre o tamanho dos vetores para diferentes formas de codificação 67
Figura 4.3 - Análise da variação do número de reestimações do caso inicial. 70
Figura 4.4 - Variação da janela com 7 reestimações e 150 vetores por segundo. 71
Figura 4.5 - Comportamento do reconhecedor com a variação da taxa de vetores de parâmetros por segundo. 72
Figura 4.6 - Comparação entre a variação do número de reestimações para dois casos diferentes. 73
Figura 4.7 - Variação do comprimento da janela com 5 reestimacões e 280 vetores de parâmetros por segundo. 74
Figura 4.8 - Comparação entre a taxa de acertos em relação ao número de reestimacões para três casos distintos. 75
Figura 4.9 - Variação da taxa de vetores por segundo para uma janela de 36,5 ms e 5 reestimações. 75
Figura 4.10 - Comparação entre os resultados dos diferentes testes ilustrados na Tabela 5 76
Figura 4.11 - Diagrama de pizza para análise dos erros vinculados a cada dígito 82
Figura 4.12 - Comparação entre os erros nos testes 1 e 2. 83
Figura 4.13 – Comparação entre os erros dos testes 4 e 5. 83
Figura 4.14 - Variação dos resultados para cada teste em função da codificação utilizada 85
Figura 4.15 - Comparação entre a taxa percentual de acertos e a variação do número de estados de cada HMM. 87
Figura 4.16 - Análise do percentual de acertos em função da variação da janela. 88
Figure 4.17 - Comparação entre a taxa de acertos e o número de reestimações no teste 6 para diferentes janelas. 89
Figura 4.18 – Análise da variação da taxa de acertos em função da variação da taxa de vetores de parâmetros por segundo, teste 6, 9 reestimações, janela de 49 ms, “MFCC_E_D_Z”. 90
Figura 4.19 – Comparação entre testes utilizando o novo conjunto de parâmetros de modo a otimizar os acertos do teste 6. 90

Figura A.1 - Registro para download do pacote necessário. 100
Figura A.2 - Escolhendo a versão de download do HTK. 101
Figura A.3 - Mensagem de configuração completa no terminal. 103
Figura A.4 - Erro de compilação do HTK. 103
Figura A.5 - Erro encontrado no arquivo de instalação. 104
Figura A.6 - Convertendo taxa de amostragem do áudio no *Audacity*. 109
Figura A.7 - Sequência de ações a serem executadas no reconhecimento. 113
Figura A.8 - Diferentes formas de dizer ao HTK onde estão as HMMs. 125
Figura C.1 - Modelo direita-esquerda de 4 estados. 150
Figura C.2 - Diagrama de blocos representando a análise LPC/*Cepstral*. 153

LISTA DE TABELAS

Tabela 1 - Dicionário fonético montado para o reconhecimento. 11

Tabela 2 - Cálculo das probabilidades do modelo de Markov de dois estados mostrado na Figura 3.3. 41

Tabela 3 - Cálculo das probabilidades de cada estado no modelo de Markov detalhado na Figura 3.4. 42

Tabela 4 - Matriz de estados para N globos com M bolas. 43

Tabela 5 - Especificação dos grupos de treinamento e teste. 63

Tabela 6 - Descrição dos parâmetros estudados no âmbito deste trabalho. 68

Tabela 7 - Variantes vocais utilizadas para treinar o HTK no teste 6. 78

Tabela 8 - Análise da performance de um reconhecedor para diferentes expressões vocais..... 79

Tabela 9 - Análise dos erros de dígitos cometidos durante cada teste 81

Tabela B.1 - Variação da taxa de acertos em função do número de reestimações, com janela de 20 ms e 150 vetores/s. 138

Tabela B.2 - Taxa de acertos em função da janela. 150 Vetores, 7 reest., teste 0. 139

Tabela B.3 - Análise da taxa de acertos com a variação da taxa de vetores de parâmetros para janela de 20 ms e 7 reestimações. Teste 0. 140

Tabela B.4 - Variação da taxa de acertos de acordo com a variação do número de vetores por segundo. Dados: Janela de 35 ms, 7 reestimações e teste 0. 141

Tabela B.5 - Taxa de acertos com variação do número de reestimações. Janela de 35 ms e 280 vet./s 142

Tabela B.6 - Taxa de acertos em função do tamanho da janela para 280 vet., 5 Reest., T= 0. 143

Tabela B.7 - Variação da taxa de acertos em relação à variação do n° de reestimações para 280 Vet./s; janela de 36,5 ms e teste 0. 144

Tabela B.8 - Comparação entre os testes para janela de 36,5 ms, 280 vet./s e 5 reestimações. 144

Tabela B.9 - Análise relativa à variação da taxa de vetores/s, com janela de 36,5 ms e 5 reestimações. 145

Tabela B.10 - Comparação entre os diferentes tipos de codificação e seus percentuais de acerto para cada um dos testes. 145

Tabela B.11 - Percentual de acertos para cada teste, considerando HMMs de 3, 5 e 7 estados, com 5 reestimações, 280 vetores de parâmetros por segundo e um comprimento de janela de 36,5 ms. 146

Tabela B.12 – Comparação entre a taxa de acertos (%) (Acc) em função da variação do comprimento de janela (ms) (Jan), para 2 reestimações, “MFCC_E_D_Z”, 280 vetores de parâmetros por segundo.....147

Tabela B.13 – Variação do número de reestimações, com diferentes tamanhos de janela.....148

Tabela B.14 – Variação da taxa de acertos em função da taxa de vetores por segundo no teste 6, para janela de 49 ms. 148

Tabela B.15 – Variação do percentual de acertos em função do teste executado, considerando-se 9 reestimações, 280 vetores de parâmetros/s, janela de 49 ms, HMM de 5 estados, codificação “MFCC_E_D_Z”.....149

SUMÁRIO

1. INTRODUÇÃO 1
1.1 TEMA 1
1.1.1 Delimitação do Tema 2
1.2 PROBLEMAS E PREMISSAS 3
1.3 OBJETIVOS 5
1.3.1 Objetivo Geral 5
1.3.2 Objetivos Específicos 5
1.4 JUSTIFICATIVA 6
1.5 ESTRUTURA DO TRABALHO 7
2. TÉCNICAS DE PROCESSAMENTO DIGITAL DE SINAIS 8
2.1 PROCEDIMENTOS METODOLÓGICOS 8
2.2 CLASSIFICAÇÃO FONÉTICA 10
2.3 ESPECTROGRAMAS 12
2.4 O APARELHO FONADOR HUMANO 15
2.5 O SISTEMA AUDITIVO HUMANO 18
2.5.1 Funcionamento do sistema auditivo humano 20
2.6 FERRAMENTAS UTILIZADAS PELO HTK 24
2.7 A ESCALA MEL 24
2.8 MEL-FREQUENCY CEPSTRAL COEFFICIENTS (MFCC) 25
2.8.1 Cálculo do MFCC 29
2.9 ENJANELAMENTO 31
2.10 FILTRO PRÉ-ÊNFASE 33
2.11 TÉCNICAS ATUAIS DE RECONHECIMENTO DE FALA 34
2.11.1 Robustez 34
2.11.2 Cocktail Party 35
2.11.3 Redes neurais profundas 35
3. MODELOS OCULTOS DE MARKOV 37
3.1 PROCESSOS DISCRETOS DE MARKOV 38
3.2 PROCESSOS OCULTOS DE MARKOV 40
3.2.1 Definições de HMMs 43
3.2.2 Os três problemas básicos em modelos ocultos de Markov 44

3.2.3 Soluções para os três problemas básicos de HMM.....45
3.3 ALGORITMO DE VITERBI.....54
3.4 RECONHECIMENTO DE FALA UTILIZANDO HMMs.....60
3.4.1 Reconhecimento de palavras isoladas.....61
4. DESENVOLVIMENTO, RESULTADOS E DISCUSSÕES.....62
4.1 DESENVOLVIMENTO DO RECONHECEDOR.....62
4.2 RESULTADOS E DISCUSSÕES.....69
4.3 COMPARAÇÃO ENTRE DIFERENTES TREINAMENTOS E TESTES.....76
4.4 COMPARAÇÃO ENTRE DIFERENTES FORMAS DE CODIFICAÇÃO.....84
4.5 COMPARAÇÃO ENTRE DIFERENTES NÚMEROS DE ESTADOS DO MODELO DE MARKOV.....86
4.6 OTIMIZANDO O CASO DEPENDENTE DE LOCUTOR.....88
5. CONCLUSÃO.....92
REFERÊNCIAS.....94
APÊNDICE A – TUTORIAL PRODUZIDO PARA UTILIZAÇÃO DO HTK.....99
A.1 INTRODUÇÃO.....99
A.2 INSTALAÇÃO DO HTK.....99
A.3 DESENVOLVENDO A GRAMÁTICA E COLETANDO DADOS.....105
A.4 PARAMETRIZAÇÃO.....110
A.5 DEFININDO AS HMMS.....113
A.6 REESTIMANDO O MODELO DE MARKOV.....122
A.7 TESTANDO O RECONHECEDOR.....127
A.8 ANALISANDO OS RESULTADOS.....129
A.9 UTILIZANDO O SCRIPT.....130
APÊNDICE B – TABELAS UTILIZADAS PARA PLOTAR OS RESULTADOS.....138
APÊNDICE C – DETALHES ADICIONAIS SOBRE HMMs.....150
C.1 TIPOS DE HMM.....150
C.2 ANÁLISE LPC-*CEPSTRAL*.....153

1. INTRODUÇÃO

1.1 TEMA

O ser humano vive em constante adaptação, buscando descobertas que pudessem auxiliar seu desenvolvimento desde os primórdios, e foi assim durante toda sua história. A evolução humana está vinculada a criar formas de estender suas capacidades e, em muitos casos, criar soluções inovadoras que muitas vezes estão ligadas ao seu modo de vida tradicional, seus movimentos mais básicos, suas características primordiais. Assim que o ser humano necessitou de formas de se comunicar com outros de sua espécie, e também com a natureza, criou maneiras pelas quais poderia se fazer entendido por meio de sons, movimentos e símbolos que unidos formavam uma linguagem pela qual poderia se expressar. Os avanços tecnológicos que aconteceram com o passar dos anos possibilitaram novas descobertas. O que até ontem parecia obra de ficção, hoje já é uma realidade e figura no cotidiano das pessoas. A fala que desde muito cedo se mostrou uma das mais poderosas ferramentas de comunicação humana, contribuindo na organização e na formação das sociedades, um dia seria transmitida e ouvida mesmo de longas distâncias. Criaram-se serviços de telefonia, os transmissores e receptores de radiofrequência, a televisão, os gravadores de voz, todos vinculados à capacidade humana de produzir sons, mas a necessidade que inicialmente era de produzi-los, tornou-se a de transmiti-los e hoje também é de reconhecê-los.

Segundo [1], as duas últimas décadas testemunharam um progresso assombroso na tecnologia de reconhecimento de fala, no sentido da maior disponibilidade de algoritmos e sistemas de alto desempenho. Em muitos casos, a transição de protótipos de laboratório para sistemas comerciais já se iniciou.

Atualmente, é chegado o momento em que tais sistemas estão presentes de forma numerosa nos hábitos da população, inclusive em ferramentas de telefonia móvel em que o operador aciona sistemas de posicionamento global sem apertar qualquer botão, somente utilizando comandos vocais.

1.1.1 Delimitação do Tema

Antes de discorrer sobre a análise de padrões acústicos vocais, deve-se primeiramente descrever alguns termos técnicos que aparecerão durante a leitura do trabalho.

- Locutor é aquele que está empregando sua fala para a análise da ferramenta.
- Fala contínua é a locução de frases de uma forma contínua, ou seja, sem pausas entre as palavras, tal como em um diálogo no cotidiano.
- Fala discreta é a locução de palavras com intervalos (pausas) entre cada uma delas. Pode ser utilizada para descrever comandos como “Ligue!”, “Desligue!”, dentre outros.
- Dependência de locutor significa que uma ou mais pessoas treinam e utilizam a ferramenta.
- Independência de locutor pode sugerir que diversas pessoas estão no mesmo ambiente e a ferramenta tenta analisar os comandos de fala delas ou, ainda, que pessoas distintas estariam utilizando a ferramenta. Neste caso, não haveria a necessidade do treinamento de fala destas pessoas.

A abordagem desse trabalho consiste em estudar um sistema que é independente do locutor, utilizando análise de fala isolada. Vale ressaltar que, embora o título se refira a uma abordagem dependente do locutor, foi estudado, apenas no teste 6, a abordagem dependente do locutor. Para os outros testes, um número maior de locutores foram utilizados no treinamento, o que sugere uma independência de locutor.

Esse trabalho se propõe a analisar uma ferramenta de reconhecimento de fala e seu desempenho quando treinada por um locutor e testada por outros locutores distintos, visando identificar quais são as particularidades presentes quando se analisam determinados fonemas, palavras e frases, as diferenças de desempenho quanto ao timbre do locutor, a altura de sua voz, dentre outros aspectos.

Também se pretende através de análise determinar quais são os padrões que retornam os melhores resultados, por exemplo, para quais fones a taxa de acerto é maior, identificar se a semelhança entre timbres pode melhorar o desempenho, diferenças de desempenho quando se utilizam frases balanceadas e desbalanceadas,

entre outros aspectos, pois tudo isso pode contribuir para descobrir formas de aprimorar os sistemas de reconhecimento vocal e auxiliar na resolução de problemas.

No contexto atual, podem-se citar sistemas de ditado, interfaces entre homem e máquina para automatização de comandos através do reconhecimento de fala, serviços de telefonia automatizados e mesmo aplicações industriais específicas [2] como algumas das principais áreas de interesse desse trabalho. Esses sistemas têm trazido diversos ganhos em produtividade nas situações em que foram utilizados. Os sistemas de reconhecimento de fala também podem ser utilizados no controle de máquinas e dispositivos, abertura e fechamento de portas e válvulas, acionamento de lâmpadas, operações financeiras e outros [1].

Para muitas aplicações, o reconhecimento dependente de locutor é suficiente, desde que um dispositivo particular seja utilizado por uma única pessoa durante um período de tempo relativamente extenso, por exemplo, um turno de trabalho. Além disso, seria conveniente para algumas aplicações que o sistema pudesse fazer reconhecimento de palavras conectadas, uma vez que uma entrada por palavras isoladas pode ser muito lenta e desconfortável [1].

1.2 PROBLEMAS E PREMISSAS

As estratégias propostas nas últimas décadas para o reconhecimento de fala funcionam impondo pelo menos uma das seguintes restrições: (1) dependência do locutor; (2) palavras isoladas; (3) vocabulário pequeno; (4) gramáticas reduzidas; (5) contexto semântico limitado e (6) ausência de ruído de fundo. Ainda não existe nenhum sistema que trabalhe sem alguma das restrições acima, embora boa parte delas já tenha sido superada com técnicas que manipulam uma quantidade muito grande de dados de treinamento, tais como os Modelos Ocultos de Markov [3].

De acordo com [3], busca-se integrar as técnicas de reconhecimento utilizadas nas últimas décadas com modelos de linguagem natural, visando identificar não somente algumas palavras, mas investigando a formação das próprias frases e reconhecendo seu significado semântico através de modelos probabilísticos.

O desenvolvimento dos principais sistemas de reconhecimento de fala teve seu início em meados da década de 70, comparando-se cada palavra dita com

palavras armazenadas em bases de dados de referência, utilizando-se dependência de locutor e vocabulários limitados, obtendo-se ótimos resultados. Em 1982, os laboratórios Bell aplicaram técnica de *clustering* para criar padrões robustos para o reconhecimento de palavras isoladas com independência de locutor. Nos anos seguintes obtiveram-se resultados satisfatórios em sistemas com vocabulário maior (até 5000 palavras). Em 1988, o sistema *Sphinx* da CMU (*Carnegie Mellon University*) conseguiu realizar o reconhecimento de fala contínua independente do locutor para grandes vocabulários, utilizando unidades de fala treináveis e insensíveis ao contexto, com a possibilidade de se adaptar a locutores individuais. As modelagens de fones foram realizadas com Modelos Ocultos de Markov [4].

As principais dificuldades encontradas no desenvolvimento de sistemas de reconhecimento vocal são:

- Segmentação das unidades acústicas: Reconhecer cada unidade acústica, por exemplo, um fone ou trifone dentro de uma frase, e dizer onde começa e onde termina.
- Problemas de redução ou contração: Ocorre quando o locutor junta palavras e acaba suprimindo a pronúncia de alguns sons. Por exemplo: ‘caixa d’água’ e ‘estrela-d’alva’ ao invés de ‘caixa de água’ e ‘estrela de alva’.
- Problemas de coarticulação: A coarticulação ocorre quando um fone anteriormente dito altera a pronúncia do fone subsequente. Por exemplo, a nasalização do fone */ui/* na palavra ‘muito’, devido a presença da consoante nasal */m/* [5]. O espectro, neste caso, é bastante diferente do caso em que o fone */ui/* é dito isoladamente.
- Uma grande variabilidade que está presente na fala;
- A formalização do conhecimento linguístico: as frases se ouvem e se falam com um conhecimento linguístico, como a semântica, o que faz com que o ouvinte possa até predizer a próxima palavra do locutor baseado nas restrições de língua portuguesa contidas na frase sendo construída. Isto faz com que, em determinados casos, certas palavras não precisem ser ditas na frase [6]. Exemplificando, pode-se dizer: “vou fazer compras” no lugar de: “eu vou fazer as compras”

Atualmente, os sistemas reconhecedores de fala contínua começam a figurar com maior relevância frente aos reconhecedores de fala isolada que dominavam o

mercado até tempos recentes. Algumas ferramentas matemáticas associadas a algoritmos computacionais permitiram que a eficiência desses sistemas ganhasse notoriedade. Para aumentar a taxa de acertos, esses sistemas contam com modelos de linguagem estatísticos, a fim de favorecer palavras ou sequências de palavras mais frequentes [1].

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Verificar o grau de usabilidade de reconhecedor de fala treinado para um locutor quando usado para locutores distintos.

1.3.2 Objetivos Específicos

- Usar a ferramenta HTK (*Hidden Markov Models Toolkit*) para treinamento e reconhecimento de fala isolada e reconhecer padrões acústicos, podendo compreender melhor sua usabilidade dentro de diversas situações possíveis (timbres diferentes, sotaques, fala mais lenta ou mais rápida, dentre outras características).
- Criar um banco de dados de fala através da gravação de frases balanceadas, pois estas apresentam resultados interessantes quando são analisados seus fones.
- Identificar quais são as principais variações de resposta quando se comparam características de fala de locutores distintos e em quais casos o reconhecedor reage melhor; por exemplo, determinar se o reconhecedor reage melhor quando o timbre do locutor é semelhante ao do treinador ou se há grandes variações entre os resultados quando o locutor é do sexo masculino ou feminino.

1.4 JUSTIFICATIVA

Atualmente os sistemas de reconhecimento de fala vêm se mostrando úteis em diversas áreas do conhecimento, sendo uma área de interesse diversificado.

Pode-se, primeiramente, pensar em questões do mundo moderno como sistemas automatizados comandados por fala, ferramentas de busca a bases de dados comandadas sem a utilização de teclas, comandos residenciais como abrir o portão de uma garagem sem a necessidade de um controle remoto, apenas com um celular que utiliza ferramenta de reconhecimento vocal, ou mesmo ligar um eletrodoméstico sem a necessidade de ir até ele.

Outro caso que pode ser citado é uma situação dentro de um ambiente médico, onde, por conta da segurança hospitalar, um cirurgião não pode tocar um determinado equipamento, correndo um risco de contaminação, por exemplo. Uma solução de acionamento por fala poderia ser ideal neste caso.

Uma justificativa mais específica para esse trabalho é testar o limite de uso de reconhecedor treinado para uma pessoa e usado para outra pessoa, pois normalmente o que se tem é cada usuário fazendo um treinamento específico da ferramenta, mas, quando utilizado por outro usuário, torna-se pouco eficiente. Um propósito para estudar esse caso seria descobrir como aperfeiçoar a ferramenta, de forma a permitir que apenas um usuário padrão precise treinar a ferramenta, aumentando seu desempenho quando utilizada com usuários distintos.

Esse sistema passa a ser interessante quando a análise é feita no domínio temporal, enquanto diversos outros sistemas, tais como afinadores digitais de instrumentos musicais ou mesmo alguns outros métodos de reconhecimento de fala, fazem análise do espectro frequencial (utilizando séries de Fourier, por exemplo).

Dentro de um universo acadêmico, uma das justificativas é poder compreender melhor os sistemas de processamento digital de sinais, muito presentes atualmente em diversos ramos do conhecimento.

1.5 ESTRUTURA DO TRABALHO

O capítulo 1 introduz o que será desenvolvido nos capítulos subsequentes e apresenta uma abordagem mais fundamentada nos principais conceitos e uma contextualização do tema na atualidade.

No capítulo 2 será apresentada uma síntese das principais técnicas de processamento digital de sinais utilizadas na atualidade e um levantamento dos principais problemas encontrados com base no reconhecimento de fala. Será também apresentada uma abordagem geral do estado da arte para os sistemas de reconhecimento de fala em algumas aplicações.

O capítulo 3 será utilizado para a discussão dos modelos ocultos de Markov em uma abordagem mais rigorosa, assim como outras técnicas anteriormente utilizadas nesses sistemas.

O capítulo 4 discutirá como foi programado o reconhecedor de fala para realizar as experimentações propostas nos capítulos anteriores. O capítulo 4 também será utilizado para apresentação dos resultados experimentais e também serão feitas as conclusões e análise sobre os resultados obtidos.

O capítulo 5 se propõe a fazer uma discussão final sobre o trabalho, apontando dificuldades encontradas, abrangência e possibilidades futuras.

Também foi produzido o apêndice A, que mostra um tutorial desenvolvido pelo autor desse trabalho para utilização da ferramenta HTK, de forma a descomplicar certas etapas de preparação e utilização do reconhecimento de fala.

O apêndice B apresentará as tabelas utilizadas para se gerar os gráficos apresentados no capítulo 4.

No apêndice C será apresentada uma descrição teórica mais aprofundada sobre modelos ocultos de Markov, de forma a ajudar o leitor a compreender o funcionamento com um enfoque mais específico.

2. TÉCNICAS DE PROCESSAMENTO DIGITAL DE SINAIS

No presente capítulo serão apresentadas certas técnicas utilizadas atualmente no contexto do reconhecimento de fala, assim como definições necessárias para o entendimento do processo de produção e reconhecimento de fala.

2.1 PROCEDIMENTOS METODOLÓGICOS

Decidiu-se por criar um reconhecedor de dígitos em fala isolada, isto é, cada locutor falará apenas um dígito de cada vez, cujo sinal é separado por silêncio antes e depois. Os dígitos variam entre zero e nove. Para cada dígito são associados determinados fones relativos à sua pronúncia na língua portuguesa. Após a listagem dos fones, é necessário preparar um banco de dados com diferentes vozes, que gravarão cada uma, os dígitos citados.

Em seguida, cada arquivo de áudio servirá para preparar dados para treinamento da ferramenta HTK. Devem-se realizar as gravações com microfone, armazenando o áudio em 16000 Hz, 16 bits por amostra, canal mono ou estéreo. O áudio gravado em muitos aplicativos tem padrão de frequência de amostragem em 44100 Hz. Assim, pode ser necessário converter para 16000 Hz por questão de praticidade da ferramenta, mas normalmente o HTK trabalha com diversas taxas de amostragem. Todos os equipamentos necessários para esse procedimento se encontram em um laboratório dentro da UTFPR.

Para a maior parte dos procedimentos, será utilizada a ferramenta computacional HTK, que é uma ferramenta necessária para a construção de HMMs (Modelos Ocultos de Markov), que podem ser utilizados para modelar qualquer série temporal.

De toda forma, o HTK foi primeiramente desenvolvido para construção de HMMs baseadas em ferramentas de processamento da fala, em especial reconhecedores. Então, a maior parte da infraestrutura da ferramenta é dedicada a essa tarefa.

Há dois estágios principais de processamento de fala envolvidos. Primeiramente, as ferramentas de treinamento do HTK são usadas para estimar os

parâmetros de um conjunto de HMMs, utilizando declarações de treinamento e suas transcrições associadas. Depois, declarações desconhecidas são transcritas utilizando-se as ferramentas de reconhecimento do HTK.

Deixa-se claro que o método utilizado em todos os procedimentos será o dependente do locutor, ou seja, um grupo de locutores irá treinar, mas será testado e validado para diversos locutores distintos, obtendo-se resultados de como se portaria a ferramenta se fosse utilizada por outras pessoas que não os próprios treinadores.

A Figura 2.1 representa o funcionamento do *software* de reconhecimento *HTK* de uma forma genérica:

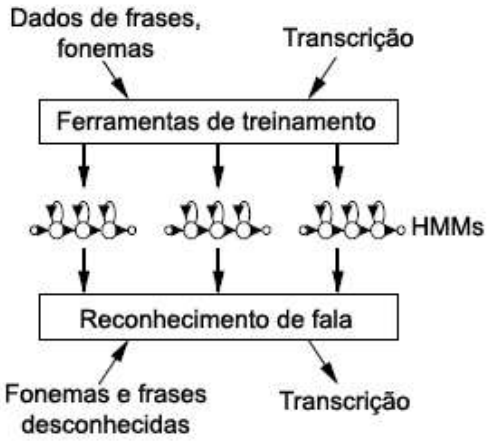

Figura 2.1 - Funcionamento básico do *software* HTK.
Fonte: [7]

Após a formação do banco de dados com as palavras e frases foneticamente balanceadas, é feita a segmentação dos trechos gravados, isto é, dizer para o programa que determinado espectro temporal corresponde a tal fone ou tal sequência de fones corresponde a tal espectro de fala temporal.

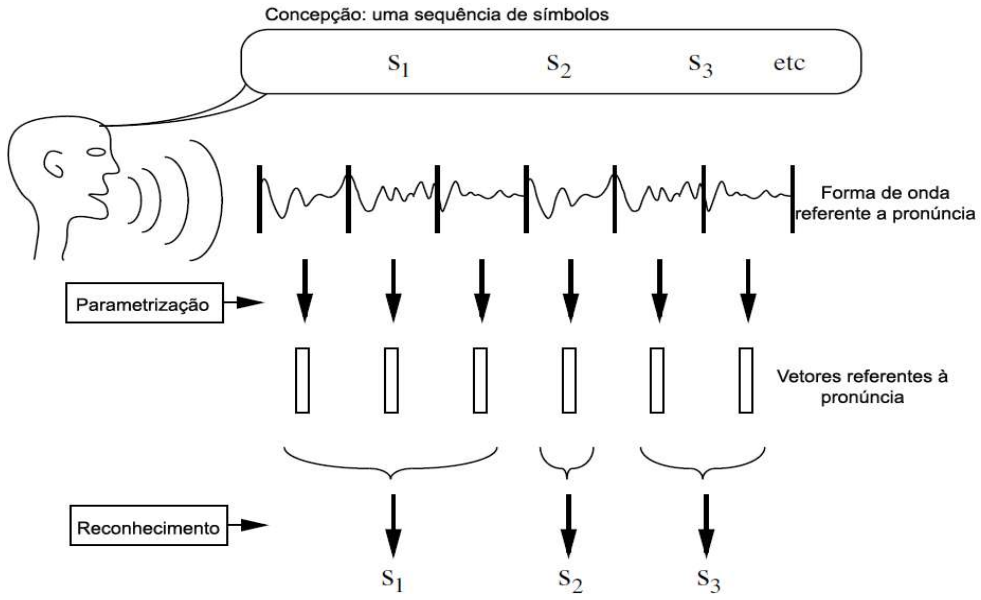

Figura 2.2 - Procedimentos utilizados durante o reconhecimento de fala.
Fonte: [7]

A Figura 2.2 demonstra as etapas e procedimentos que serão utilizados pelo HTK para reconhecer os dados de fala contínua.

Foram coletadas amostras de fala de 35 locutores, cada um deles pronunciando os dígitos entre zero e nove. Alguns destes dados serão utilizados para o treinamento do *software* de reconhecimento e outros, gravados por diferentes locutores, para os testes e obtenção dos resultados.

2.2 CLASSIFICAÇÃO FONÉTICA

Os segmentos fonéticos, para além de se distinguirem pela presença ou ausência de vozeamento, são ainda diferenciados por classes, dependendo do modo de articulação:

- Vogais;
- *Glides*;
- Oclusivas;
- Fricativas;

- Nasais
- e líquidas.

Dentro de cada classe, os segmentos fonéticos distinguem-se, ainda, pelo ponto de articulação no trato vocal. Para representar cada um dos segmentos fonéticos é utilizado um alfabeto fonético, sendo o mais conhecido o alfabético fonético internacional (IPA - *International Phonetic Alphabet*).

No âmbito deste trabalho, será utilizado o seguinte dicionário fonético, detalhado na Tabela 1, assim como os símbolos fonéticos e separação silábica.

Tabela 1 - Dicionário fonético montado para o reconhecimento.

Dígito	**Fonético**	**Separação silábica**
Zero	[zɛro]	Ze-ro
Um	[ũ]	Um
Dois	[dois]	Do-is
Três	[tres]	Três
Quatro	[kuatro]	Qua-tro
Cinco	[sĩko]	Cin-co
Seis	[seis]	Se-is
Sete	[sɛte]	Se-te
Oito	[oito]	Oi-to
Nove	[nɔve]	No-ve

Para elucidar os conceitos de ferramentas vinculadas ao tema do trabalho, é necessário entender alguns tópicos em que serão apresentados conceitos de processamento digital de sinais.

Primeiramente, é necessário compreender como se definem os espectrogramas.

2.3 ESPECTROGRAMAS

O espectrograma é um gráfico que é construído a partir da relação entre frequência, intensidade e tempo. Com ele se pode trabalhar a percepção de alguns parâmetros vocais como a frequência (graves e agudos são densos em baixas e altas frequências, respectivamente), controle da intensidade, pausa, etc. A forma usual para gerar o espectrograma é traçar diferentes cores para indicar a intensidade da densidade espectral de energia, variando do violeta ao vermelho do espectro visível [8].

Um exemplo de espectrograma é dado pela Figura 2.3, gerada pelo *software Matlab*. Este gráfico analisa dinamicamente a densidade de energia.

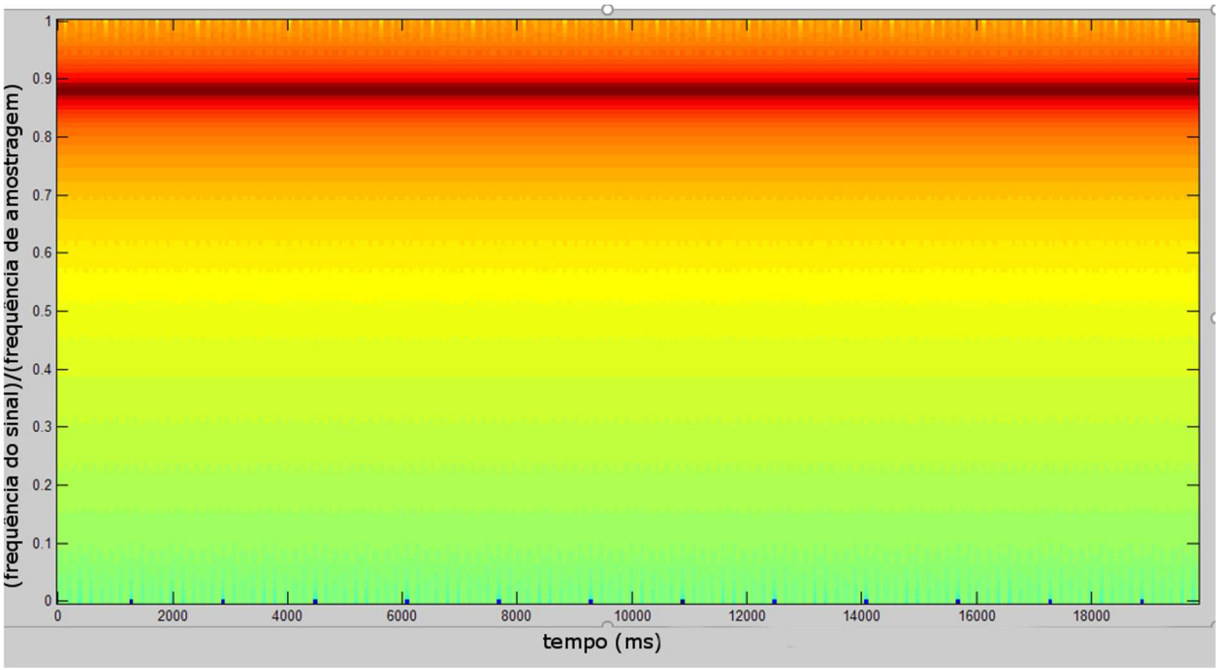

Figura 2.3 – Espectrograma de frequências da nota musical Lá7 (7,04 kHz) amostrada a 8 kHz ($f_{sampling}$). Cores escuras indicam maior energia espectral.
Fonte: Produção própria, *Matlab*.

Uma forma de se traçar um espectrograma vocal é mostrado a seguir nas Figuras 2.4 e 2.5:

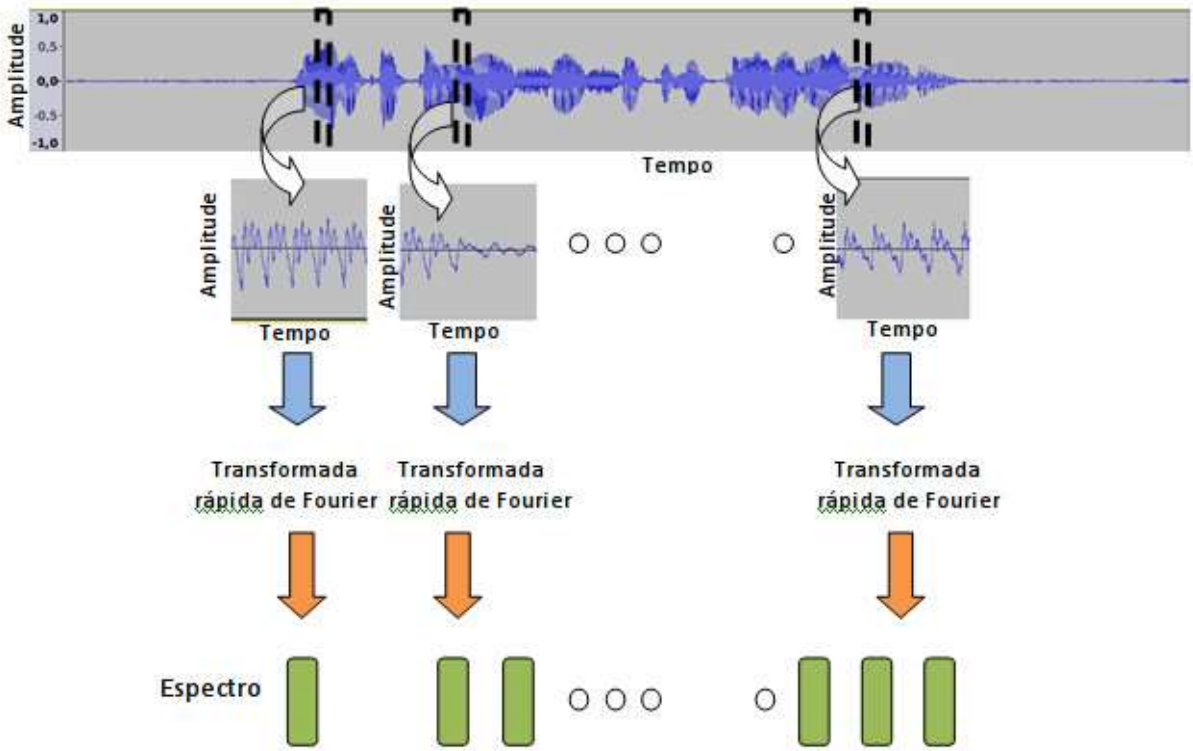

Figura 2.4 - Segmentação de sinais de áudio para se criar o espectrograma.
Fonte: Produção própria.

Primeiramente o sinal de áudio é dividido em N pequenos trechos individuais, como uma segmentação de passagens do sinal original e a cada um desses “pacotes” do sinal é aplicada uma transformada de Fourier para se obter um espectro frequencial de cada uma das partes segmentadas:

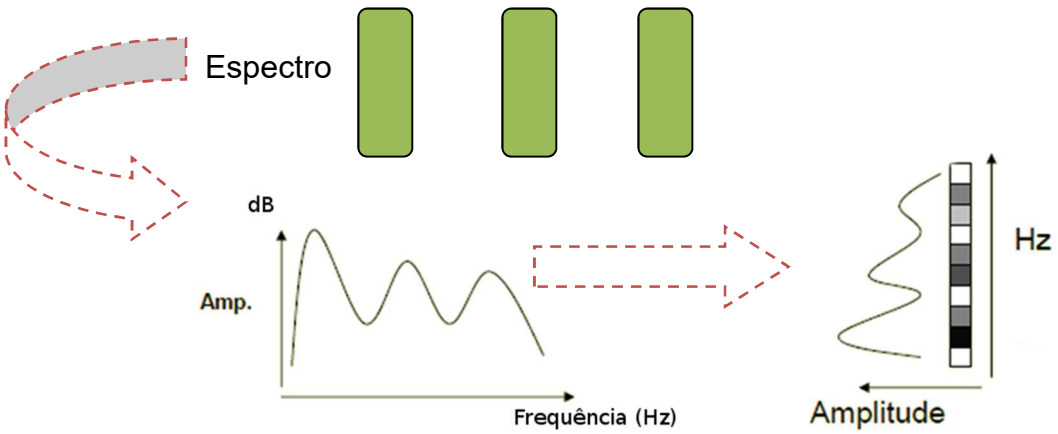

Figura 2.5 - Transformação do conteúdo frequencial em níveis de Grey.
Fonte: Produção própria.

Então, o conteúdo frequencial deste sinal é rotacionado de um ângulo de 90° (esta rotação é apenas para efeito visual) e cada trecho de amplitude espectral é convertido em um nível Grey (entre 0 e 255), dependendo de sua intensidade. O nível 0 representará a cor mais escura e 255 representará a cor mais clara, quanto maior for a amplitude frequencial, mais escura será a região correspondente.

A Figura 2.6 mostra alguns dos sinais que serão utilizados para o reconhecimento e seus respectivos espectrogramas:

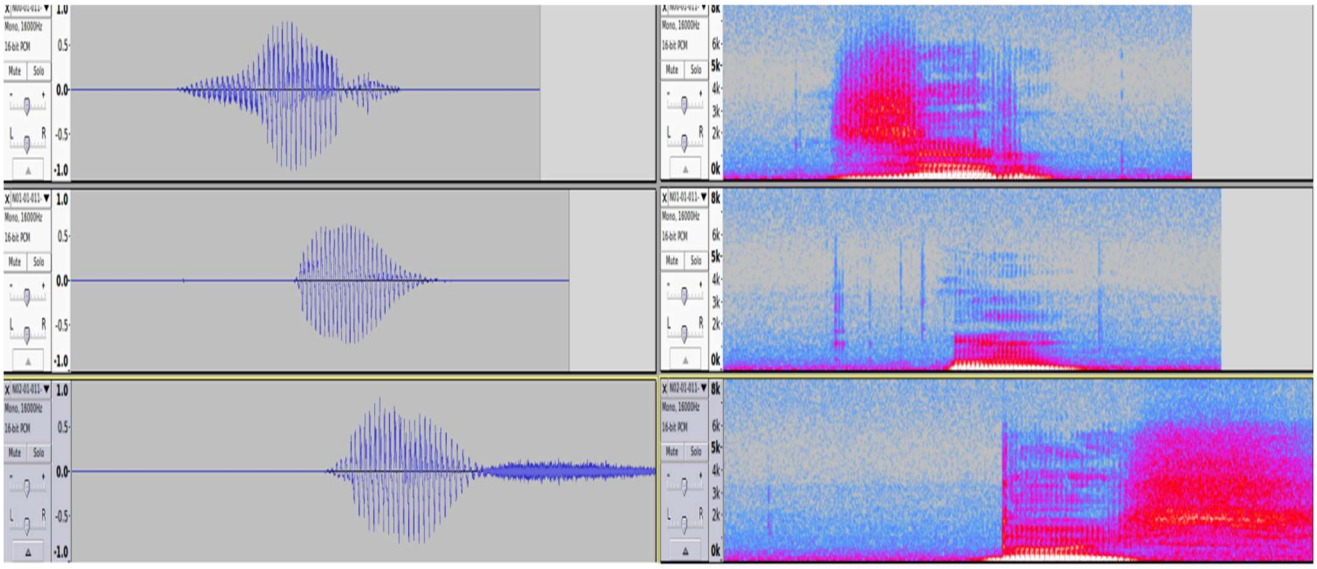

Figura 2.6 - Sinais de fala (esquerda) e seus respectivos espectrogramas (direita) para os dígitos zero, um e dois por um mesmo locutor.

Nota-se, pelo último espectrograma da Figura 2.6 uma quantidade considerável de ruído após a locução do dígito, assim como sua influência no espectrograma. Normalmente tecnologias de reconhecimento de fala trazem ferramentas que tornam possível a identificação mesmo com ruído de fundo. Em um dos arquivos de fala gravados, o som de uma televisão ligada era claramente audível no fundo da locução, na maior parte do tempo, isto não influenciou na propabilidade de acertos.

A Figura 2.7 compara os espectrogramas de três locutores distintos dizendo os dígitos zero (esquerda) e um (direita). Cada linha representa um locutor.

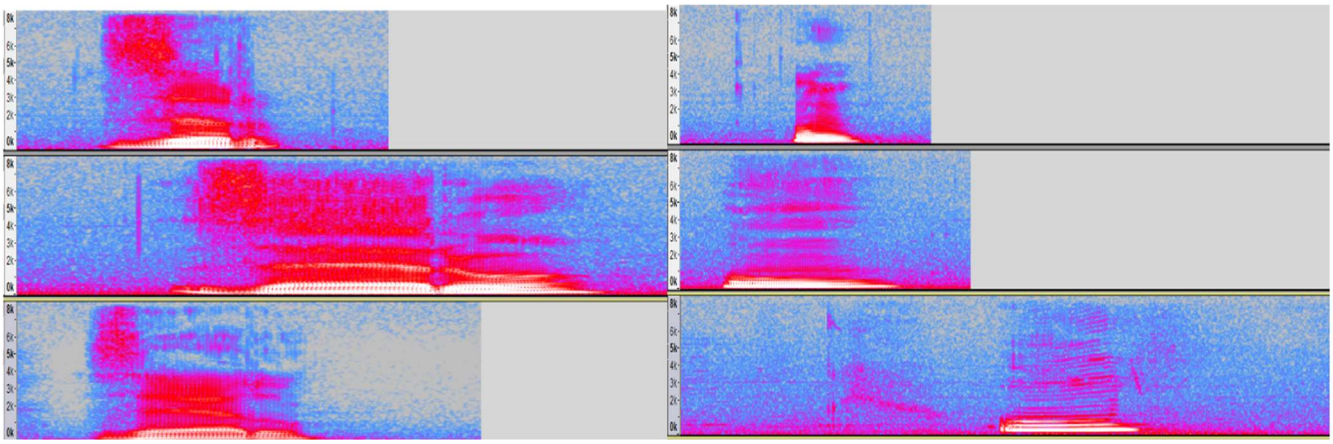

Figura 2.7 - Espectrogramas para o sinal de voz de três locutores diferentes dizendo os dígitos zero (esquerda) e um (direita).

Após a obtenção de dados sobre a taxa de acertos via HTK, será necessário analisar os dados e então escrever as conclusões relativas ao estudo desses sinais.

Antes de tentar entender melhor certas tecnologias utilizadas na análise e reconhecimento de sinais de fala, é necessário compreender como esses sinais são gerados e recebidos pelo organismo humano. Os próximos tópicos abordarão um contexto sobre a produção de fala e o sistema auditivo humano.

2.4 O APARELHO FONADOR HUMANO

De acordo com [9], a produção oral específica de uma única palavra só é realizada através de uma sincronia perfeita dos movimentos articulatórios que, se coordenados e articulados no tempo certo, com a tensão ideal e sequência precisa, tenderão a emitir os sons e fones precisos para a produção dessa determinada palavra. Determinados estudos com pacientes apresentando algum distúrbio de linguagem [10] confirma a existência de múltiplas áreas cerebrais envolvidas na produção de fala. Essa precisão e especialização dos movimentos de fala apresentam representação cortical definida que se dá no córtex motor primário.

Após a inalação de ar nos pulmões, os sinais de fala são produzidos durante a fase de exalação. O fluxo de ar, com a vibração das pregas vocais situadas na laringe, excita o trato vocal, constituído pela faringe, cavidade bucal, língua, lábios e

dentes. Para a produção de sons nasalados, o véu palatino abre, onde o ar, depois de passar pelo trato nasal, é radiado pelas narinas [9].

A corrente expiratória, ao sair da laringe, entra na cavidade da faringe que lhe oferece duas vias de acesso ao exterior: o canal bucal e o nasal, dependendo do canal, o som sairá de determinada forma, por exemplo:

/a/ (oral), */ã/* (nasal)

Conforme as palavras:

/a/ lá (oral), */ã/* lã (nasal)
/a/ mato (oral), */ã/* manto (nasal)

A Figura 2.8 ilustra o aparelho fonador humano, destacando as cavidades bucal e nasal.

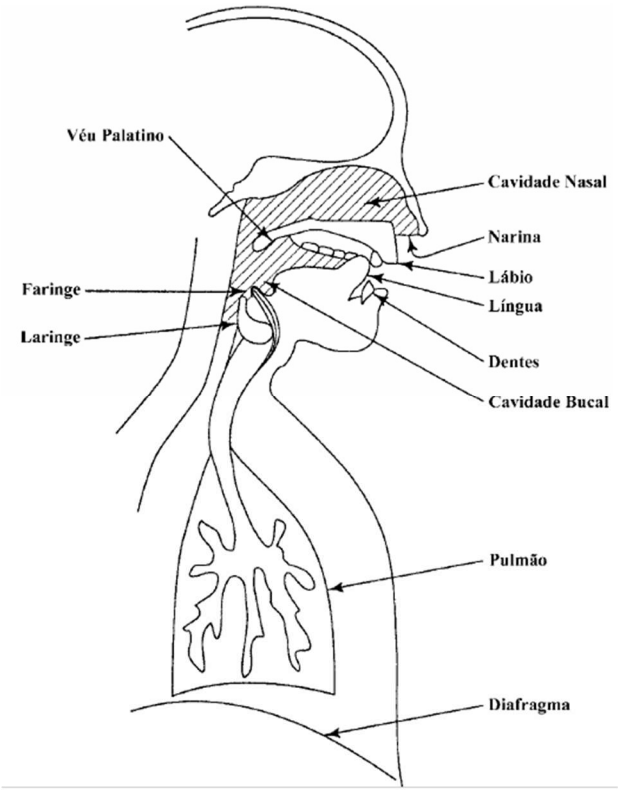

Figura 2.8 - Representação do aparelho fonador humano.
Fonte: http://blogdelinguistica.blogspot.com.br/2011/04/funcionamento-do-aparelho-fonador.html
Acesso em 30 nov. 2016, 19h57.

Segundo [9], há dois tipos de sons produzidos pelo aparelho fonador: os vozeados, produzidos com a vibração das pregas vocais, ou seja, com a abertura e fechamento da glote (espaço entre as pregas vocais) e os não-vozeados, produzidos com ausência de vibração destas. Nas zonas vozeadas, a taxa de abertura da glote varia de acordo com a vibração das pregas, da mesma forma como o volume de ar proveniente dos pulmões, que passa através da glote, variará. É esta variação periódica na velocidade do volume de ar na glote que vai excitar o trato vocal, produzindo sons com harmônicos da frequência de vibração das cordas vocais, ou seja, da frequência fundamental (f_0). Já nas zonas não-vozeadas, a glote mantém-se aberta e o ar proveniente dos pulmões, ao passar com suficiente velocidade por uma constrição do trato vocal, produz sons com turbulência.

Esta frequência fundamental dependerá da dimensão e espessura da glote. Para locutores homens, a gama de vibração das cordas vocais situa-se entre 50 e 250 Hz, enquanto que, para locutores do gênero feminino, essa gama situa-se entre 120 e 300 Hz, podendo chegar até 500 Hz para as crianças [9]. Segundo [9], é comum um locutor apresentar uma variação que pode atingir em fala natural uma oitava, isto é, 80 a 160 Hz para um locutor masculino, podendo atingir duas oitavas no caso de fala forçada ou cantada. Variações mais acentuadas requerem um esforço físico considerável. A Figura 2.9 apresenta a forma de onda de um segmento vozeado */e/* e de um segmento não-vozeado */s/*, pronunciados por um locutor do gênero masculino e por um locutor do gênero feminino.

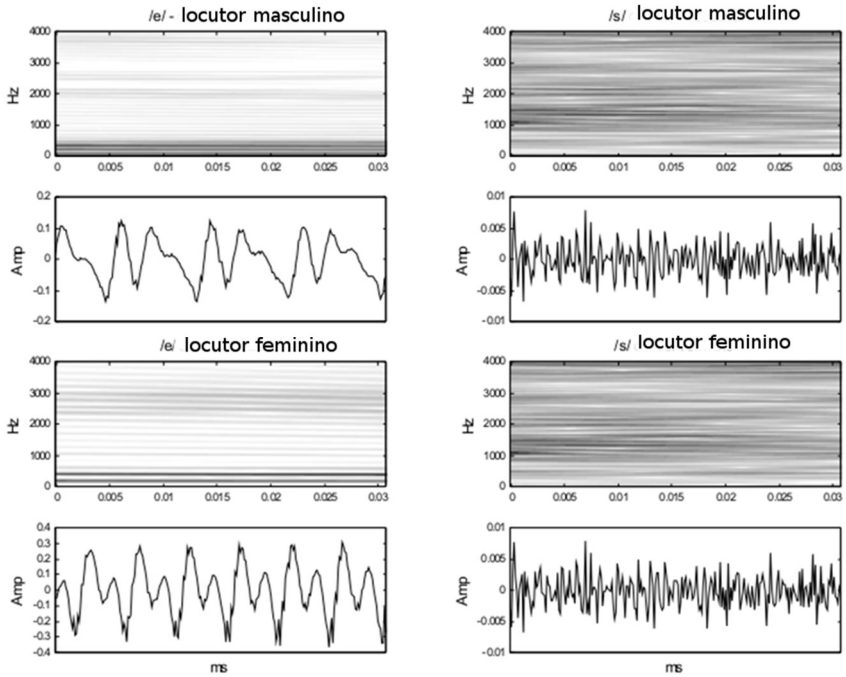

Figura 2.9 - Comparação dos sinais de fala masculino e feminino.
Fonte: [9]

É possível notar pelos espectrogramas mostrados na Figura 2.9 que as zonas vozeadas tem uma característica passa-baixas e as zonas não-vozeadas apresentam maior energia nas altas frequências.

2.5 O SISTEMA AUDITIVO HUMANO

O sistema auditivo humano pode ser dividido em duas partes: o sistema auditivo periférico e o sistema central.

O sistema auditivo periférico permite a transmissão de uma onda sonora desde o pavilhão auricular até os primeiros neurônios do nervo auditivo, sendo composto por três partes:

- O ouvido externo irá captar o som. Ele compreende o pavilhão auricular, ocasionalmente conhecido como “orelha” (única parte visível do sistema), o canal auditivo externo e a membrana do tímpano [11]. O tímpano é um pedaço de pele fina, em forma de cone e tem aproximadamente 10 mm de largura e sua função é converter o som em vibrações e as transmitir ao ouvido médio. O funcionamento do tímpano é

semelhante ao de um diafragma em um microfone. As vibrações da onda sonora o empurram para frente e para trás. Ondas sonoras de maior frequência acabam movendo a membrana mais rapidamente e sons mais graves movem a membrana por uma distância maior, uma vez que têm maior período e frequências mais baixas. [12].

- Em seguida encontramos o ouvido médio, que compreende um espaço muito pequeno onde se situam três ossículos: a bigorna, o martelo e o estribo. As vibrações provenientes do tímpano são amplificadas no ouvido médio aproximadamente vinte e duas vezes até serem transmitidas ao ouvido interno. O ouvido médio é conectado à garganta através da trompa de Eustáquio. Assim, a diferença entre a pressão interna e externa no tímpano é nula, visto que o ar atmosférico entra pela boca e pelo ouvido e o equilíbrio de pressão faz com que o tímpano se mova para frente e para trás de forma livre [12].
- Já o ouvido interno é responsável por “codificar” a mensagem e a preparar para enviar ao cérebro. Compreende uma estrutura chamada cóclea, onde é possível encontrar mais de vinte mil células ciliadas (internas e externas) [11]. As externas irão amplificar ou inibir as vibrações que chegam, enquanto as internas convertem as vibrações em pulsos que serão enviados ao cérebro através das fibras do nervo auditivo e o cérebro lhes atribuirá um significado e determinará uma sensação auditiva. Ligado à cóclea está o vestíbulo, também responsável pelo equilíbrio, e o conjunto dessas duas partes é chamado labirinto. “Esse sistema é constituído por uma estrutura óssea dentro da qual se encontra um sistema de tubos membranosos cheios de líquido, cujo movimento – provocado por movimentos da cabeça – estimula células ciliadas que enviam impulsos nervosos ao cérebro ou diretamente a centros que controlam o movimento dos olhos ou os músculos que mantêm o corpo numa posição de equilíbrio” [13].

A Figura 2.10 exibe uma representação gráfica dos componentes do sistema auditivo periférico:

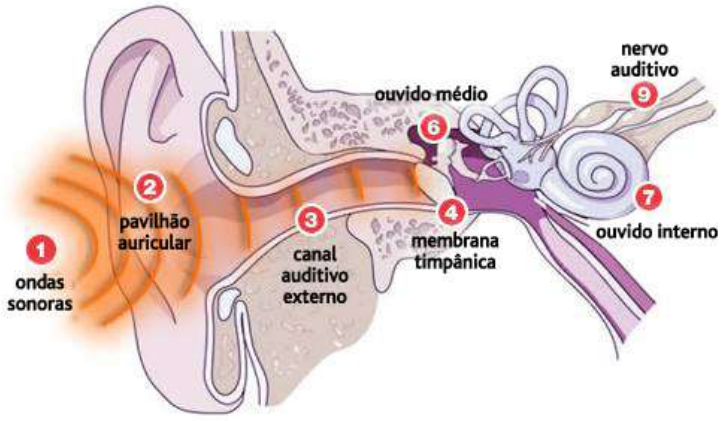

Figura 2.10 - Representação ilustrativa do ouvido humano e seus componentes.
Fonte: [11]

O sistema auditivo central é responsável pela comunicação entre o sistema auditivo periférico e o sistema nervoso central. Ele é composto pelas vias auditivas no nível do tronco cerebral e do córtex auditivo [14]. O sistema periférico se comunica com o sistema central através das fibras nervosas (neurônios) aferentes que partem da cóclea até o córtex auditivo e das fibras nervosas eferentes que fazem o caminho inverso. A informação disponível no nervo auditivo chega ao cérebro com diferentes atrasos e esses vários atrasos são responsáveis por um tratamento adicional da mensagem auditiva. Além de transmitir a informação sonora, as diferentes vias auditivas fornecem a informação relativa à frequência do som (e altura), à intensidade e à posição espacial da fonte emissora.

2.5.1 Funcionamento do sistema auditivo humano

O som aparece em decorrência de uma perturbação aérea, resultante de rápidas flutuações de pressão se propagando sobre a forma de uma onda [15]. Os sons do ambiente não são definidos por uma só frequência, mas por um conjunto de frequências que compõe um espectro. A audição humana está limitada a certos níveis de frequência, enquanto outras espécies podem ouvir frequências muito mais elevadas ou até menos elevadas que nossa espécie.

Podemos dizer que nosso sistema auditivo se comporta como um prisma acústico, fazendo uma separação de frequências sonoras em intervalos frequenciais.

A cóclea é constituída de um longo túnel enrolado sobre si mesmo. O estribo coloca em vibração a primeira extremidade desse túnel. As frequências sonoras se propagam dentro da cóclea e um efeito de ressonância ocorre para que cada frequência sonora vibre uma determinada parte da membrana.

Quando o som percebido for de alta frequência, se propagará apenas em um pequeno trecho da membrana e o movimento de vibração será mínimo por toda a membrana. Um som de baixa frequência propagar-se-á ao longo de quase toda a membrana [16]. O cérebro então distinguirá entre graves, médios e agudos de acordo com a porção vibrante da membrana.

A Figura 2.11 esquematiza a cóclea e as frequências de som vinculadas a cada região:

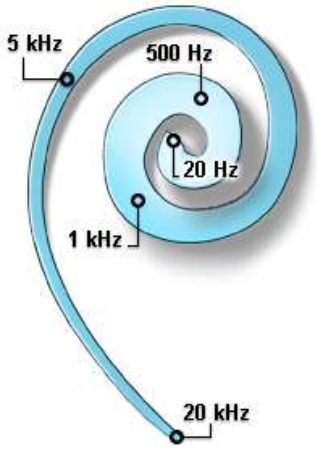

Figura 2.11 - Esquematização da cóclea e níveis de frequência perceptíveis.
Fonte: [17]

Diferentes zonas são especializadas na conversão de frequências: as frequências elevadas (agudas) são captadas no início da cóclea e as frequências mais baixas (graves) são captadas no final da espiral [17].

A intensidade de um som é determinada pela intensidade de movimento das fibras internas à membrana. Quanto maior o deslocamento, para frente e para trás, mais intensamente as células sensitivas são estimuladas e mais estímulos serão transmitidos ao cérebro para indicar a magnitude dessa intensidade [16]. O som mais fraco que o ouvido humano pode perceber corresponde a uma vibração de 20 mPa, sendo que consegue suportar pressões várias vezes superiores [17]. O nível de pressão acústica é dado pela equação abaixo:

$$L_p = 20 \cdot \log_{10}\left(\frac{p}{p_0}\right)_{com\ p_0 = 20mPa}$$

O sistema auditivo humano apresenta um nível mínimo para a audição de um som (limiar da audição, L_p = 0 dB) e um nível máximo (limiar da dor, normalmente em torno de L_p = 120 dB), estes limites podem variar em função da frequência. Valores próximos do limiar da dor podem causar danos permanentes ao sistema auditivo. A 140 dB, o tímpano pode ser destruído [17].

A Figura 2.12 mostra os limiares descritos acima para diferentes níveis de frequências:

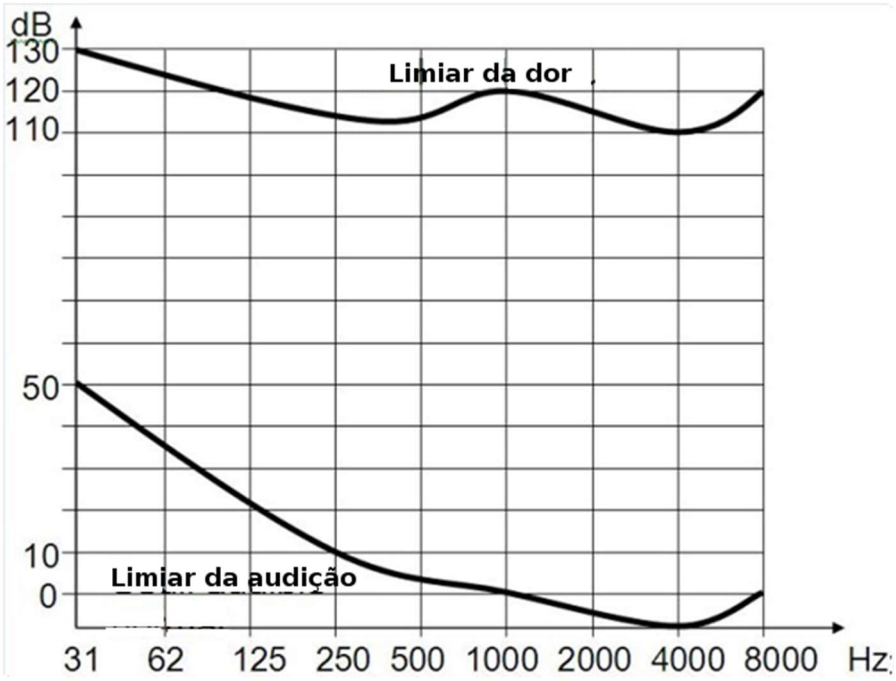

Figura 2.12 - Faixa de amplitudes que o ouvido humano consegue suportar.
Fonte: [15]

A Figura 2.13 mostra as variações de sons que podem ser ouvidas de acordo com cada idade em função das frequências que as definem:

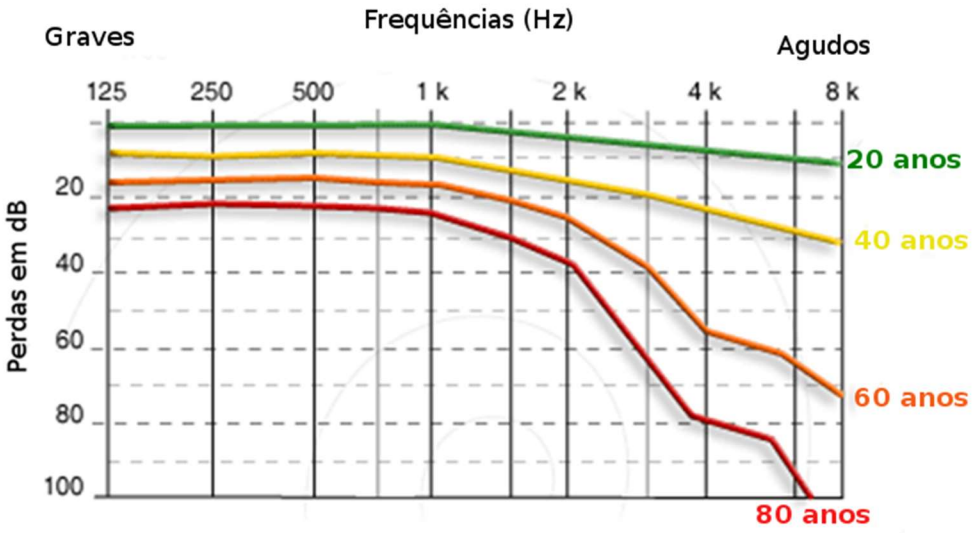

Figura 2.13 - Capacidade humana de reconhecer frequências.
Fonte: [16]

Por fim, a Figura 2.14 define os intervalos frequenciais que o ouvido humano consegue captar:

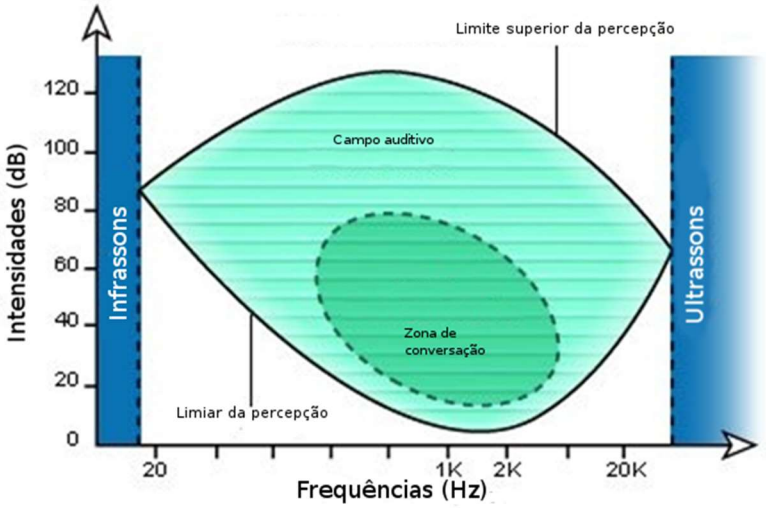

Figura 2.14 - Faixa de frequências onde opera a audição humana.
Fonte: [16]

2.6 FERRAMENTAS UTILIZADAS PELO HTK

De agora em diante, tratar-se-á um pouco a respeito de algumas tecnologias utilizadas na análise de sinais de áudio, assim como na codificação desses sinais em vetores que serão utilizados para o reconhecimento de fala.

O *software* HTK recebe arquivos de áudio como entrada. Embora muitos formatos sejam possíveis de se utilizar, como "*.wav*", "*.ogg*", "*.mp3*", será necessária uma parametrização para que o HTK possa interpretar esses arquivos não como áudio, mas como coeficientes relacionados a dados estatísticos.

Para realizar tal conversão, o HTK disponibiliza uma ferramenta que transforma os arquivos de áudio em vetores de parâmetros *MFCC* (*Mel-frequency Cepstral Coeficients*).

Para entender estes conceitos, inicialmente será apresentada uma descrição da escala frequencial Mel na seção 2.6, seguida de uma descrição de *Cepstrum* na seção 2.7, para então entrar no mérito dos *MFCC*s também na seção 2.7.

Um outro parâmetro muito importante na conversão destes dados é o tamanho da janela que o HTK irá utilizar para esta tarefa. O conceito de enjanelamento será discutido na seção 2.8.

Durante a obtenção destes parâmetros utilizados no reconhecimento de fala, é muito comum o uso de um filtro pré-ênfase no processo. Uma descrição deste procedimento está disponível na seção 2.9.

Para finalizar o capítulo, uma abordagem das técnicas atuais de processamento de fala será descrita na seção 2.10.

2.7 A ESCALA MEL

A escala MEL, chamada assim por Stevens, Volkmann e Newman em 1937 [18] é uma escala físico-acústica de alturas de sons, na percepção de sua identificação entre graves e agudos. O ponto de referência entre essa escala e a escala tradicional de frequências é definido atribuindo-se uma altura percentual de 1000 mels para um tom de 1000 Hz, 40 dB acima do limiar da audição. Acima de 500 Hz, intervalos cada

vez maiores são necessários para que, na percepção dos ouvintes, sejam produzidos incrementos de altura iguais. Como resultado, quatro oitavas da escala Hz acima de 500 Hz se fazem necessárias para compreender aproximadamente duas oitavas da escala MEL. Uma forma de converter a frequência em Hz para Mel é dada pela seguinte equação [19]:

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \quad (1)$$

A Figura 2.15 dá a correspondência entre as duas escalas de frequência: no eixo das abscissas a escala habitual em Hertz e no eixo das ordenadas a escala Mel.

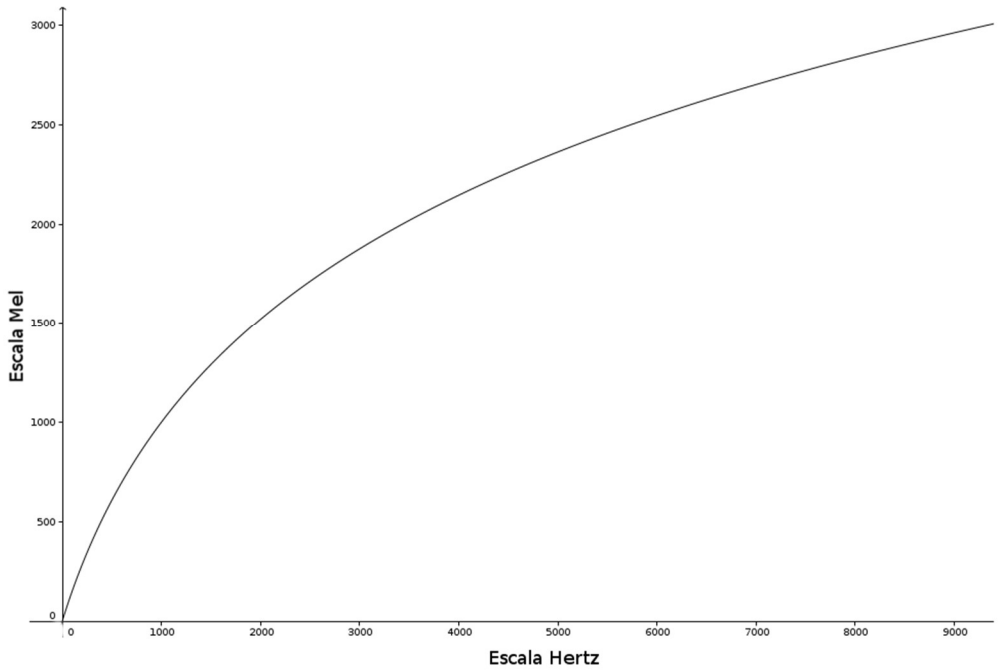

Figura 2.15 - Correspondência entre as escalas de frequências.
Fonte: Produção própria, Geogebra.

2.8 MEL-FREQUENCY CEPSTRAL COEFFICIENTS (MFCC)

Um *Cepstrum* é definido como sendo a transformada inversa de Fourier de uma transformada de Fourier de escala logarítmica:

$$c[n] = \mathcal{F}^{-1}\{\log_{10}|X[k]|\} \quad (2)$$

Para um trecho de fala com enjanelamento *y[n]*, o *Cepstrum* é dado pela equação (3):

$$c[n] = \sum_{n=0}^{N-1} \log_{10}\left(\left|\sum_{n=0}^{N-1} x[n]e^{-j\cdot\frac{2\pi}{N}\cdot kn}\right|\right) e^{j\cdot\frac{2\pi}{N}\cdot kn} \quad (3)$$

O diagrama de blocos mostrado na Figura 2.16 detalha as etapas de cálculo do *Cepstrum*:

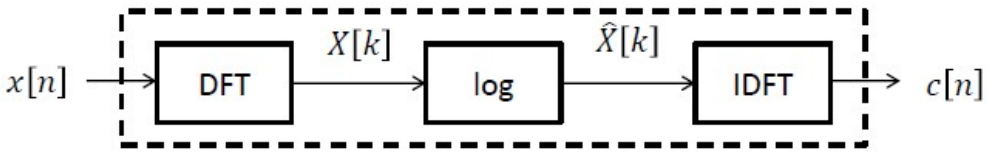

Figura 2.16 - Procedimento para se gerar um *cepstrum*.
Fonte: [20]

Em se tratando de um sinal *x[n]*, a magnitude do espectro de frequências *|F{x[n]}|* contém harmônicos em intervalos de frequência eventualmente espaçados, os quais diminuem de magnitude com o aumento da frequência. Calculando-se o espectro em escala logarítmica é possível comprimir a escala e diminuir as diferenças de amplitude entre harmônicos.

Agora, imaginando que o espectro logarítmico fosse uma forma de onda, nesse caso, deveríamos descrevê-la como quasi-periódica com algumas formas de amplitude modulada. Aplicando-se a Transformada de Fourier, espera-se encontrar um largo “pico” nos arredores do período do sinal e alguns componentes de baixas frequências devido à modulação em amplitude. Um filtro simples permitiria separar as duas componentes do sinal.

O sinal de fala pode ser modelado como sendo a convolução da fonte glotal u[n], o trato vocal v[n] e a radiação *r[n]* [20]:

$$y[n] = u[n] \otimes v[n] \otimes r[n] \quad (4)$$

Como esses sinais são convoluídos, torna-se complicado separá-los em uma representação temporal.

Por conveniência, é possível combinar $v'[n] = v[n] \otimes r[n]$ e, a partir disso,

$$y[n] = u[n] \otimes v'[n] \quad (5)$$

Tomando a transformada de Fourier do sinal *y[n]* teremos:

$$Y(e^{j\omega}) = U(e^{j\omega}).V'(e^{j\omega}) \quad (6)$$

Aplicando-se logaritmo nos dois lados da equação (6):

$$\log_{10}\left(Y(e^{j\omega})\right) = \log_{10}\left(U(e^{j\omega})\right) + \log_{10}\left(V'(e^{j\omega})\right) \quad (7)$$

Representando que fonte e filtro estão adicionados. Tomando a transformada inversa de Fourier na equação (7), retornamos ao domínio temporal:

$$c[n] = c_u[n] + c_v[n] \quad (8)$$

Quando se aplica a transformada de Fourier de um sinal e em seguida a sua transformada inversa, o que se espera é obter novamente o sinal original, mas quando se calcula a transformada inversa do espectro logarítmico, se retira a informação do espectro de magnitude, ignorando o espectro de fase. O *Cepstrum* é importante visto que consegue separar fonte (sinal glotal) e filtro (modulação na cavidade supraglotal). Se há interesse na excitação da glote, mantém os altos coeficientes; se o interesse for o trato vocal, separam-se então os coeficientes baixos. Os coeficientes cepstrais são uma representação muito compacta do envelope espectral e como são não-correlacionados acabam sendo úteis em modelos [20].

A Figura 2.17 mostra uma comparação entre o espectro de frequências de um sinal x[n], o mesmo espectro em uma escala logarítmica e a transformada inversa do espectro logarítmico (*Cepstrum*).

O eixo das abcissas no gráfico do *Cepstrum* é representado como "*Quefrequency*" e a únidade é *ms*, esse foi o nome dado pelos desenvolvedores da técnica para representar que o *Cepstrum,* na verdade, está no domínio do tempo, mas é analisado pensando no domínio frequencial.

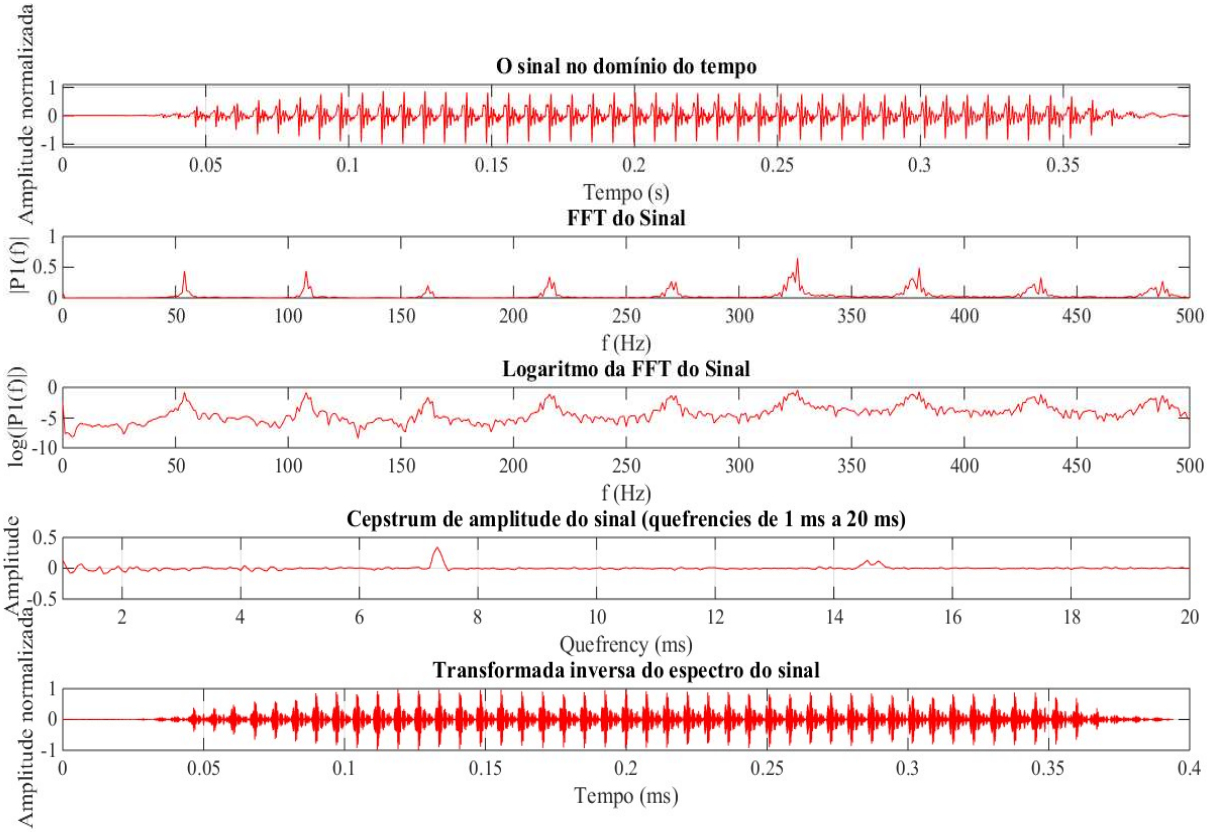

Figura 2.17 – Comparação entre espectros de frequência, sinal no tempo e seu respectivo *Cepstrum*. Fonte: Produção própria, *Matlab*.

Em processamento de som, MFC (*Mel-Frequency Cepstrum*) é uma representação de intervalos curtos de tempo do espectro de potência de um determinado sinal sonoro, baseado na transformada linear do cosseno do espectro de potências logarítmico. MFCC (*Mel-Frequency Cepstral Coefficients*) são, na verdade, coeficientes que, coletivamente, formam uma MFC. Eles são derivados de uma forma de representação *Cepstral* do trecho de áudio (um espectro não-linear do espectro). Seu cálculo será visto na seção seguinte.

A diferença entre o *Cepstrum* e MFC é que no MFC as bandas de frequência são igualmente espaçadas na escala MEL, o que aproxima a resposta do sistema auditivo humano de forma mais parecida do que a escala frequencial espaçada linearmente usada no *Cepstrum*.

Os coeficientes MFCC são, provavelmente, a parametrização mais comum em reconhecimento de fala, combinando as vantagens da análise do *Cepstrum* com a escala de frequência baseada nas bandas críticas.

2.8.1 Cálculo do MFCC

Primeiramente, o sinal de fala é analisado usando a Transformada de Fourier de curta duração. Então, os valores da transformada de Fourier são agrupados em bandas críticas e ponderados de acordo com a função de ponderação triangular mostrada abaixo na Figura 2.18:

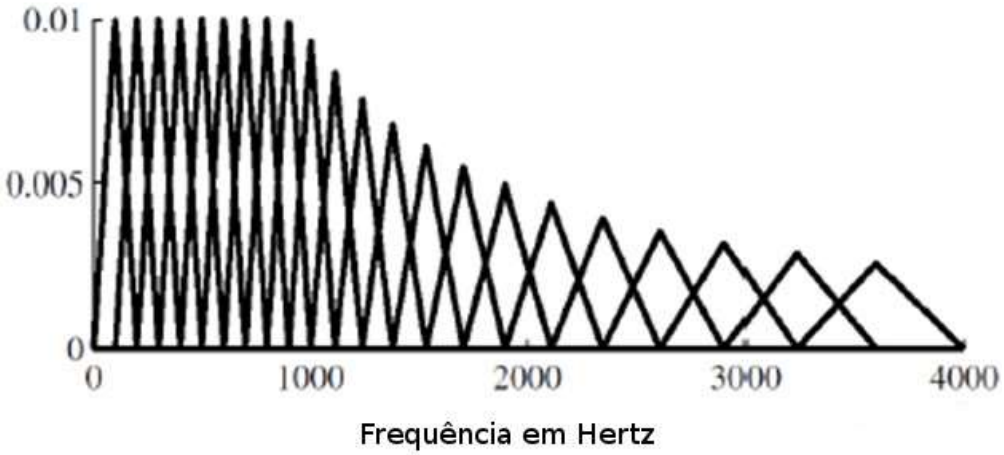

Figura 2.18 - Função de ponderação triangular.
Fonte: [21]

O espectro de frequência MEL analisado no instante n é definido pela equação (8):

$$MF[r] = \frac{1}{A_r} \sum_{k=L_r}^{U_r} |V_r[k] \cdot X(n,k)|, \tag{9}$$

em que $V_r[k]$ é a função de ponderação triangular do r-ésimo filtro variando desde o índice da Transformada de Fourier L_r até U_r e:

$$A_r = \sum_{k=L_r}^{U_r} |V_r[k]|^2, \tag{10}$$

o que dá um fator de normalização para o r-ésimo filtro, de modo que um achatamento perfeito do espectro de Fourier fornece também um espectro Mel achatado.

Para cada quadro m analisado do sinal, uma transformada discreta do cosseno é calculada da magnitude logarítmica das saídas do filtro para obter os coeficientes MFCC:

$$MFCC[m] = \frac{1}{R}\sum_{r=1}^{R} log(MF[r]).cos\left[\frac{2\pi}{R}\cdot\left(r + \frac{1}{2}\right)\cdot m\right] \tag{11}$$

Tipicamente, *MFCC[m]* é avaliada para um número de coeficientes N_{MFCC} menor que o número de filtros *R*. Por exemplo, para F_s = 8 kHz, valores típicos são N_{MFCC} = 13 e *R* = 22.

A Figura 2.19, mostrada na sequência, representa uma comparação entre três tipos de parametrização. A curva traçada em cinza representa a transformada de Fourier de determinado sinal, logo, mostra o espectro do respectivo sinal no domínio frequencial. As amplitudes são representadas na escala logarítmica, em dB.

O traçado vermelho mostra a aproximação do envelope espectral da curva cinza utilizando-se a parametrização MFCC. Percebe-se que no total há 22 pontos nesta representação, o que indica o número de filtros R e não o N_{MFCC}.

A curva em verde representa a aproximação do envelope espectral para uma parametrização LPC de 12 coeficientes(p). Esta parametrização é discutida no apêndice C. Neste trabalho, analisar-se-á as diferenças entre MFCC e LPC no capítulo 4.

A curva tracejada em azul é a aproximação utilizando-se parametrização homomórfica com 13 coeficientes (nco). Este tipo de aproximação não é analisada durante este trabalho.

Comparando-se as curvas definidas na Figura 2.19, nota-se que a aproximação LPC, descreve muito bem os picos de amplitude no espectro, assim como a aproximação MFCC. Mas ambas não representam tão bem o envelope

espectral quando se há uma variação busca na amplitude (frequência entre 3500 e 4000 Hz). A aproximação homomórfica parece captar uma média da amplitude espectral e consegue perceber melhor essas variações acentuadas.

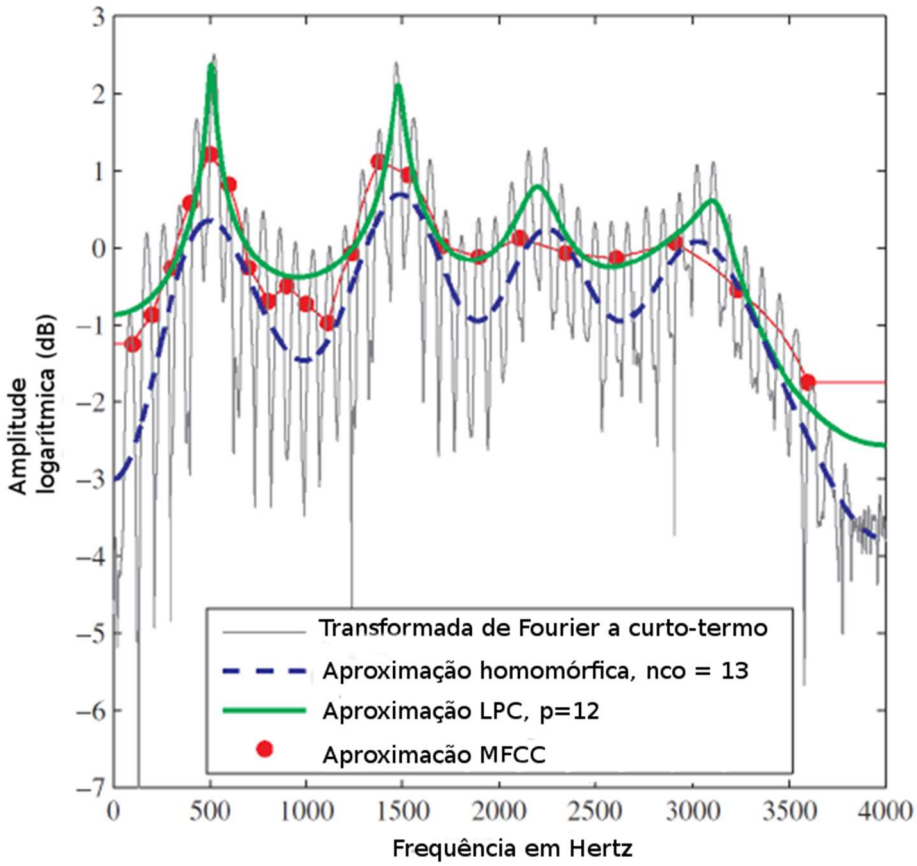

Figura 2.19 - Comparação entre os métodos de parametrização do sinal.
Fonte: [20]

2.9 ENJANELAMENTO

A Transformada Rápida de Fourier (*FFT*) é utilizada para se obter a componente frequencial de determinado sinal, mas, para isso, a análise é feita sob um conjunto discreto de dados em um espectro contínuo igual a um período de um sinal periódico.

No entanto, é comum que o sinal adquirido não seja periódico, resultando em uma forma de onda truncada, com características diferentes do sinal original contínuo no tempo, o que pode introduzir descontinuidades no sinal medido entre o final de uma aquisição e o início da próxima.

Essas descontinuidades são mostradas na *FFT* como componentes harmônicas, não presentes no sinal original. Essas frequências podem ser muito mais altas que a frequência de Nyquist, distorcendo o espectro e dispersando energia de uma frequência para outras frequências.

O enjanelamento reduz a amplitude das descontinuidades nas bordas de cada sequência finita adquirida pelo amostrador. O enjanelamento consiste na multiplicação do sinal amostrado por uma janela de comprimento finito que varia de maneira uniforme e gradual até se anular nas bordas. Com isso, os pontos extremos da forma de onda se encontram e a forma de onda será contínua, sem transições abruptas [22]. A Figura 2.20 compara a FFT de um sinal não-periódico não utilizando enjanelamento (direita) com a FFT de um sinal não-periódico utilizando enjanelamento (esquerda).

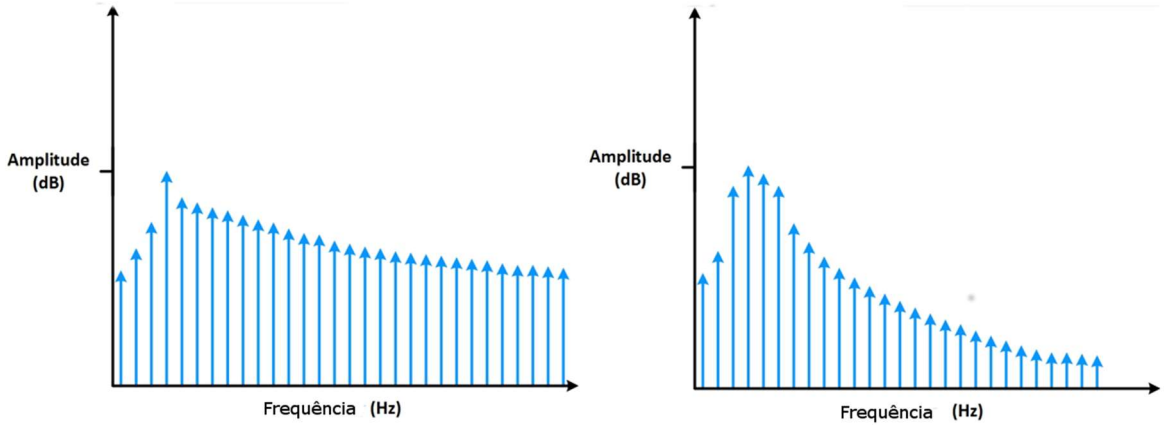

Figura 2.20 - FFT de um sinal não-periódico utilizando (esquerda) e não-utilizando (direita) enjanelamento.
Fonte: [22]

A largura do lóbulo principal irá caracterizar sua resolução em frequência, enquanto a amplitude máxima das oscilações no lóbulo secundário caracterizará a dinâmica do espectro útil.

A janela mais simples é a retangular, mas outras janelas amplamente utilizadas em tratamento digital de sinais incluem a janela de Bartlett, janela de Von Hann, de Hamming, triangular, etc. Dependendo da dinâmica desejada, cada uma dessas janelas pode se tornar útil.

O sinal de fala, ao ser transformado em coeficientes e analisado pelo *software* HTK, passa por um enjanelamento para seu tratamento. A Figura 2.21 compara a resposta de diversos tipos de janela em função da frequência. Nota-se que, dependendo da janela, a banda de transição e os lobos secundários podem variar significativamente.

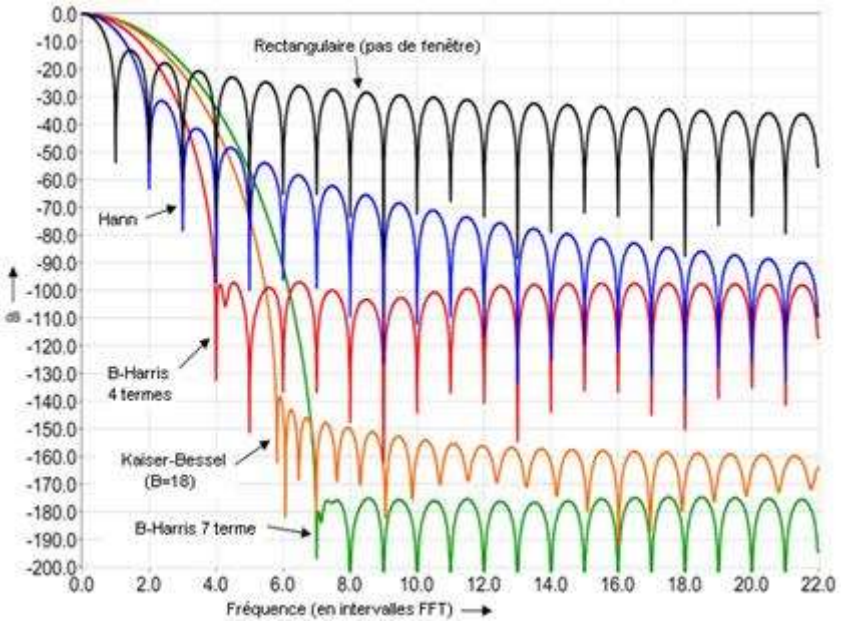

Figura 2.21 - Comparação entre diversas janelas.

Fonte: http://www.ni.com/white-paper/4844/fr/
Acesso em 30 nov. 2016, 19h57.

2.10 FILTRO PRÉ-ÊNFASE

A pré-ênfase é um método de tratamento utilizado em processamento de sinais de áudio eletrônicos, que aumenta (em determinada banda de frequências) a amplitude de certas frequências (geralmente altas), respeitando a magnitude de outras (normalmente mais baixas), para aumentar a taxa de sinal-ruído, minimizando os efeitos adversos de distorção na atenuação ou saturação. A operação contrária é chamada de-ênfase e o sistema todo é chamado de ênfase [23].

Esta operação é obtida geralmente através de filtros calibrados. Sua resposta em frequência é decidida através de constantes de tempo especiais e ocasionalmente é utilizada em telecomunicações, gravação de sinais de áudio, transmissões de rádio FM e na visualização dos espectrogramas de sinais de fala.

No caso deste trabalho, o filtro pré-ênfase terá como função amplificar as frequências altas, em relação às baixas frequências, no caso do tratamento de sinais de fala. O ruído, que normalmente se concentra em componentes de alta frequência, será atenuado no processo de de-ênfase. O sinal de origem é tratado através de um filtro digital passa-altas. A transformada Z desse filtro, em geral, é:

$H(z) = 1 - a \cdot z^{-1}$, onde a pode variar entre 0,9 e 1.

O objetivo da pré-ênfase é compensar parte de alta frequência apagada durante o mecanismo de produção de som humano, assim como amplificar a importância das formantes de alta frequência [24]. Formantes são picos de energia em determinada região espectral. Elas são importantes para caracterizar timbres.

Um exemplo deste sistema é a curva de equalização nos discos de vinil de 33 e 45 RPM. Outro caso é no sistema *Dolby* de redução de ruído usado em fitas magnéticas [23].

2.11 TÉCNICAS ATUAIS DE RECONHECIMENTO DE FALA

Atualmente, três tecnologias se destacam nas pesquisas de reconhecimento de fala; são elas: Robustez, *Cocktail Party* e *DNN*s (*Deep Neural Networks* ou redes neurais profundas), as quais serão discutidas nos tópicos a seguir.

2.11.1 Robustez

A robustez de um sistema de reconhecimento de fala trata de desenvolver algoritmos e técnicas de tratamento de sinal, que implique em criar certa imunidade a ruídos, barulhos e sons gerais, no plano de fundo do locutor, que não fazem parte da locução e que podem vir a interferir na qualidade do reconhecimento.

Existem certos algoritmos, como o proposto por [25], que adapta os modelos ocultos de Markov para um auto-aprendizado, o que torna o reconhecimento de fala robusto. Através deste algoritmo de adaptação, modelos bem treinados podem ser adaptados para novas condições de elocução ou novos locutores. O auto-aprendizado

é tal que, durante o reconhecimento, o modelo pode ser aprimorado através de novas informações obtidas. Ambos os procedimentos equivalem a aumentar o conjunto de treinamento.

Já [26] cita que o sistema auditivo humano é altamente robusto contra ruídos de fundo e variabilidade de canais, quando comparado com sistemas automatizados.

2.11.2 Cocktail Party

Essa tecnologia diz respeito ao reconhecimento de fala de um locutor enquanto outras pessoas conversam ao redor, ou seja, um reconhecimento dependente do locutor, com interferência de outros locutores nas proximidades.

Segundo [27], a captura de uma cena em um espaço e sua reprodução requer a extração direcional das informações a partir dos sinais captados. O autor apresenta a proposta de usar múltiplos microfones para a captação. Então o som poderia ser processado usando, tanto a informação direcional, quanto a informação da distância e seletivamente reproduzir trechos específicos da cena de som capturado. Para [28], é possível, através de uma aproximação de aprimoramento de fala por um procedimento iterativo, suprimir os sinais de fala indesejados e ruído em um ambiente ruidoso ou com pessoas conversando. A ideia chave na aproximação proposta é que a correlação da diferença de fase baseada no critério DPCF (*difference phase-correlation-based function,* ou baseado em correlação de fase) é introduzida para trabalhar como controle de convergência em um filtro de Wiener iterativo modificado (este filtro é focado na redução de ruído). O componente de interferência é então atenuado continuamente até a ocorrência da convergência.

2.11.3 Redes neurais profundas

Estes estudos permitem a melhoria do desempenho em sistemas de reconhecimento de fala. Um exemplo é a tecnologia utilizada pelo *Google* para identificação de palavras e comandos de fala em telefones portáteis e computadores. De acordo com [29], muitos pesquisadores tentaram desenvolver sistemas de reconhecimento de fala nas décadas passadas. No entanto, o aproveitamento desses

desenvolvimentos sempre esteve defasado quando comparado com o sistema auditivo humano. Métodos adaptando os parâmetros do modelo acústico, assim como as ferramentas de entrada dos sistemas de reconhecimento de fala, para o aumento de seu desempenho, foram propostos para a solução desse problema.

Em um primeiro método, vetores de fala de cada locutor foram adaptados não linearmente depois de algumas iterações, utilizando os algoritmos denominados de direita-esquerda. Já em um segundo método, o sistema de reconhecimento é modificado de forma a permitir adaptar dinamicamente as variabilidades dos locutores. Esta adaptação ocorre em tempo real. Alguns experimentos demonstraram que esses métodos conseguem um melhoramento de aproximadamente 2 a 6 % nas taxas de acerto.

A partir de [30] é proposta uma rede neural profunda (DNN) para extrair informações articulatórias e explorar diferentes formas de utilizar esta informação em sistemas de reconhecimento de fala contínua. Esta DNN foi treinada para estimar trajetos articulatórios das entradas nos testes, onde os dados de treinamento eram palavras em inglês sintéticas por um sistema de produção de fala. Os resultados demonstraram que esses recursos articulatórios aumentaram o desempenho do reconhecimento quando combinados com outros recursos cepstrais.

3. MODELOS OCULTOS DE MARKOV

Existem razões interessantes pelas quais se podem utilizar modelos de sinais como, por exemplo, facilitar a descrição deste sinal teoricamente, buscando-se analisar padrões em um sistema de processamento de sinal, que seria utilizado para prover uma determinada saída desejada. Segundo [31], uma aplicação prática seria a atenuação de ruído em sinais de fala corrompidos, onde utilizando um modelo preciso para o sinal, o ruído poderia ser removido de forma otimizada. Outra motivação para a utilização de modelos de sinais seriam simulações nas quais não é possível ter acesso à fonte que produz o sinal em questão. Mas a principal razão para utilizar modelos de sinais é que eles funcionam extremamente bem na prática e permitem realizar sistemas práticos importantes como sistemas de predição, sistemas de reconhecimento, identificação de sistemas entre outros de forma bastante eficiente.

A análise *LPC-Cepstral*, uma variante do MFCC que será testada durante o processo de reconhecimento de fala, será abordada no apêndice C. Tanto MFCC quanto *LPC-Cepstral* fornecem os vetores de dados observados e estão vinculados diretamente ao processo de reconhecimento de fala utilizado aqui.

De acordo com [31], poderíamos dividir os sinais em dois grupos: determinísticos e estocásticos. O primeiro caso é caracterizado por sinais em que temos acesso a parâmetros de controle deste, isto é, existe a possibilidade de representarmos matematicamente o sinal através de sua amplitude, defasagem, frequência, período, constante de tempo, etc. Exemplos de sinais determinísticos são senoides, somas de exponenciais, dentre outros. Já o segundo grupo é caracterizado por modelos estatísticos que tentam caracterizar somente as propriedades incertas de um sinal. Processos gaussianos, processos de Poisson, processos markovianos e processos ocultos de Markov, entre outros, podem ser classificados como estocásticos, sendo que os dois últimos serão discutidos no presente capítulo. Mas, para representar tais modelos, torna-se necessário assumir que o sinal pode ser caracterizado com processos de parâmetros aleatórios e tais parâmetros podem ser estimados de uma forma bem definida.

Existem três problemas fundamentais para o desenvolvimento de um modelo oculto de Markov: a evolução da probabilidade de obtermos uma determinada sequência de observações dada uma HMM específica; a determinação dos estados

do modelo a partir do problema e o ajuste dos parâmetros do modelo para melhor estimar um sinal observado.

3.1 PROCESSOS DISCRETOS DE MARKOV

Será considerado um sistema que possa ser descrito a qualquer momento como sendo participante de um conjunto de estados distintos S_1, S_2, ..., S_N, como ilustrado pela Figura 3.1:

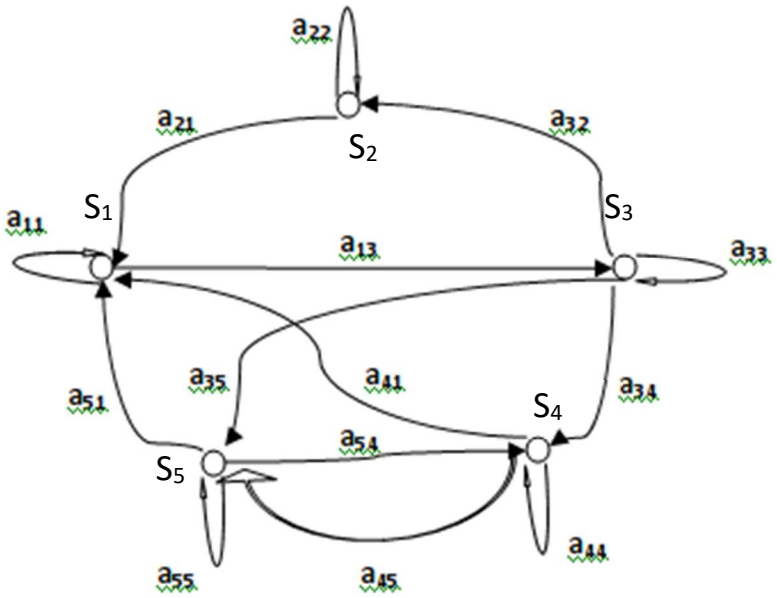

Figura 3.1 – Modelo discreto de Markov com 5 estados.
Fonte:Produção própria, baseado em [31].

A partir de [31] é possível deduzir que o sistema discreto mudará de estado (podendo continuar no mesmo estado) de acordo com um conjunto de probabilidades associadas a cada um destes estados. No caso da Figura 3.1, mostrada acima, existem 5 estados possíveis. Denotaremos instantes de tempo como t = 1, 2, ..., T e o estado atual no tempo t como q_t. Uma descrição probabilística completa do sistema da Figura 3.1 demandaria especificação do estado atual assim como todos os outros estados anteriores. Para o caso específico de uma cadeia de Markov discreta de

primeira ordem, essa descrição probabilística é simplificada para apenas o estado atual e o imediatamente anterior:

$$P\left[q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, \ldots\right] = P\left[q_t = S_j | q_{t-1} = S_i\right] \tag{12}$$

Além disso, apenas serão considerados os processos nos quais o lado direito da equação (12) é independente do tempo, levando assim ao conjunto de probabilidades de transições dadas por:

$$a_{ij} = P\left[q_t = S_j | q_{t-1} = S_i\right], \quad 1 \leq i, j \leq N, \tag{13}$$

com os coeficientes de transição de estados definidos pelas probabilidades:

$$a_{ij} \geq 0 \; e \sum_{j=1}^{N} a_{ij} = 1, \tag{14}$$

uma vez que estes coeficientes respeitam as restrições probabilísticas (elementos positivos cuja soma representa 100 %) . O procedimento descrito pode ser nomeado de modelo de Markov observável, desde que a saída do processo seja o conjunto de estados a cada instante de tempo, onde cada estado corresponde a um evento físico (observável). Para ilustrar melhor este caso, é possível imaginar um exemplo no qual um computador acoplado a um amplificador e alto-falante emite notas musicais na escala de dó maior (Dó, Ré, Mi, Fá, Sol, Lá e Si) em um mesmo timbre, mas aleatoriamente, de forma que haja certa probabilidade de transição entre as notas, estas sendo observadas como sendo:

Estado 1 = Dó, Estado 2 = Ré, Estado 3 = Mi, Estado 4 = Fá, Estado 5 = Sol, Estado 6 = Lá e Estado 7 = Si.

Suponha que a matriz de transição de probabilidades seja definida como:

$$A = a_{ij} = \begin{array}{c} \\ Dó(1) \\ Ré(2) \\ Mi(3) \\ Fá(4) \\ Sol(5) \\ Lá(6) \\ Si(7) \end{array} \begin{array}{c} \begin{array}{ccccccc} Dó(1) & Ré(2) & Mi(3) & Fá(4) & Sol(5) & Lá(6) & Si(7) \end{array} \\ \begin{bmatrix} 0,3 & 0,05 & 0,22 & 0,13 & 0,03 & 0,1 & 0,17 \\ 0,04 & 0,02 & 0,37 & 0,12 & 0,21 & 0,14 & 0,1 \\ 0,25 & 0,13 & 0,07 & 0,05 & 0,17 & 0,3 & 0,03 \\ 0,09 & 0,18 & 0,14 & 0,13 & 0,29 & 0,16 & 0,01 \\ 0,15 & 0,07 & 0,12 & 0,34 & 0,16 & 0,11 & 0,05 \\ 0,11 & 0,12 & 0,09 & 0,24 & 0,13 & 0,09 & 0,22 \\ 0,21 & 0,01 & 0,32 & 0,13 & 0,04 & 0,11 & 0,18 \end{bmatrix} \end{array} \quad (15)$$

Sabendo que a primeira nota executada (t = 1) é Fá (estado 4), torna-se possível calcular a probabilidade da seguinte sequência de notas a partir de Fá: Sol-Fá-Ré-Dó-Dó-Sol-Mi-Lá. Formalmente: O = { S_4, S_5, S_4, S_2, S_1, S_1, S_5, S_3, S_6} corresponde ao tempo t = 1, 2, ..., 8 e deseja-se determinar a probabilidade de O ocorrer dado o modelo:

$$\begin{aligned} P(O|Modelo) &= P[S_4, S_5, S_4, S_2, S_1, S_1, S_5, S_3, S_6|Modelo] \\ &= P[S_4] \cdot P[S_5|S_4] \cdot P[S_4|S_5] \cdot P[S_2|S_4] \cdot P[S_1|S_2] \cdot P[S_1|S_1] \cdot P[S_5|S_1] \\ &\cdot P[S_3|S_5] \cdot P[S_6|S_3] \end{aligned}$$

$$\begin{aligned} P(O|Modelo) &= \pi_4 \cdot a_{45} \cdot a_{54} \cdot a_{42} \cdot a_{21} \cdot a_{11} \cdot a_{15} \cdot a_{53} \cdot a_{36} \\ &= 1 \cdot 0{,}29 \cdot 0{,}34 \cdot 0{,}18 \cdot 0{,}04 \cdot 0{,}3 \cdot 0{,}03 \cdot 0{,}12 \cdot 0{,}3 = 0{,}230014 \cdot 10^{-6} \end{aligned}$$

Onde π_i = P[$q_1 = S_i$], para $1 \leq i \leq N$. Isto é, π_4 representa a probabilidade do estado 4 ser o estado inicial.

3.2 PROCESSOS OCULTOS DE MARKOV

No exemplo anterior, foi discutido um caso em que se utiliza um modelo discreto de Markov, no entanto, o modelo e o número de estados eram conhecidos. Agora, é possível imaginar outro exemplo, no qual alguém do outro lado de uma cortina, onde não é possível saber qual procedimento estaria sendo usado, começa a lançar uma ou mais moedas e ditar o resultado dos lançamentos de moeda como: cara, cara, coroa, coroa, cara, coroa, cara, cara, coroa, coroa, cara... Essa é a única informação disponível.

Pelo menos três maneiras de modelar esse problema poderiam ser utilizadas:

(a) Uma única moeda estaria sendo lançada:

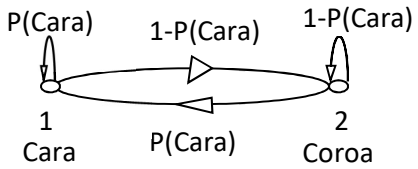

Figura 3.2 – Modelo oculto de Markov com 2 estados: os lados da moeda.
Fonte:Produção própria, baseado em [31].

O = cara, cara, coroa, coroa, cara, coroa, cara, cara, coroa, coroa, cara...

S = 1,1,2,2,1,2,1,1,2,2,1... (estados: Cara e Coroa).

Nesse caso o modelo de Markov é dito observável.

(b) Duas moedas estariam sendo lançadas:

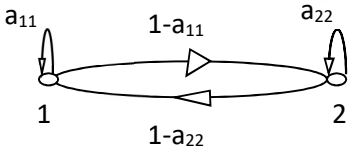

Figura 3.3 – Modelo oculto de Markov com 2 estados: moeda 1 e moeda 2.
Fonte:Produção própria, baseado em [31].

Tabela 2 - Cálculo das probabilidades do modelo de Markov de dois estados mostrado na Figura 3.3.
Fonte: [31]

Moeda 1:	Moeda 2:
P(Cara) = P_1	P(Cara) = P_2
P(Coroa) = 1 – P_1	P(Coroa) = 1 – P_2

O = cara, cara, coroa, coroa, cara, coroa, cara, cara, coroa, coroa, cara...

S = 2,1,1,2,2,2,1,2,2,1,2... (estados: Moeda 1 e Moeda 2).

Nesse caso, cada estado corresponde a uma moeda diferente sendo lançada e cada estado é caracterizado por uma probabilidade de distribuição de caras e coroas e as transições entre estados são caracterizados por uma matriz de transição apresentada na Tabela 2.

(c) Três moedas diferentes estão sendo lançadas:

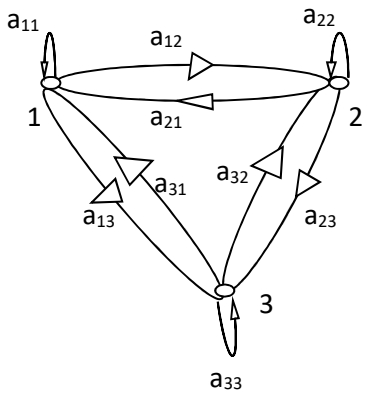

Figura 3.4 – Modelo oculto de Markov com 3 estados: as três moedas.
Fonte:Produção própria, baseado em [31].

Tabela 3 - Cálculo das probabilidades de cada estado no modelo de Markov detalhado na Figura 3.4.
Fonte: [31]

Prob.\Moeda:	1	2	3
P(Cara)	P_1	P_2	P_3
P(Coroa)	$1-P_1$	$1-P_2$	$1-P_3$

O = cara, cara, coroa, coroa, cara, coroa, cara, cara, coroa, coroa, cara...

S = 3,1,2,3,3,1,1,2,3,1,3... (estados: Moedas 1, 2 e 3).

De acordo com [31], o primeiro problema de interesse é decidir a quais estados o modelo corresponde e então decidir o número de estados a serem modelados. Para completar a explicação, um exemplo mais geral é sugerido.

Imaginemos que do outro lado da parede existam N globos rotatórios para sorteio e cada globo possui M bolas, essas numeradas entre 1 e M, dentro de cada globo há bolas numeradas igualmente. A partir de um procedimento aleatório um desses globos é escolhido e então uma bola é tirada deste globo e seu número anotado. Essa bola sorteada retorna então para o globo. Então um novo globo é escolhido e mais uma bola é tirada, o processo se repete. O processo vai gerar uma sequência observável finita de números e o objetivo é modelar uma saída de HMM condizente com essa sequência.

O modelo mais simples de HMM para esse caso é aquele no qual cada estado corresponde a um globo específico e uma probabilidade de cada número ser sorteado

é definida para cada um dos N globos. A escolha do globo é dada pela matriz de transições apresentada na Tabela 4.

Tabela 4 - Matriz de estados para N globos com M bolas.
Fonte: [31]

Globo 1	Globo 2	...	Globo N
P(1) = $b_1(1)$	P(1) = $b_2(1)$	...	P(1) = $b_N(1)$
P(2) = $b_1(2)$	P(2) = $b_2(2)$	...	P(2) = $b_N(2)$
...	...	...	...
P(M)= $b_1(M)$	P(M)= $b_2(M)$	...	P(M)= $b_N(M)$

Para cada globo existe também uma matriz de probabilidades de símbolos observáveis que mostra a probabilidade de cada número sair dependendo do globo sorteado. Após um sorteio que ocorre em uma sala escondida, alguém traz uma sequência de números sorteados nos globos. O desafio é criar uma forma de dizer como foram sorteados esses números e qual é a melhor sequência de globos que explica esse sorteio.

3.2.1 Definições de HMMs

Um modelo oculto de Markov é definido por:

- *N*, o número de estados do modelo: Em geral os estados são conectados de forma que cada estado possa ser complemento de outro estado. $S = \{S_1, S_2, ..., S_N\}$ e o estado no tempo t é definido como q_t;
- *M*, número de símbolos observáveis distintos para cada estado: $V = \{V_1, V_2, ..., V_M\}$;
- Densidade de distribuição de estados $A = \{a_{ij}\}$, onde:

$$a_{ij} = P\left[q_{t+1} = S_j \mid q_t = S_i\right], \qquad 1 \leq i, \mathrm{j} \leq \mathrm{N} \tag{16}$$

No caso em que a_{ij} pode assumir qualquer valor de transição entre estados, temos $a_{ij} > 0$ para todo *i* e *j*. Em outros tipos de HMM é também possível encontrar

valores nulos de a_{ij} para um ou mais pares (*i*, *j*), quando uma determinada transição acaba sendo improvável ou impossível.

- Distribuição de probabilidade de símbolos observáveis no estado *j*, onde:

$$b_j(k) = P[v_k \, em \, t | q_t = S_i], 1 \leq j \leq N, 1 \leq k \leq M \quad (17)$$

A distribuição π que define os valores iniciais dos parâmetros, onde:

$$\pi_i = P[q_1 = S_i], 1 \leq i \leq N \quad (18)$$

Assim, a especificação completa de uma HMM requer a especificação de dois parâmetros do modelo (*M* e *N*), especificação dos símbolos possíveis de serem observados e especificação de três medidas de probabilidade *A*, *B* e *π*. Dados os valores apropriados, a HMM será usada para gerar uma sequência da forma:

$$O = O_1 \cdot O_2 \ldots \cdot O_T$$

Por conveniência é usada a seguinte notação para representar um modelo a partir de seus parâmetros:

$$\lambda = (A, B, \pi)$$

3.2.2 Os três problemas básicos em modelos ocultos de Markov

Dentro do estudo dos modelos ocultos de Markov, há três problemas destacados e a resposta desses problemas é o que permite utilizar essa ferramenta dentro do estudo do reconhecimento de fala. Estes são descritos a seguir.

(a) Dada uma determinada sequência de saída $O = O_1 \cdot O_2 \cdot O_3 \ldots \cdot O_T$, e um modelo $\lambda = (A, B, \pi)$, como calcular $P(O|\lambda)$ a probabilidade de encontrar a sequência O a partir do modelo escolhido?

(b) Dada uma determinada sequência de saída $O = O_1 \cdot O_2 \cdot O_3 \ldots O_T$, e um modelo λ, como escolher uma sequência de estados correspondentes $Q = q_1 q_2 q_3 \ldots q_T$ que possa ser ótimo segundo algum critério (melhor explicável)?

(c) Como ajustar os parâmetros do modelo λ = (*A*, *B,* π) de forma a maximizar $P(O|\lambda)$?

Em se tratando da aplicação em reconhecimento de fala, para cada palavra de um vocabulário de *W* palavras, cria-se o desafio de construir uma HMM de *N* estados. Representa-se o sinal de fala de uma determinada palavra como sequência temporal de vetores espectrais codificados e assume-se que esta codificação é feita de acordo com uma base de dados espectrais com *M* vetores espectrais distintos. Para cada uma das palavras do vocabulário definido existirá uma sequência treinada consistindo de um número de repetições dos trechos definidos no banco de dados de palavras (por um ou mais treinadores). Os estados do modelo serão segmentados a partir das palavras (ou sequências) treinadas [31].

3.2.3 Soluções para os três problemas básicos de HMM

(a) Deseja-se calcular a probabilidade da aparecer na saída do sistema estudado uma sequência $O = O_1 \cdot O_2 \cdot O_3 \ldots \cdot O_T$, a partir do modelo *λ*, isto é, $P(O|\lambda)$. A primeira solução seria enumerar cada sequência de estados possível com comprimento *T* (número de observações). Dada uma sequência de estados fica:

$$Q = q_1 \cdot q_2 \ldots \cdot q_T \; onde \; q_1 \text{ é } o \; estado \; inicial$$

A probabilidade de observar a sequência *O* para os estados da sequência *Q* é:

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(O_t|Q_t,\lambda) \tag{19}$$

Assumindo-se que cada observação é independente das outras temos:

$$P(O|Q,\lambda) = b_{q1}(O_1) \cdot b_{q2}(O_2) \ldots \cdot b_{qT}(O_T)$$

$$P(O|Q,\lambda) \;=\; b_{q1}(O_1)\cdot b_{q2}(O_2)\ldots\cdot b_{qT}(O_T) \tag{20}$$

A probabilidade de aparecer uma sequência de estados *Q* pode ser descrita por:

$$P(Q|\lambda) \;=\; \pi_{q1}\cdot a_{q1q2}\cdot a_{q2q1}\ldots\cdot a_{q(t-1)qt} \tag{21}$$

A probabilidade de *O* e *Q* ocorrerem simultaneamente:

$$\begin{aligned} P(O,Q|\lambda) \;&=\; P(O|Q,\lambda)\cdot\; P(Q|\lambda) \\ &= \sum_{q1q2\ldots qT} \pi_{q1}\cdot b_{q1}(O_1)\cdot a_{q1.q2} \\ &\quad\cdot b_{q2}(O_2)\ldots\cdot a_{q(T-1).qT}\cdot\; b_{qT}(O_T) \end{aligned} \tag{22}$$

Seriam necessárias aproximadamente $(2T\text{-}1)\cdot N^T$ multiplicações e N^T-1 adições para resolver este problema, um número extremamente alto de se imaginar, onde:

$$N = n\acute{u}mero\ de\ estados;$$
$$T = n\acute{u}mero\ de\ observáveis.$$

Felizmente existe um processo mais eficiente que é chamado de *forward-backward*, esse é o algoritmo que responde ao primeiro problema.

Define-se $\alpha_t(i)$ como segue:

$$\alpha_t(i) \;=\; P(O_1\cdot O_2\ldots\cdot O_t, q_t \;=\; S_i|\lambda), \tag{23}$$

isto é, a probabilidade de se observar uma sequência parcial $O_1\cdot O_2\ldots\cdot O_t$ (até o instante t) e estado S_i, dado o modelo λ.

Pode-se resolver intuitivamente para $\alpha_t(i)$ da seguinte maneira:

1) Inicialização:

$$\alpha_1(i) = \pi_i \cdot b_i(O_1), \quad 1 \leq i \leq N \tag{24}$$

2) Indução:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij}\right] \cdot b_j(O_{t+1}), \quad 1 \leq t \leq T-1 \, e \, 1 \leq j \leq N. \tag{25}$$

3) Finalização:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{26}$$

A função desse algoritmo no trecho de avanço (*forward*) é fazer uma varredura entre todas as possibilidades de transição de estado e decidir o caminho ótimo, mas com uma diferença: ele apenas considerará o estado imediatamente anterior, esquecendo todo o passado de transições, isto é, apenas o estado atual e o estado imediatamente anterior serão considerados na busca pelo melhor caminho, isso reduz muito o tempo de cálculo e o número de operações. No entanto, haveria a possibilidade de um estado anteriormente desconsiderado no passado vir a ser o melhor caminho, implicando em uma passagem de estados diferente da encontrada pelo algoritmo. Para isso, existe o trecho de retorno (*backward*) que fará o caminho oposto para confirmar que a sequência de estados descoberta é realmente o melhor caminho.

A Figura 3.5, que pode ser vista abaixo, ilustra a sequência de operações requeridas para o cálculo da variável de avanço $\alpha_{t+1}(j)$.

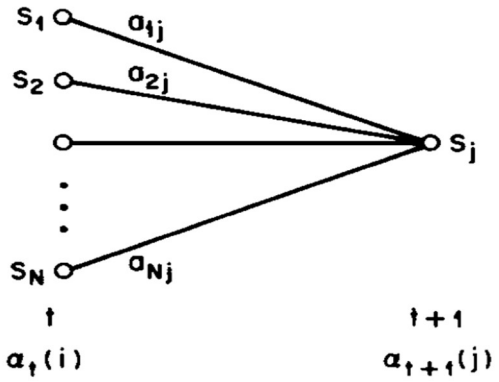

Figura 3.5 – Explicação do método *forward-backward* no trecho *forward*.
Fonte:[31].

No trecho de retorno (*backward*) calculamos $\beta_t(i)$:

$$\beta_t(i) = P(O_{t+1} \cdot O_{t+2} \ldots \cdot O_T | q_t = S_i, \lambda) \quad (27)$$

Esta função é responsável por fazer um mapeamento ao inverso, ou seja, após a escolha de um caminho ótimo em um primeiro momento, o algoritmo vai realizar o mesmo trajeto, mas de trás para frente, buscando verificar se a sequência escolhida é realmente a ideal nesse estudo.

A Figura 3.6, representada abaixo, ilustra a sequência de operações requeridas para o cálculo da variável de retorno $\beta_t(i)$.

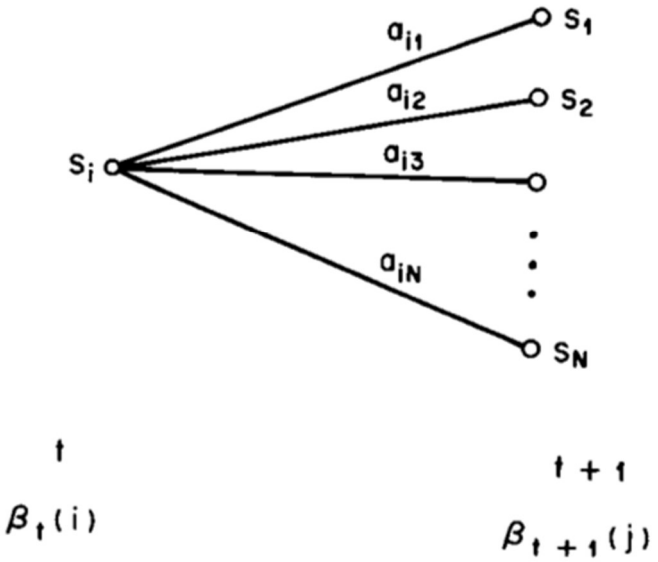

Figura 3.6 – Explicação do método *forward-backward* no trecho *backward*.
Fonte: [31].

Da mesma forma como no primeiro trecho, deduz-se a equação de $\beta_t(i)$ intuitivamente:

1) Inicialização:

$$\beta_T(i) = 1 \qquad 1 \leq i \leq N \tag{28}$$

2) Indução:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j) \qquad t = T-1, T-2, \ldots, 1 \; e \; 1 \leq i \leq N \tag{29}$$

(b) Solução para o segundo problema:

Há muitas possibilidades de definição para o critério de otimização. Uma possibilidade é escolher os estados q_t que sejam individualmente “os preferidos”:

$$\gamma_t(i) = \frac{\alpha_t(i) \cdot \beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i) \cdot \beta_t(i)} \quad (30)$$

Usando $\gamma_t(i)$ podemos resolver para o estado q_t preferido no instante t:

$$q_t = arg \max_{1 \le i \le N} [\gamma_t(i)], \qquad 1 \le t \le T \quad (31)$$

No entanto, mesmo que esse procedimento maximize o número de estados corretos, haveria problemas com a sequência de estados resultante. Poder-se-ia resolver para a sequência de estados que maximize o número esperado de pares de estados corretos (q_t, q_{t+1}) ou a trinca (q_t, q_{t+1}, q_{t+2}), mas o critério mais utilizado é encontrar uma melhor sequência única de estados existente baseado em métodos de programação dinâmica (Algoritmo de Viterbi), método que será descrito em seção posterior.

(c) Solução para o terceiro problema:

Não existe nenhuma forma otimizada conhecida que possa encontrar um modelo que maximize a probabilidade da sequência observada na saída do sistema. Pode-se, no entanto, escolher λ = (A, B, π) de modo que P(O|λ) seja localmente maximizado utilizando um procedimento iterativo tal qual Baum-Welch, ou senão, utilizando técnicas de gradiente.

ξ_t (i, j), a probabilidade de estar em um estado S_i, no tempo t e estado S_j no tempo t + 1 dado o modelo e a sequência de observações, isto é:

$$q_t = arg \max_{1 \le i \le N} [\gamma_t(i)], \qquad 1 \le t \le T \quad (32)$$

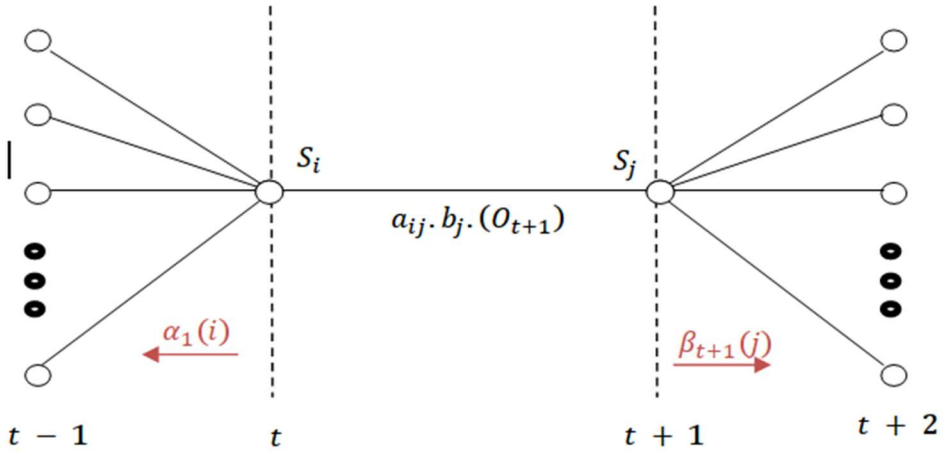

Figura 3.7 - Descrição do método de Baum-Welch
Fonte: Produção própria, baseada em [31].

Escreve-se $\xi_t(i, j)$ na forma:

$$\xi_t(i,j) = \frac{\alpha_t(i) \cdot a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i) \cdot a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) \cdot a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)}$$

$$\delta_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{33}$$

Então:

$$\sum_{t=1}^{T-1} \delta_t(i) = \text{número esperado de transições a partir de } S_i \text{ ;}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{número esperado de transições desde } S_i \text{ até } S_j \text{ .}$$

Um conjunto de fórmulas razoáveis para a reestimação de π, A e B é:

$\overline{\pi_f}$ = frequência esperada (número de vezes) no estado S_i no tempo (t = 1) = $\delta_1(i)$;

$$\overline{a_{ij}} = \frac{número\ esperado\ de\ transições\ do\ estado\ S_i\ ao\ estado\ S_j}{número\ esperado\ de\ transições\ do\ estado\ S_i}$$

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \delta_t(i)} \tag{34}$$

$$\overline{b_j}(k) = \frac{número\ esperado\ de\ vezes\ no\ estado\ S_j\ e\ símbolo\ observado\ v_k}{número\ esperado\ de\ vezes\ no\ estado\ S_j}$$

$$\overline{b_j}(k) = \frac{\sum_{\substack{t=1 \\ s.t.O_t=v_k}}^{T} \delta_t(i)}{\sum_{t=1}^{T} \delta_t(i)} \tag{35}$$

Definindo o modelo atual como $\lambda = (A, B, \pi)$ e usando-se isso para calcular o lado direito das equações acima, atribui-se um modelo reestimado da forma $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$, assim como apresentado no lado esquerdo de tais equações. Foi provado por Baum e seus colegas que: ou o modelo inicial λ definirá um ponto crítico da função de verossimilhança no caso de $\bar{\lambda} = \lambda$, ou, o modelo reestimado $\bar{\lambda}$ é mais adequado que o modelo λ no sentido em que $P(O|\bar{\lambda}) > P(O|\lambda)$. Ou seja, um novo modelo $\bar{\lambda}$ é calculado e este aproxima a sequência observada de uma forma melhor que a anterior.

$$\max_{\bar{\lambda}}\left[Q(\lambda, \bar{\lambda})\right] \rightarrow P\left(O\middle|\bar{\lambda}\right) \geq P(O|\lambda), \tag{36}$$

em que:

$$Q\left(\lambda, \bar{\lambda}\right) = \sum_{Q} P(Q|O, \lambda) \cdot \log_{10}(P(O, Q|\bar{\lambda})) \tag{37}$$

Baseando-se na otimização de Lagrange padrão, utilizando os multiplicadores de Lagrange, podemos dizer que P será máxima quando as seguintes condições forem satisfeitas:

$$\pi_i = \frac{\pi_i \cdot \frac{\partial P}{\partial \pi_i}}{\sum_{k=1}^{N} \pi_k \cdot \frac{\partial P}{\partial \pi_k}} \; ; \; a_{ij} = \frac{a_{ij} \cdot \frac{\partial P}{\partial a_{ij}}}{\sum_{k=1}^{N} a_{ik} \cdot \frac{\partial P}{\partial a_{ik}}} \; e \; b_j(k) = \frac{b_j(k) \cdot \frac{\partial P}{\partial b_j(k)}}{\sum_{\ell=1}^{M} b_j(\ell) \cdot \frac{\partial P}{\partial b_j(\ell)}}$$

Denotamos o conjunto de K $(1 \leq k \leq K)$ observações da forma:

$$O = \left[O^{(1)}, O^{(2)}, \ldots O^{(k)}\right],$$

em que $O^{(k)}$ é a k-ésima sequência de observações.

Torna-se necessário ajustar os parâmetros do modelo para maximizar:

$$P(O|\lambda) = \prod_{k=1}^{K} P\left(O^{(k)}\middle|\lambda\right) = \prod_{k=1}^{K} P_k$$

Como as fórmulas de reestimação são baseadas em frequências de ocorrência de vários eventos, quando há múltiplas sequências de observações as modificamos, então:

$$\overline{a_{ij}} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k - 1} \hat{\alpha}_t^k(i) \cdot a_{ij} \cdot b_j \cdot \left(O_{t+1}^{(k)}\right) \cdot \hat{\beta}_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k - 1} \hat{\alpha}_t^k(i) \cdot \hat{\beta}_t^k(i)} \quad , \tag{38}$$

$$\overline{b_j}(\ell) = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{\substack{t=1 \\ s.t. O_t = v_k}}^{T_k - 1} \hat{\alpha}_t^k(i) \cdot \hat{\beta}_t^k(i)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k - 1} \hat{\alpha}_t^k(i) \cdot \hat{\beta}_t^k(i)} \; , \tag{39}$$

em que $\hat{\alpha}_t^k(i)$ e $\hat{\beta}_t^k(i)$ representam α e β após um procedimento de escala. Isto é necessário visto que são números decimais que se multiplicam muitas vezes, chegando a tal ponto que se tornam ínfimos e a precisão dos computadores não pode mais calcular.

3.3 ALGORITMO DE VITERBI

O algoritmo de Viterbi é uma ferramenta essencial dentro do estudo de modelos ocultos de Markov. Proposto por Andrew Viterbi em meados da década 1960, tem como função encontrar a sequência mais provável de estados a ter produzido uma determinada saída observada, no caso funcionando como solução para o segundo problema citado anteriormente.

No caso de um lançamento de dados às escuras, por exemplo, em que não se sabe exatamente qual o procedimento utilizado para o sorteio nem quantos dados estão sendo lançados, uma saída possível a ser observada é *O* = 2, 6, 1, 1, 5, 1, 6, 3, 4 e 4. Em um caso como esse, o algoritmo de Viterbi buscará encontrar a sequência de estados mais provável que gerou essa sequência durante o sorteio. Se cada estado representar um dado sendo lançado, uma sequência de estados possível seria: dado 1, dado 3, dado 4, dado 1, dado 2, dado 3, dado 2, dado 4, dado 1 e dado 4 em um modelo de 4 estados.

Seja $\delta_i(t)$ a probabilidade do caminho percorrido mais provável terminando no estado i, em um tempo t.

$$\delta_i(t) = \max_{q_1,q_2,\ldots,q_{t-1}} P(q_1\, q_2\, \ldots\, q_{t-1}, q_t = i, O_1\, O_2\, \ldots O_t|\lambda) \tag{40}$$

Sendo π_i a matriz de probabilidades de estados iniciais em t = 1, então $\delta_j(t)$ pode ser calculado recursivamente usando:

$$\delta_j(t) = \max_{1 \le i \le N} [\delta_i(t-1) \cdot a_{ij}] \cdot b_i\,(O_t) \tag{41}$$

Assim como a inicialização:

$$\delta_i(1) = \pi_i \cdot b_1(O_1) \quad 1 \le i \le N \tag{42}$$

em que N é o número total de estados do modelo, a finalização será:

$$P^* = \max_{1 \leq i \leq N} [\delta_i(T)] \tag{43}$$

em que T representa o tempo correspondente ao último símbolo observado na saída. Em suma, isto significa que, chegando ao final da sequência, será calculada a maior probabilidade no ponto de saída e então será feito o caminho inverso para localizar a melhor passagem [31].

Para ilustrar esse conceito, basta imaginar uma nova máquina caça-níquel fictícia instalada dentro de um cassino. Essa máquina possui três janelas, semelhante a um *jackpot* tradicional. A diferença é que, nesta nova máquina, entre as três janelas, apenas uma será atualizada a cada rodada, escolhida de forma aleatória, mas cuja probabilidade de escolha é conhecida para cada uma das três janelas. Nesse ponto, é importante fazer a distinção da máquina tradicional em cassinos que atualizará as três janelas de uma só vez.

A cada vez que o jogador insere uma moeda, aperta o botão e puxa a alavanca, tal qual mostrado pela Figura 3.8, uma das janelas irá girar e para cada janela haverá quatro símbolos possíveis de aparecer: limão, cerejas, barra e sete. Cada um destes símbolos possui uma probabilidade de aparição dependendo da janela sorteada. O jogador será premiado quando as três janelas mostrarem o mesmo símbolo.

O sistema apresentado será modelado como uma HMM de três estados onde cada estado corresponderá a uma das janelas, então o vetor de estados é *S* = {*janela 1*, *janela 2*, *janela 3*}.

A lógica neste caso é que, se um usuário anterior abandonou o jogo com uma sequência {*sete, sete, limão*} nas respectivas janelas, o novo jogador terá de torcer para dois eventos: a máquina escolher a terceira janela para ser atualizada e nesta janela o símbolo *sete* ser sorteado. Assim, a nova sequência será {*sete, sete, sete*} e o jogador ganhará o prêmio. Caso ocorresse a atualização da janela 1 ou janela 2 o jogador não ganha o prêmio nesta rodada e teria de colocar novas moedas na expectativa de acontecer o esperado. O mesmo pode ocorrer se a janela 3 atualizar para um símbolo diferente de *sete*.

Figura 3.8 – Máquina similar à descrita.
Retirado de:
<http://www.top10casinoeurope.com/2016/04/19/slots-machines-win>.
Acesso em 18 nov. 2015, 16:10.

As probabilidades de emissão de símbolos são dadas por:

$$b_1(limão) = 0{,}7;\ b_1(cerejas) = 0{,}3;\ b_1(barra) = 0{,}2;\ b_1(sete) = 0{,}25;$$

$$b_2(limão) = 0{,}1;\ b_2(cerejas) = 0{,}4;\ b_2(barra) = 0{,}4;\ b_2(sete) = 0{,}6;$$

$$b_3(limão) = 0{,}2;\ b_3(cerejas) = 0{,}3;\ b_3(barra) = 0{,}4;\ b_3(sete) = 0{,}15;$$

E a matriz de probabilidades de transição de estados é dada por:

$$a_{ij} = \begin{bmatrix} 0{,}15 & 0{,}8 & 0{,}05 \\ 0{,}6 & 0{,}1 & 0{,}3 \\ 0{,}4 & 0{,}25 & 0{,}35 \end{bmatrix}$$

A probabilidade de estado inicial é π_i = 0,45.

Agora supondo que após algumas rodadas, o jogador anota os três últimos símbolos que variaram, mas esquece de anotar em qual janela eles apareceram, aparece em sua agenda: {*sete, limão, limão*}. Há o interesse em descobrir a sequência de estados ótima que pode ter gerado essa sequência de símbolos, ou seja, saber

quais foram as janelas que rodaram para aparecer a sequência mostrada. Executando o algoritmo de Viterbi manualmente, temos:

$$\delta_j(1) = \pi_i \cdot b_j(sete)$$
$$\delta_j(2) = \max_{1 \le i \le 3}[\delta_i(1) \cdot a_{ij}] \cdot b_j(limão)$$
$$\delta_j(3) = \max_{1 \le i \le 3}[\delta_i(2) \cdot a_{ij}] \cdot b_j(limão)$$

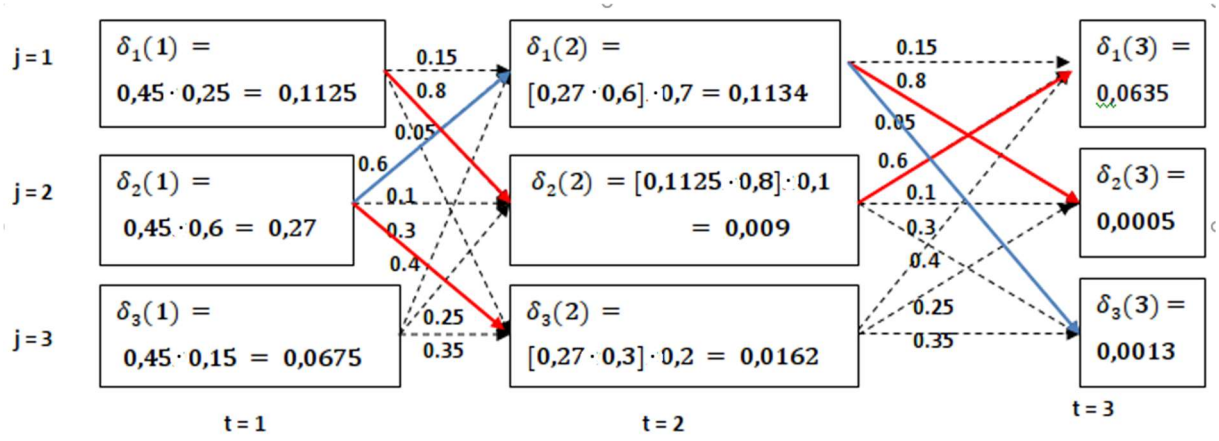

Figura 3.9 - Exemplo do cálculo das probabilidades.
Fonte: Produção própria.

Em um primeiro momento (t=1), são inicializadas as probabilidades para $\delta_j(1) = \pi_i \cdot b_j(sete)$ com j = 1,2,3. O resultado é mostrado na primeira coluna da esquerda. No segundo passo (t = 2), determina-se primeiramente $\delta_1(2)$ considerando as três possibilidades de transição: $\delta_1(1) \cdot a_{11} = 0{,}1125.0{,}15 = 0{,}016875$; $\delta_2(1) \cdot a_{21} = 0{,}27 \cdot 0{,}6 = 0{,}162$ *e* $\delta_3(1) \cdot a_{31} = 0{,}0675 \cdot 0{,}4 = 0{,}027$. O maior é o segundo, então, de acordo com o algoritmo, atualiza-se $\delta_j(2) = 0{,}162 \cdot b_j(li\ \ ão) = 0{,}162 \cdot 0{,}7 = 0{,}1134$. Então o melhor caminho para se chegar em j=1 no tempo t=1 é a partir do terceiro estado. Repetem-se os mesmos passos para j = 2 e j = 3. Segue-se então a mesma linha de pensamento e $\delta_j(3)$é atualizado utilizando o algoritmo de Viterbi. A linha azul aponta as melhores probabilidades de caminho descobertas enquanto as linhas vermelhas apontam caminhos secundários.

Como há apenas três símbolos observados, é possível agora passar ao passo de finalização. Neste caso, o melhor caminho é aquele das setas azuis, ou seja, Q = 2, 1, 3, com probabilidade de P^* = 0,0013. Observa-se que, na verdade, em t = 3, o

estado j=1 obteve a maior probabilidade, mas como o caminho escolhido anteriormente foi o estado 2, a melhor transição do estado 1 seria para o estado 3. Mas após isso, o algoritmo fará o caminho reverso para definir qual realmente é o melhor caminho.

As Figuras 3.10 e 3.11, mostradas a seguir, auxiliam a compreensão de como é feita essa seleção de estados:

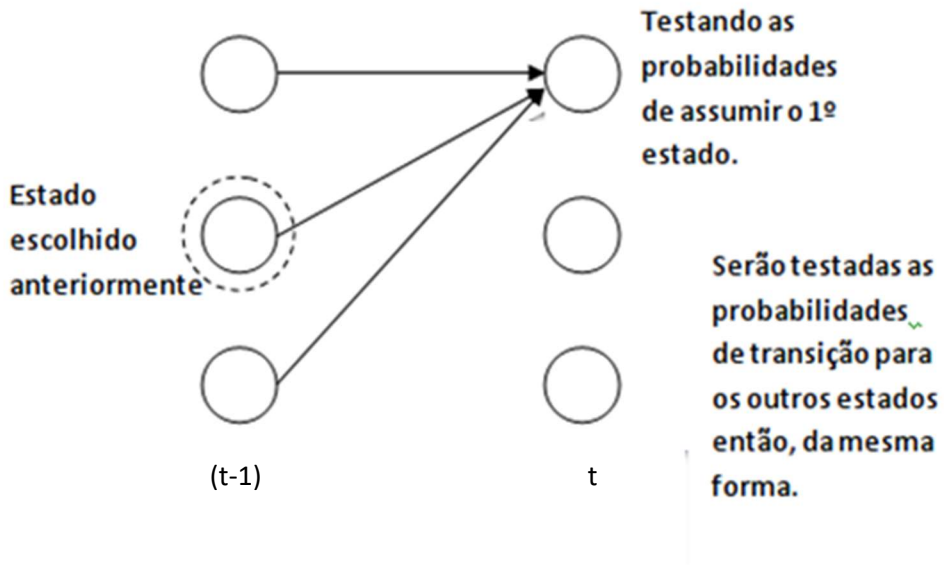

Figura 3.10 - Apenas os estados de (t-1) e t interessam nessa fase.
Fonte: Produção própria, baseada em [31].

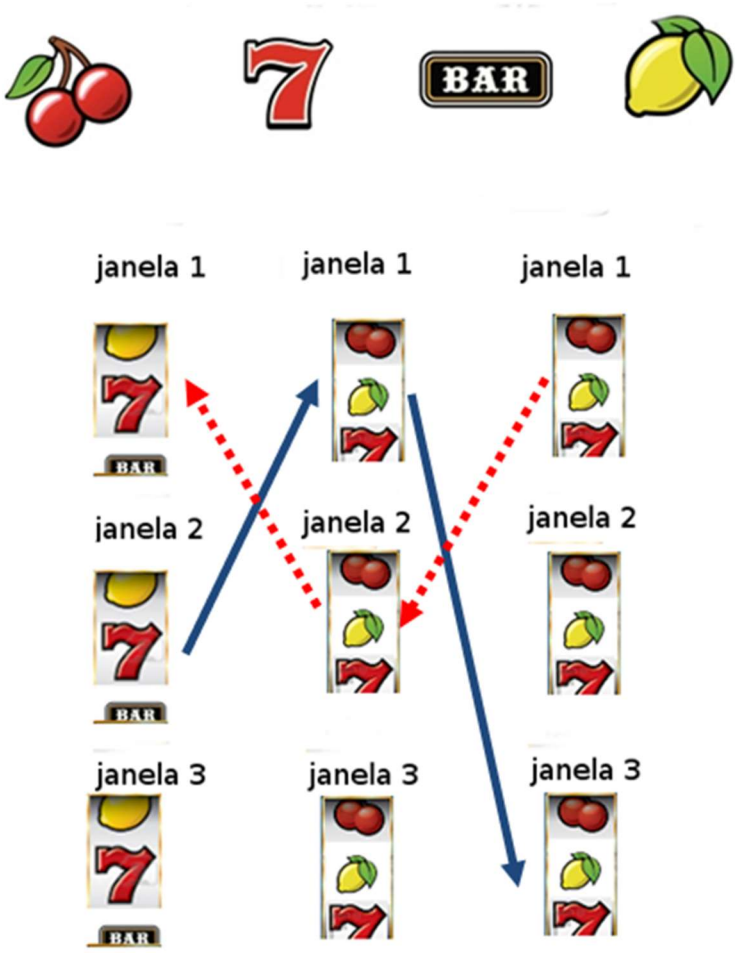

Figura 3.11 - Reconstrução do caminho de sequência ótima.
Produção própria.

A Figura 3.11 representa de forma ilustrativa a sequência de janelas ótima calculada pelo algoritmo de Viterbi tanto no passo de *forward* (setas azuis), quanto no passo de *backward* (setas tracejadas vermelhas).

Outro caso relevante dentro desse trabalho é o exemplo em que o algoritmo de Viterbi é utilizado para reconhecimento de palavras isoladas, a situação ilustrativa é o caso em que um canal de transmissão ruidoso acaba cortando os acentos das palavras de um texto, a partir de uma mensagem com a palavra “abóbora”, o receptor recebe a mensagem “abobora”. Muitas mensagens poderiam levar a essa observação : Ábobora, aboborá, abõbora, âbobõra,... abóbora... Pode-se investigar a probabilidade de cada letra analisando sua probabilidade de aparição na língua portuguesa e procurando a probabilidade de que uma letra apareça no instante em que é precedida por outra e assim por diante, definindo o que se chama bigramas ou trigramas e aplicando o algoritmo para descobrir qual era a mensagem original.

3.4 RECONHECIMENTO DE FALA UTILIZANDO HMMs

O diagrama na Figura 3.12 mostra os principais procedimentos no reconhecimento de fala desde o sinal gravado através de um microfone até seu reconhecimento após análise:

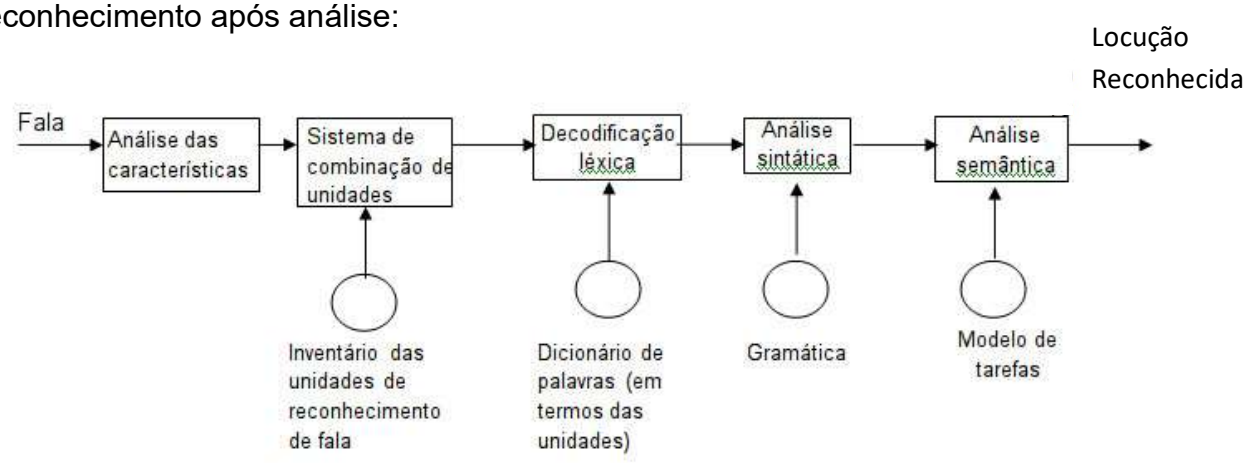

Figura 3.12 - Diagrama de blocos mostrando as etapas do reconhecimento.
Fonte: Produção própria, baseada em [31].

A primeira etapa é a análise temporal e/ou espectral do sinal de fala. Esta fase é útil para encontrar vetores que podem ser utilizados no treinamento de HMMs que caracterizam vários sons de fala. Em seguida, é preciso definir uma unidade de referência para a análise dos símbolos de fala, podendo ser utilizados fones, conjuntos de dois ou três fones, palavras inteiras, sub-palavras, sílabas, meias-sílabas, entre outras unidades, onde essa será parâmetro de referência para a análise.

Durante a etapa de decodificação léxica, serão impostas restrições ao sistema combinatório de unidades, tanto que as passagens investigadas são correspondentes à sequências de unidades de fala que estão no dicionário de palavras. Assim como a decodificação léxica impõe condições de combinação entre as possibilidades, a análise sintática será responsável por dizer se os trechos investigados correspondem às palavras e se essas estão dispostas em uma boa sequência.

Depois disso, a análise semântica irá impor novas restrições ao sistema reconhecedor de fala. Uma forma na qual são utilizadas restrições semânticas é via um modelo dinâmico do estado do reconhecedor [31].

Outra tarefa importante é separar o fundo silencioso dos trechos de fala. Há pelo menos três maneiras de realizar isso:

- Baseando-se na energia e duração de um sinal;
- Criando um modelo de silêncio de fundo, leia-se, modelo estatístico, e representando um sinal como uma sequência arbitrária de momentos de fala e silêncio:

(silêncio) – fala – (silêncio)

- Estendendo as unidades dos modelos de fala de forma que o silêncio de fundo seja incluído junto com primeiro ou o último estado do modelo.

3.4.1 Reconhecimento de palavras isoladas

Para cada uma das *V* palavras de um vocabulário será necessário construir um modelo oculto de Markov λ^v, isto é, precisamos estimar os parâmetros do modelo (*A*, *B*, *π*) que otimizam a verossimilhança do conjunto de vetores de observações no treinamento das V palavras.

Em cada palavra desconhecida a ser analisada, o processamento da Figura 3.12 precisará ser realizado. Esse procedimento é chamado de medição da sequência observada via análise das características de fala correspondente à palavra, seguido do cálculo das verossimilhanças do modelo para todos os modelos possíveis, $P(O|\lambda^v)\ 1 \leq v \leq V$, seguido pela seleção que mais corresponde ao modelo, ou:

$$v^* = \underset{1 \leq v \leq V}{\operatorname{argmax}}[P(O|\lambda^v)] \tag{44}$$

Esse cálculo probabilístico normalmente é efetuado servindo-se do algoritmo de Viterbi (isto é, a máxima verossimilhança é usada) e requer uma ordem de $V{\cdot}N^2{\cdot}T$ cálculos, onde:

$$V = número\ de\ palavras\ do\ vocabulário;$$
$$N = número\ de\ estados;$$
$$T = número\ de\ observáveis.$$

4. DESENVOLVIMENTO, RESULTADOS E DISCUSSÕES

4.1 DESENVOLVIMENTO DO RECONHECEDOR

Para se analisar os resultados obtidos através do reconhecimento de fala utilizando o HTK, faz-se necessário, primeiramente, programar o software para que possa executar o reconhecimento desejado e para isso, etapas de treinamento, configuração, teste e análise de resultados são necessárias. Essas etapas levam um bom tempo até atingir o resultado esperado e pode-se afirmar que é o ponto mais complicado do presente trabalho.

Com o intuito de explicar de forma mais aprofundada todo o desenvolvimento do reconhecedor de fala, assim como ajudar futuramente outros pesquisadores que venham a lidar com as mesmas dificuldades encontradas, produziu-se um guia de utilização desta ferramenta explicitando os detalhes para se fazer funcionar um reconhecedor de dígitos entre zero e nove, com todas as etapas descritas em português. Algo difícil de encontrar atualmente, pois boa parte dos tutoriais disponíveis sobre o assunto apenas trazem informações sobre cada ferramenta individualmente do HTK, mas poucos mostram um passo a passo que saia do início e chegue até os resultados. O próprio manual disponível no HTK apresenta mais de 300 páginas com muitos exemplos específicos, mas não ajuda aquele que está trabalhando com a ferramenta pela primeira vez por não seguir um caminho lógico passo a passo.

O guia descrito é apresentado neste trabalho no apêndice A e representa em 40 páginas as etapas necessárias desde a instalação do HTK até a finalização, inclusive fornecendo *scripts* para facilitar a coleta e análise de resultados.

Utilizou-se as vozes gravadas de 35 locutores distintos, sendo eles:

- 15 do gênero masculino;
- 20 do gênero feminino.

Todas as vozes gravadas em áudio estéreo (dois canais), amostradas à frequência de 44100 Hz, com cada locutor dizendo, em arquivos de áudio independentes, os dígitos entre zero e nove.

No total de todos os testes realizados, 436 arquivos de áudio diferentes foram utilizados. Destes, 90 foram produzidos por um único locutor para análise da dependência do locutor. O total de sua duração, incluindo silêncios antes e depois da locução do dígito é 715,538 segundos, o que equivale a aproximadamente 12 minutos. A duração média destes arquivos é 1,641 segundos.

Inicialmente, decidiu-se por converter as frequências de amostragem para 16000 Hz e transformar cada arquivo de áudio em mono (apenas um canal). Para isso serviu-se do *software Audacity* para esta função, mas esta tarefa pode ser agilizada com alguns comandos no terminal do *Ubuntu* (distribuição de sistema operacional *Linux, open-source*).

Estes 35 locutores foram divididos em vários casos diferentes com o intuito de analisar as variações nos índices de acertos dependendo de quem treinaria e quem acertaria, assim como verificar o “caso ótimo” deste sistema. A cada grupo deu-se um número e ao total oito formatos diferentes de teste foram desenvolvidos:

Tabela 5 - Especificação dos grupos de treinamento e teste.

Número do teste	Quem treina	Quem testa
0	17 locutores sendo: 8 homens e 9 mulheres.	18 locutores sendo 7 homens e 11 mulheres.
1	15 locutores, todos homens.	20 locutores, todos mulheres.
2	20 locutores, todos mulheres.	15 locutores, todos homens.
3	18 locutores sendo 7 homens e 11 mulheres.	17 locutores sendo: 8 homens e 9 mulheres.
4	5 locutores, todos mulheres.	30 locutores, sendo 15 homens e 15 mulheres.
5	5 locutores, todos homens.	30 locutores, sendo 10 homens e 20 mulheres.
6	Apenas um locutor treina com diversos tipos de voz.	35 locutores, sendo 20 mulheres e 15 homens.
7	35 locutores, sendo 20 mulheres e 15 homens.	Apenas um locutor testa com diversos tipos de voz.

Lembrando apenas que em todos os casos descritos pela Tabela 5, os locutores treinadores não irão testar o reconhecimento. Nos dois últimos casos, a única voz que treina no teste 6 e testa no teste 7 é a do autor do trabalho (voz masculina), onde foram gravados nove vezes os dígitos entre zero e nove utilizando expressões diferentes em cada gravação, em alguns casos voz aguda, em outros voz grave, assim como fala lenta, fala veloz, voz rouca, voz com entonação crescente, como se estivesse perguntando (“um?”), voz com entonação constante, voz com entonação de tristeza. Isso tenta representar alguns formatos de voz para testar em que casos o reconhecedor teria mais dificuldade de acertar.

Deu-se ênfase aos casos que analisam a diferença entre gênero dos locutores que formam a base de informações que o HTK usará para comparar quando testando e aqueles que testam. Outra possibilidade seria dividir os locutores em grupos formados a partir de sua faixa etária, tentando-se treinar com jovens, por exemplo, e testar com idosos ou adultos. Este caso não será discutido neste trabalho, pois dentre todos os áudios utilizados, nenhum locutor era menor de 18 anos.

Algo interessante a se destacar é que, a maior parte dos áudios utilizados foi gravada por pessoas passando por dificuldades de saúde e que muitas vezes têm sua voz afetada, causando rouquidão e dificuldades de fala. Optou-se por não formar um novo banco de dados apenas com pessoas que não apresentam problemas de saúde visto que os prazos para recrutar locutores voluntários e processar os arquivos de áudio estavam curtos. No entanto, seria interessante uma análise futura deste caso para eventuais comparações.

Outra causa possível para alguns resultados ruins é que, durante todo o trabalho permitiu-se que o HTK fizesse toda a segmentação dos arquivos de áudio buscando os fones utilizados no dicionário fonético gerado, assim como atribuindo a cada um dos áudios os fones contidos, sem dizer a duração de início e término de cada um deles.

Além das diferentes locuções utilizadas no treinamento e teste, outros parâmetros são importantes e diretamente relacionados aos resultados, como o comprimento da janela, o intervalo entre duas janelas, o tipo de coeficientes (MFCC ou LPC) utilizados, o número de reestimações realizadas durante o treinamento, o número de estados de cada modelo oculto de Markov, o tipo de janela utilizada, o número de coeficientes MFCC, entre outros. No âmbito deste trabalho, foram analisados os cinco primeiros parâmetros citados.

Essa conversão descrita a seguir é válida quando se usa áudio já gravado anteriormente ou através da ferramenta *HWave* do HTK, e não dados coletados em tempo real. De acordo com [32], o período entre cada vetor de parâmetros determina a taxa de amostragem na saída e é configurada pelo parâmetro *TARGETRATE*. Neste trabalho, por vezes tal parâmetro será citado como taxa de vetores por segundo ou número de vetores de parâmetros. O trecho da forma de onda utilizado para determinar cada vetor de parâmetros é normalmente referenciado como a janela. Seu comprimento é especificado pelo parâmetro *WINDOWSIZE*. A Figura 4.1 descreve essa etapa.

O parâmetro SOURCERATE está ligado de maneira inversamente proporcional à taxa de amostragem original do arquivo de entrada.

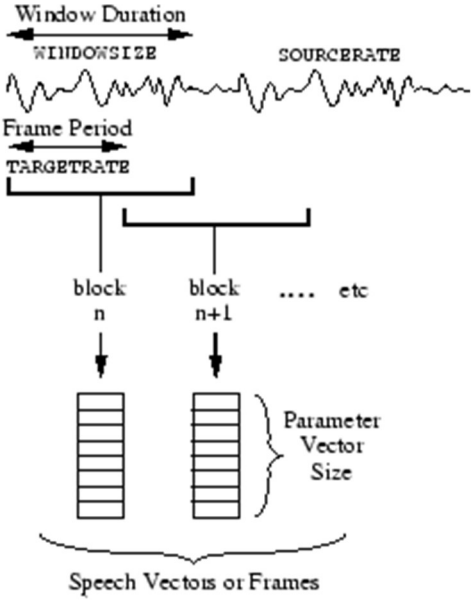

Figura 4.1 - Parâmetros de conversão de áudio para MFCC.
Fonte: [32]

Nota-se, pela Figura 4.1, que cada trecho do arquivo de fala especificado na entrada do reconhecedor, tanto para treinamento quanto para testes, é convertido em “vetores de fala” de acordo com as características desejadas. Essa conversão faz-se necessária para que os áudios sejam traduzidos para a “linguagem do HTK”.

Nesta etapa também especifica-se se a janela utilizada será a de Hamming, assim como o coeficiente utilizado no filtro de pré-ênfase, o número de canais do banco de filtros triangulares, o tipo de codificação MFCC ou LPC e suas variantes e o número de coeficientes mel-cepstrais.

Optou-se por utilizar o enjanelamento de Hamming, coeficiente de pré-ênfase de 0,97, usualmente empregado em estudos de reconhecimento de fala, enquanto o número de coeficientes mel-cepstrais é inicialmente 12 com 26 canais no banco de filtros triangulares (vide Figura 2.18) e codificação MFCC.

Sobre os tipos de MFCC, utilizou-se para a maior parte do trabalho "*MFCC_0_D_A*":

- *MFCC_0*: Significa que C_0 será utilizado como componente de energia;
- *MFCC_C*: Utiliza coeficientes mel-cepstrais compactados;
- *MFCC_E*: Converte os coeficientes mel-cepstrais com componente de energia logarítmica acrescentado.
- *MFCC_0_D*: Além de utilizar C_0 como componente de energia, utiliza coeficientes delta.
- *MFCC_E_D*: Além de utilizar componente de energia logarítmica, acrescenta coeficientes delta.
- *MFCC_0_D_A*: Além de utilizar C_0 como componente de energia, utiliza coeficientes delta e coeficientes de aceleração.
- *MFCC_E_D_Z*: Além de utilizar componente de energia logarítmica, acrescenta coeficientes delta e normalização cepstral média.
- *MFCC_E_D_A*: Além de utilizar componente de energia logarítmica, acrescenta coeficientes delta de aceleração ao conteúdo.

As mesmas variações são possíveis utilizando-se LPC ao invés de MFCC, exceto o fato de que LPC não pode utilizar o C_0 como coeficiente de energia. Busca-se entender as variações que cada um destes casos pode causar aos resultados, então alguns destes casos são testados com o reconhecedor de fala desenvolvido.

A Figura 4.2, extraída de [7], compara o tamanho de diversos vetores de parâmetros LPC (o que nesse caso é equivalente para MFCC), dados os diferentes tipos de codificação citados.

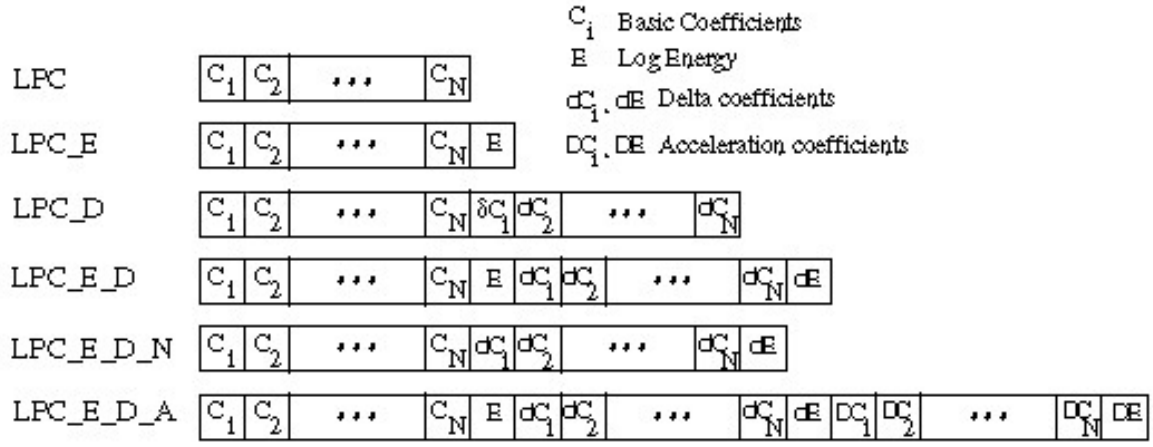

Figura 4.2 - Comparação entre o tamanho dos vetores para diferentes formas de codificação.
Fonte: [7]

Segundo [7], todas as formas parametrizadas dos arquivos de dados do HTK consistem em uma sequência de vetores. A Figura 4.2, demonstra que o tamanho desses vetores varia em função desses especificadores citados.

Outro fator de interesse no estudo do reconhecimento de fala é como o número de estados de uma HMM pode afetar os resultados. Todos os testes iniciais foram executados utilizando 5 estados, mas também há interesse em variar e descobrir como afeta os resultados. A dificuldade de se fazer variar os estados de um modelo oculto de Markov no HTK é a necessidade de se gerar um novo modelo protótipo para cada número de estados a ser testado, por isso optou-se fazer apenas poucos testes variando o número de estados dentro do possível.

A matriz de possibilidades de transição é calculada pelo HTK a partir dos dados coletados durante o treinamento, ou seja, não é necessário o usuário dizer as probabilidades de variação entre fones.

O dicionário fonético foi criado visando diferenciar diferentes fones encontrados nas palavras referentes aos dígitos utilizados.

Após o treinamento, um arquivo de texto é gerado contendo matrizes, cujos elementos são atualizados a cada reestimação. Essas matrizes dão uma estimação das probabilidades de transição entre estados e observáveis. E no momento do teste, o HTK utilizará esse arquivo de texto na análise do reconhecimento de fala. Isto implica que, o HTK, uma vez treinado, não necessita dos arquivos de áudio originais ou dos arquivos convertidos para o formato .mfc, ele necessita apenas desse arquivo

contendo as matrizes calculadas. O que permite ocupar um espaço reduzido de memória (normalmente alguns kBytes) durante uma implementação prática.

No momento do teste, o programa buscará associar os dados dos arquivos de teste com os dados armazenados para o treinamento. Caso não encontre a resposta correta, ele apontará para um outro dígito que foi o mais provável encontrado, dado o processamento efetuado. Por exemplo, o locutor disse “cinco”, se o programa encontrar a resposta correta, sua saída será: “cinco”, caso contrário, direcionará para outro dígito.

O apêndice A descreve com detalhes todas as etapas utilizadas para se programar o HTK visando o reconhecimento de dígitos isolados desde a instalação do software até a coleta de resultados.

Os parâmetros controlados durante os experimentos realizados são resumidos na Tabela 6, mostrada a seguir:

Tabela 6 - Descrição dos parâmetros estudados no âmbito deste trabalho.

Nome do parâmetro	**Descrição**
Número de reestimações	Quantas vezes o HTK reestimará o modelo até definir a matriz de treinamento
TARGETRATE	Taxa de vetores de parâmetros por segundo
WINDOWSIZE	Comprimento da janela em ms
Número de estados	Tamanho do modelo oculto de Markov utilizado para modelar cada fone
Falas utilizadas durante treinamento e teste	Diferentes testes realizados variando-se locutores que treinavam e testavam
Características de parametrização	Tipo de codificação utilizada e suas respectivas variantes. Por exemplo: Componentes derivativos e de aceleração, energia, etc.

4.2 RESULTADOS E DISCUSSÕES

O primeiro passo foi tentar encontrar as melhores configurações de parâmetros do HTK para um caso específico dentre os oito previamente descritos no capítulo 4. Escolheu-se o primeiro caso, com 17 locutores que emprestam sua voz para o treinamento e 18 locutores que emprestam sua voz para testes como caso geral e a partir deste tentou-se variar os parâmetros de número de reestimações do algoritmo de Baum-Welch (descrito pelas equações 33 a 39), assim como o comprimento da janela e a taxa de vetores de parâmetros.

A estimativa inicial era:

- 7 reestimações;
- Janela de 20 ms;
- 150 vetores de parâmetros por segundo.

O tipo de MFCC utilizado na maior parte dos testes era "*MFCC_0_D_A*" e os outros parâmetros são descritos pelo capítulo 4.

Primeiramente, buscou-se variar o número de reestimações e saber como se comporta o reconhecedor de fala em função deste parâmetro. Fixando a janela e a taxa de vetores de parâmetros com os valores anteriormente citados, fez-se variar o número de reestimações do modelo entre 1 e 14. Os resultados são mostrados na Figura 4.3 e os dados utilizados para traçá-la estão disponíveis no apêndice B, Tabela B.1. Todas as figuras deste capítulo foram traçadas pelo software *SciDAVis*. Algo importante de se destacar é que, na maioria dos gráficos plotados, usou-se linhas para ligar os pontos, estas linhas não representam que entre esses dois pontos há valores contidos nessa linha, é apenas uma forma para se facilitar a visualização, pois os dados aqui demonstrados são relativos à estatísticas.

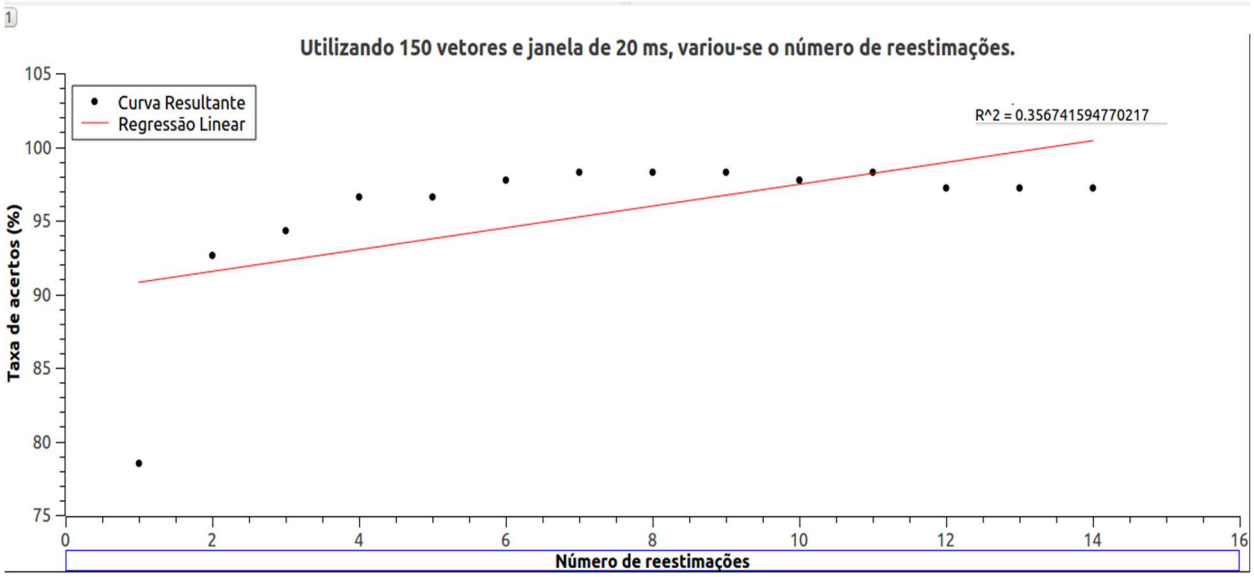

Figura 4.3 - Análise da variação do número de reestimações do caso inicial.

Conseguiu-se para este caso uma taxa de acertos máxima de 98,31 % usando entre 7 e 9 reestimações. Um número baixo de reestimações aumenta significativamente a taxa de erros, mas, embora aumentar o número de reestimações melhore o índice, aumentar muito não significa atingir o máximo. Como observado na Figura 4.3, utilizar mais do que 11 reestimações diminui a taxa de acertos, além de deixar o reconhecimento de fala cada vez mais lento. Visando utilizar o menor número de reestimações que proporcionasse o melhor resultado, decidiu-se por seguir utilizando 7 reestimações por enquanto.

Em seguida, decidiu-se variar o tamanho da janela entre 16 ms e 150 ms, mantendo a taxa de vetores por segundo em 150 e com 7 reestimações. Os resultados são mostrados na Figura 4.4, assim como na Tabela B.2.

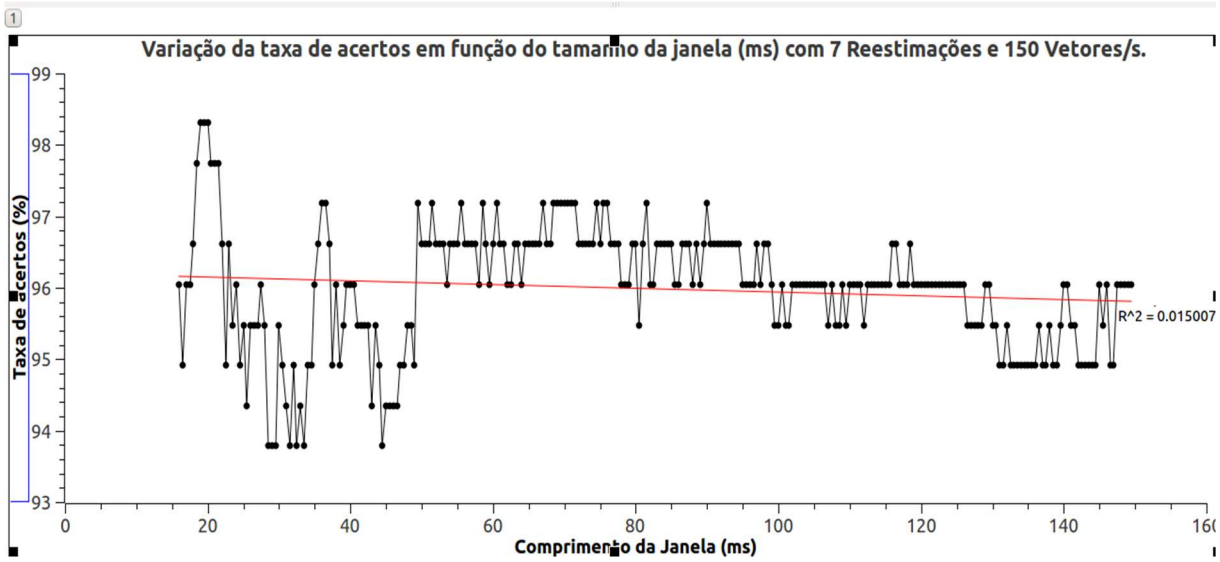

Figura 4.4 - Variação da janela com 7 reestimações e 150 vetores por segundo.

Nota-se, observando a Figura 4.4, que o caso ótimo está situado em torno de 20 ms e, aumentando a janela, a tendência é de diminuição da taxa de acertos, mas essa variação não ocorre de forma linear. Assim, pode-se afirmar que, para cada reconhecedor de fala desenvolvido, haverá um intervalo em que a janela será ideal, dependendo não somente dos dados utilizados no treinamento, mas também do número de reestimações e da taxa de vetores de parâmetros (como mostrado na sequência).

O comprimento da janela está intimamente ligado à capacidade de captar unidades fonéticas dentro de uma frase, ou seja, uma janela mais longa pode captar mais de um fone de uma vez, enquanto uma janela muito curta pode não captar fones inteiros.

A partir disso, usando valores de janela de 20 ms e 35 ms e mantendo o número de reestimações em 7 e todos os outros parâmetros constantes, decidiu-se por fazer variar a taxa de vetores de parâmetros por segundo, observando o que acontece aos resultados, mostrados na Figura 4.5 e Tabelas B.3 e B.4.

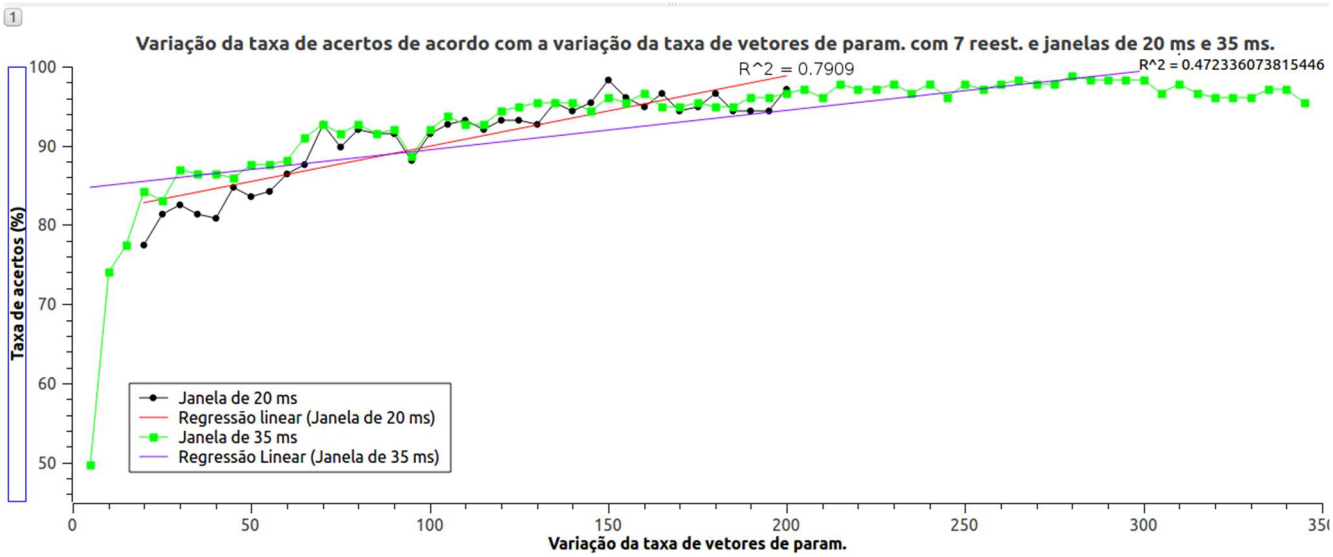

Figura 4.5 - Comportamento do reconhecedor com a variação da taxa de vetores de parâmetros por segundo.

A curva verde na Figura 4.5 representa a variação para uma janela de 35 ms enquanto a curva preta representa o caso em que se usa uma janela de 20 ms. Neste caso, o número de vetores por segundo divido por 10 não pode ser maior que o comprimento da janela em milissegundos, por isso o traçado negro termina em 200 vetores/s, enquanto o verde continua até 350 vetores/s.

Analisando os dados obtidos, nota-se que o aumento da taxa de vetores por segundo causa uma tendência de aumento na taxa de acertos, mas há um limite. Percebe-se que, a partir de determinado ponto, a taxa de acertos passa a oscilar em torno de determinado ponto e inclusive diminui quando a taxa de vetores se aproxima de 350 com janela de 35 ms.

Notou-se que, neste caso, o máximo encontrado foi de 98,87 % para uma janela de 35 ms e 280 vetores por segundo, melhorando inclusive o índice anterior de 98,31 %. A conclusão que se tira disso é que a janela de 20 ms era ideal para o caso em que 150 vetores de parâmetros eram utilizados, mas aumentando o número de vetores de parâmetros, o que sugere uma melhora nos resultados, uma janela maior, de 35 ms, apresenta melhores taxas de acertos.

Atribuindo-se agora um padrão de 35 ms para a janela e 280 vetores de parâmetros por segundo, fez-se variar novamente o número de reestimações para descobrir se 7 ainda era o número mínimo ideal para o reconhecedor de fala. Resultados são apresentados então na Figura 4.6, que faz uma comparação entre o caso anterior e o caso atual, assim como nas Tabelas B.1 (caso anterior) e B.5 (caso atual).

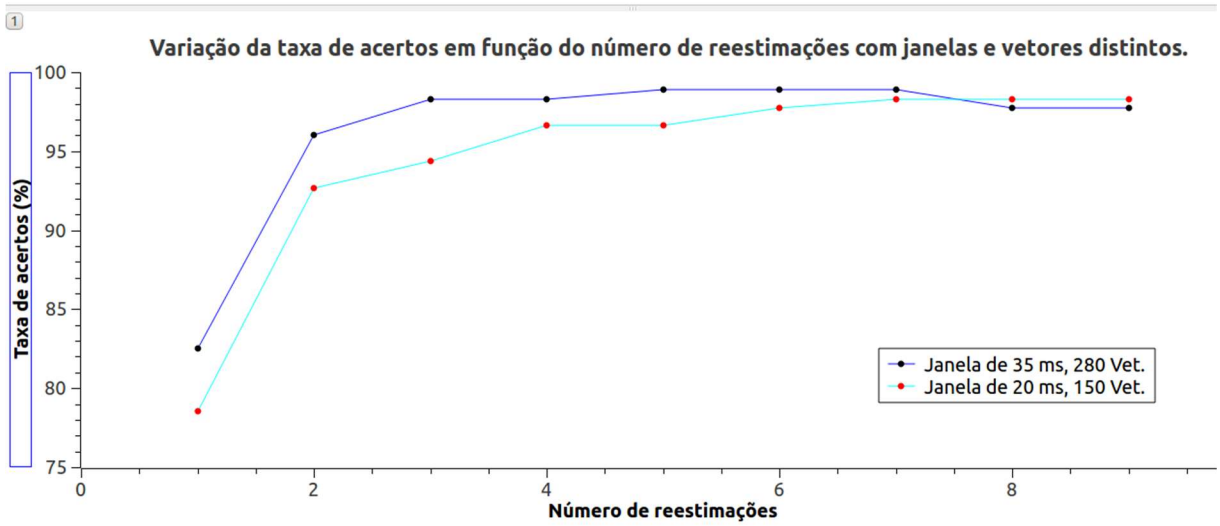

Figura 4.6 - Comparação entre a variação do número de reestimações para dois casos diferentes.

Fez-se variar o número de reestimações entre 1 e 9 e coletou-se os dados quando a janela era de 20 ms e com 150 vetores/s, assim como quando a janela era de 35 ms, utilizando-se 280 vetores/s. O que é possível afirmar é que houve uma melhora significativa dos resultados para o caso onde usa-se uma janela maior e mais vetores/s. Antes era necessário 7 reestimações para uma taxa de acertos de 98,31 %, agora são necessários apenas 5 reestimações para uma taxa de acertos de 98,87 %. Assim sendo, além de melhorar a chance de acerto, o reconhecedor torna-se mais rápido, exigindo menos reestimações e menos processamento para isso.

Agora buscou-se, mais uma vez, saber se o tamanho da janela utilizada era realmente o ideal, considerando 5 reestimações e uma taxa de vetores de parâmetros por segundo de 280, lembrando que todas as outras variáveis apresentadas no capítulo 4 são mantidas constantes.

Variando-se a janela entre 28 e 199,5 ms, já que o comprimento da janela em milissegundos não pode ser menor que 10% do número de vetores por segundo utilizado (280 neste caso), mantendo 5 reestimações, obteve-se os resultados disponibilizados na Figura 4.7 e baseados na Tabela B.6.

Figura 4.7 - Variação do comprimento da janela com 5 reestimacões e 280 vetores de parâmetros por segundo.

Como sugere o coeficiente R^2 da curva de regressão linear igual a 0,835 apresentado na Figura 4.7, há uma forte tendência de piora quando se aumenta o tamanho da janela, mas uma janela muito pequena também causa diminuição da taxa de acertos. Observou-se que o ideal neste caso era uma janela de 36,5 ms, o que gerou uma taxa de acertos de 99,44 %, melhorando o índice anterior para uma janela de 35 ms. Este foi o melhor caso encontrado dentre todos os testados até então e significa que, entre 177 dígitos falados, o reconhecedor acertou 176.

Fixando agora o número de vetores por segundo em 280 e utilizando a janela de 36,5 ms, busca-se verificar se o número de reestimações é melhorado ou permanece estagnado. A Figura 4.8 mostra os resultados, assim como as Tabelas B.7 (caso atual), B.1 e B.5 (casos anteriores).

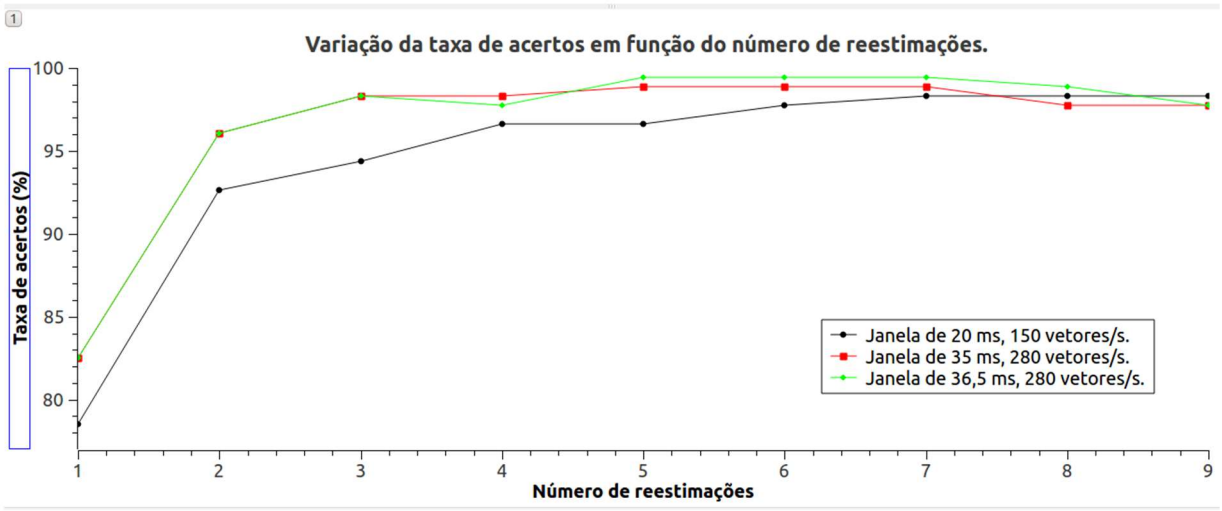

Figura 4.8 - Comparação entre a taxa de acertos em relação ao número de reestimacões para três casos distintos.

A curva em preto representa o primeiro caso, com comprimento de janela de 20 ms e 150 vetores por segundo, enquanto as curvas em vermelho e verde representam, respectivamente, um comprimento de janela de 35 ms e 36,5 ms com 280 vetores de parâmetros por segundo.

Os resultados do último teste desta etapa são mostrados pela Figura 4.9, assim como pela Tabela B.9.

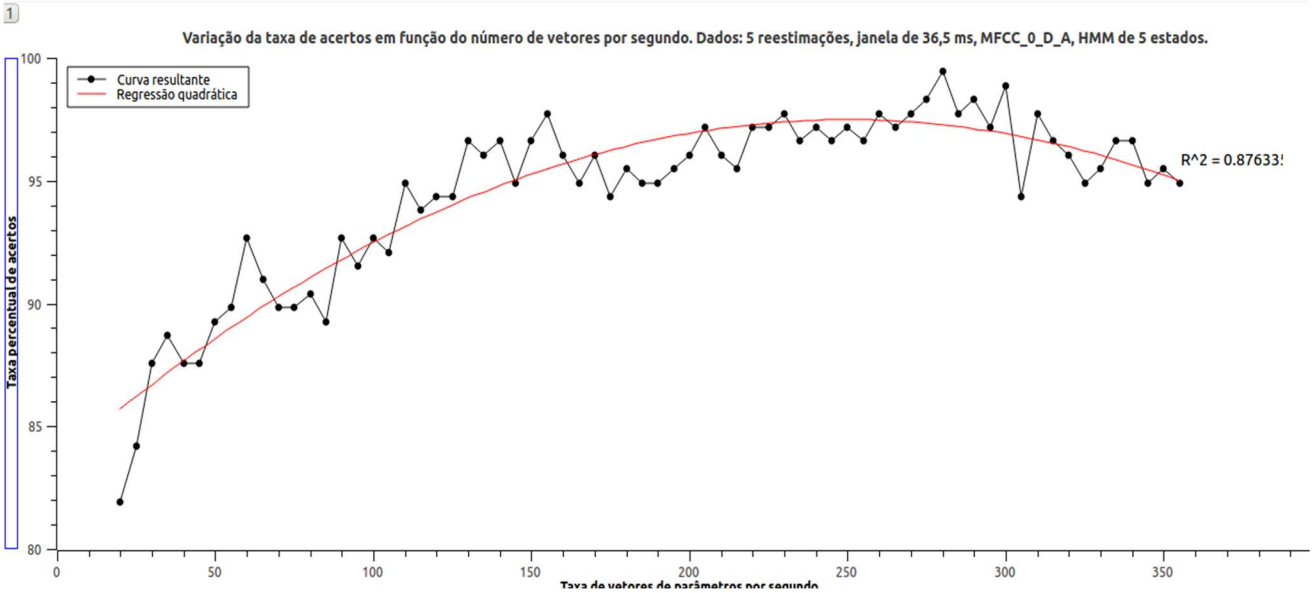

Figura 4.9 - Variação da taxa de vetores por segundo para uma janela de 36,5 ms e 5 reestimações.

Desta forma, pode-se afirmar que 280 vetores por segundo é a taxa ideal para esse reconhecedor. A tendência é de aumento com o aumento do número de vetores por segundo.

Além disso, a partir dos resultados representados pela Figura 4.8, pode-se dizer que, embora o índice de acertos aumente para 5 reestimações, não há uma melhora em relação a 4 ou menos reestimações. Assim sendo, 5 reestimações é o ideal para este reconhecedor.

Volta-se a recordar que todos os resultados apresentados até aqui referem-se apenas ao teste número 0, descrito pela Tabela 5. Busca-se aqui o caso ideal para um teste e os mesmos parâmetros serão utilizados para se testar os outros casos descritos na Tabela 5 para efeito de comparação. Este teste foi escolhido por apresentar um caso geral em que locutores homens e mulheres de diversas faixas etárias treinam o reconhecedor, assim como outro grupo diferente irá testar.

4.3 COMPARAÇÃO ENTRE DIFERENTES TREINAMENTOS E TESTES

Uma vez encontrado os parâmetros ideais para um reconhecedor de fala, é hora de mantê-los fixos e variar a forma de treinamento e testes, buscando entender, efetivamente, em que casos ele funciona melhor e em que casos deixa a desejar.

O gráfico de barras ilustrado pela Figura 4.10, gerado a partir da Tabela B.8, compara o desempenho deste reconhecedor para os diferentes testes ilustrados na Tabela 5.

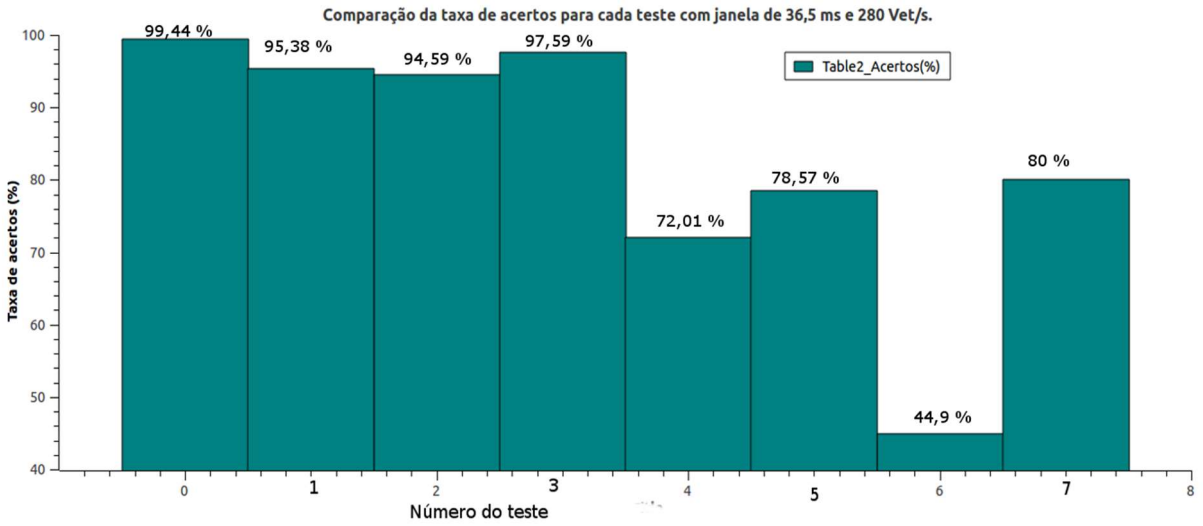

Figura 4.10 - Comparação entre os resultados dos diferentes testes ilustrados na Tabela 5.

Os melhores resultados foram obtidos para o caso em que o grupo original (caso 0) de 17 locutores (8 masculinos, 9 femininos) treinam o HTK e 18 locutores (7 masculinos, 11 femininos) testam o reconhecedor de fala com 99,44% de acertos e o caso contrário em que os 17 locutores que anteriormente treinaram irão testar o reconhecedor de fala treinado pelos 18 locutores que anteriormente testaram, neste caso, 97,59 % de acertos.

Outros casos com bons resultados foram os casos 1 e 2, em que apenas um gênero treinava e o outro testava. No caso em que apenas homens treinavam e apenas mulheres testavam (caso 1), a taxa de acertos foi de 95,38 %, enquanto que no caso em que todas as mulheres treinavam e todos os homens testavam, a taxa de acertos foi de 94,59 %.

Quando tentou-se diminuir o número de treinadores e treinar apenas com 1 gênero e todos os outros locutores, masculinos e femininos, que não participaram do treinamento, testavam o reconhecimento de fala, houve uma diminuição significativa da taxa de acertos para 72,01 % quando apenas 5 mulheres treinaram o sistema e 78,57 % quando apenas 5 homens treinavam o HTK.

O pior resultado foi o do caso 6, em que apenas um locutor treinava o reconhecedor com nove diferentes expressões vocais e todos os outros locutores testavam o reconhecimento de fala. Este caso obteve apenas 44,9 % de acertos.

O último caso (caso 7) foi o inverso do caso 6: todos os locutores disponíveis anteriormente treinavam o HTK, enquanto o locutor do caso 6 testava. Este caso obteve 80 % de acerto.

Pode-se concluir, a partir de todos esses resultados, que o HTK foi configurado com o caso 0 sendo o caso ótimo, ou seja, parâmetros de reestimação, comprimento da janela, taxa de vetores de parâmetros por segundo foram definidos em função deste caso, então é esperado que ele tenha as melhores taxas de acertos. Talvez a variação de alguns desses parâmetros para os outros casos ajude a melhorar as taxas de acertos, mas isso não seria interessante, pois dificultaria a comparação, visto que é esperado um teste sob as mesmas condições para todos os casos.

Os parâmetros configurados podem atingir seu ponto ótimo em função de cada treinamento e das vozes utilizadas neles, quando se varia os dados de treinamento, o HTK assume outros valores para a modelagem dos modelos ocultos

de Markov, diferentes probabilidades de transição entre estados e símbolos observáveis.

Outra análise que se faz é que diminuindo o número de locutores no treinamento, as chances de acerto são reduzidas, pois menos informação estará disponível para o HTK formular os modelos. No caso 3 ocorre que o número de locutores treinando o HTK aumentou, mas em compensação são 11 mulheres e 7 homens, enquanto no teste são 9 mulheres e 8 homens. Os resultados pioraram em comparação com o caso inicial. Isso pode ser devido a diversidade vocal utilizada no treinamento. Em outras palavras, os dados utilizados no treinamento do caso 0 podem representar uma variação vocal mais abrangente do que no treinamento do caso 3 e quando testava-se com os outros locutores, os locutores do caso 3 poderiam ter em sua voz características não muito bem representadas no treinamento.

O pior caso ocorreu quando apenas um locutor médio treinou a ferramenta. Mesmo tentando representar diversas expressões vocais, apresentadas na Tabela 7, não foi possível representar todas as variantes necessárias contidas nas 35 vozes que testaram os reconhecedores.

Tabela 7 - Variantes vocais utilizadas para treinar o HTK no teste 6.

Expressão utilizada	Descrição
Fala lenta	Fala cada dígito de forma arrastada.
Fala alegre	Fala de forma animada, semelhante a voz do apresentador Sílvio Santos.
Fala rápida	Fala cada dígito bem rapidamente.
Fala aguda	Fala com a voz em tom bem alto, semelhante ao de uma criança.
Fala grave e trêmula	Fala de forma parecida ao do apresentador Cid Moreira.
Fala rouca	Fala de forma grave e rouca, semelhante à do ex-presidente Lula.
Fala crescente	Fala como se estivesse fazendo um questionamento.
Fala nervosa	Fala exclamando como se estivesse dando uma bronca.
Fala normal	Fala em ritmo constante, sem alteração de tom.

No caso da Tabela 7, as características citadas são apenas atribuições dadas pelo autor do texto para distinguir entre falas distintas, onde cada expressão pode ser interpretada de forma diferente por outro ouvinte.

Mesmo que mais variações da mesma voz tivessem sido utilizadas, seria muito difícil ter uma boa taxa de acertos porque cada voz possui características distintas e representá-las por apenas um locutor que não possui perícia para tal tarefa, tal qual um imitador, coralista profissional ou um dublador, é praticamente impossível.

Um detalhe importante observado é que inicialmente tentou-se distinguir durante o treinamento os locutores que falavam "*seti*" e os locutores que falavam "*sete*", assim como, os locutores que falavam "*novi*" e os locutores que diziam "nove", ou até mesmo "nov", dizendo ao HTK que os fones ditos para estes dígitos poderiam ser diferentes dependendo do locutor. O que ocorreu é que a taxa de acertos diminuiu para 93,7 % quando especificava-se diferentes vogais reduzidas no dicionário enquanto quando não se fazia essa distinção, a taxa de acertos aumentava para 99,4% novamente. Especificar as vogais reduzidas pode não ser relevante em um corpus pequeno.

Talvez essa distinção possa confundir o HTK, porque em muitos casos as pronúncias eram semelhantes entre si, então poderia acontecer do dígito ser dito "*novi*" e o HTK não diferenciar do "nove", acabando por modelar os fones de forma incorreta. Outro fator que pode acabar influenciando neste caso é que o "nove" normalmente tem uma pronúncia mais anasalada do que os outros dígitos.

Tabela 8 - Análise da performance de um reconhecedor para diferentes expressões vocais.

Característica da fala	**Dígitos reconhecidos incorretamente**
Fala lenta	2, 4, 5, 6, 8 e 9
Fala alegre	6
Fala rápida	1, 5, 6, 7 e 9
Fala aguda	nenhum
Fala grave e trêmula	5
Fala rouca	9
Fala crescente	9
Fala nervosa	7 e 8
Fala normal	9

Agora, discutindo o último caso, em que 35 locutores (20 femininos e 15 masculinos) treinam e apenas um locutor com as variantes vocais descritas na Tabela 7 testa, buscou-se descobrir em que casos as falhas aconteciam. A Tabela 8 mostra os erros cometidos pelo reconhecedor neste caso.

Os casos que causaram mais erros foram os casos relativos à velocidade em que o dígito é falado, tanto quando falado muito lentamente quanto quando falado muito rapidamente. Ambos os casos geraram apenas 40 % de acertos. O que pode-se inferir é que o HTK foi treinado por locutores que, em média, falam os dígitos em um ritmo constante, o que leva o reconhecedor de fala a reconhecer os dígitos facilmente quando falados naquela velocidade, mas quando há uma variação grande na velocidade de locução em relação à média dos locutores que treinaram, isso acaba dificultando a identificação.

Algo que também pode causar essa diferença é o tamanho da janela e parâmetro de taxa de vetores por segundo terem sido ajustados para uma situação, no caso utilizar um comprimento de janela diferente aumenta a probabilidade de acertos, o que será discutido em seção posterior.

Em termos de velocidade, considerou-se pelo autor uma fala normal, comparando-se apenas o tempo utilizado para pronunciar o dígito zero, sem considerar silêncios, uma média de 0,794 s entre 15 amostras.

Uma fala rápida seria 0,329 s, por exemplo, ou com duração menor do que 0,5 s enquanto uma fala lenta teria duração superior 1,1 s, no caso da locução em fala lenta seria de 1,3 s.

No primeiro caso em que houve apenas 1 erro entre 177 tentativas, o erro foi cometido por um locutor que além de falar com voz muito rouca, falou o dígito 9 rapidamente, o que corrobora com a hipótese de que a velocidade de locução seja fator agravante neste reconhecedor de fala, visto que se utiliza um tamanho de janela fixa configurado anteriormente para maximizar o número de acertos em um caso em que os locutores falavam cada dígito em ritmo constante.

Outra informação que se pode absorver da Tabela 8 é que o dígito 9 foi reconhecido incorretamente em 5 dos 9 casos, o que pode dizer que a maior dificuldade de reconhecimento esteja neste dígito, seguido pelos dígitos 5 e 6 (3 erros em 9 tentativas). Os dígitos 0 e 3 foram os únicos em que não causaram erros de reconhecimento, enquanto os dígitos 1, 2 e 4 foram reconhecidos corretamente em 8 de 9 tentativas. Algo que talvez possa ser a razão é que os dígitos com maior chance

de erro possuem mais fones em comum com outros dígitos, enquanto os dígitos que menos produzem erros possuem fones modelados unicamente para eles.

Outra hipótese é o caso do dígito nove ser dito de diferentes formas, “nove”, “nov”, “novi”, criando assim uma confusão na definição do último fone. Também pode ser agravante o fato de que em geral os locutores seguiam uma sequência de gravação crescente, dizendo primeiro o dígito 0 e terminando pelo 9. Normalmente os locutores possuem uma tendência a falar o último dígito de forma diferente dos anteriores. O ideal seria utilizar também gravações de uma sequência de dígitos decrescentes ou mixta para poder se tirar mais informação deste caso.

A Tabela 9 foi gerada justamente para esta análise. Deseja-se saber se um dígito determinado é mais fácil de ser reconhecido do que outros. Analisando sete dos oito testes descritos pela Tabela 5, montou-se uma relação entre cada teste e o número de erros para cada dígito.

Tabela 9 - Análise dos erros de dígitos cometidos durante cada teste

Teste\Dígito	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Total de erros do teste**
0	0	0	0	0	0	0	0	0	0	1	1
1	1	0	2	1	2	0	2	1	0	0	9
2	1	1	0	0	1	0	1	0	0	4	8
3	1	1	0	0	1	0	1	0	0	0	4
4	7	16	3	1	9	13	2	8	0	23	82
5	15	5	6	0	10	9	5	6	0	8	64
7	0	1	1	0	1	3	3	2	2	5	18
Total de erros de dígito	25	24	12	2	24	25	14	17	2	41	

A partir da Tabela 9, que também é chamada de “matriz de confusão”, em estudos de mineração de dados, pode-se dizer que o dígito 9 é o que mais produziu erros durante os testes executados (aproximadamente 22%), mas, diferentemente do que se imaginava, outros dígitos produziram quantidade considerável de erros. O diagrama da Figura 4.11 representa a quantidade percentual de erro para cada dígito:

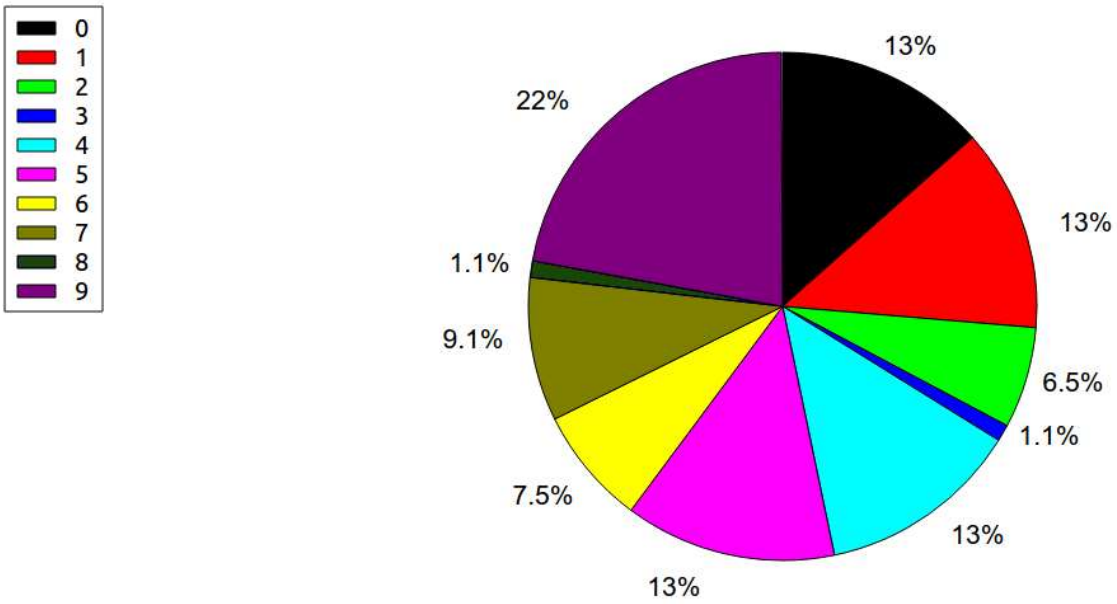

Figura 4.11 - Diagrama de pizza para análise dos erros vinculados a cada dígito.

Percebe-se que os dígitos 3 e 8 representam cada um, apenas 1,1 % do total de erros. Isso quer dizer que é muito difícil para esse reconhecedor de errar esses dígitos. Assim sendo, pode-se dizer que há um compromisso entre a taxa de acertos e o dígito falado pelo locutor, independentemente do treinamento. Isso se deve à própria fonética desses dígitos que, quando pronunciado, distinguem-se dos outros mais facilmente. O que se pode notar tambem e que os dígitos com mais erros apresentam vogais medias, que apresentam caracteristicas formanticas mais “confusas”, mais instáveis e variáveis. Uma solução seria aumentar a quantidade de ocorrências desses dígitos nos treinamentos.

Analisando a Tabela 9, percebe-se que determinados testes produziram mais erros para determinados dígitos do que outros testes. Isto é, o teste 4 gerou 23 erros para o dígito 9 e 16 erros para o dígito 1, enquanto o teste 5 gerou 8 erros para o dígito 9 e apenas 5 para o dígito 1. No teste 5, houveram mais erros para o dígito 0, o que não era tão significativo no teste 4.

A tentativa agora é analisar se o gênero dos locutores envolvidos está relacionado com esses dados. Para isso, serão analisados primeiramente os casos 1 e 2 onde apenas um dos gêneros treina o reconhecedor de fala e o outro gênero testa.

No caso 1, 15 locutores do gênero masculino treinaram o HTK e 20 locutores do gênero feminino testaram e no caso 2 inverteram-se os papeis. Os resultados da análise para esses dois casos são mostrados na Figura 4.12:

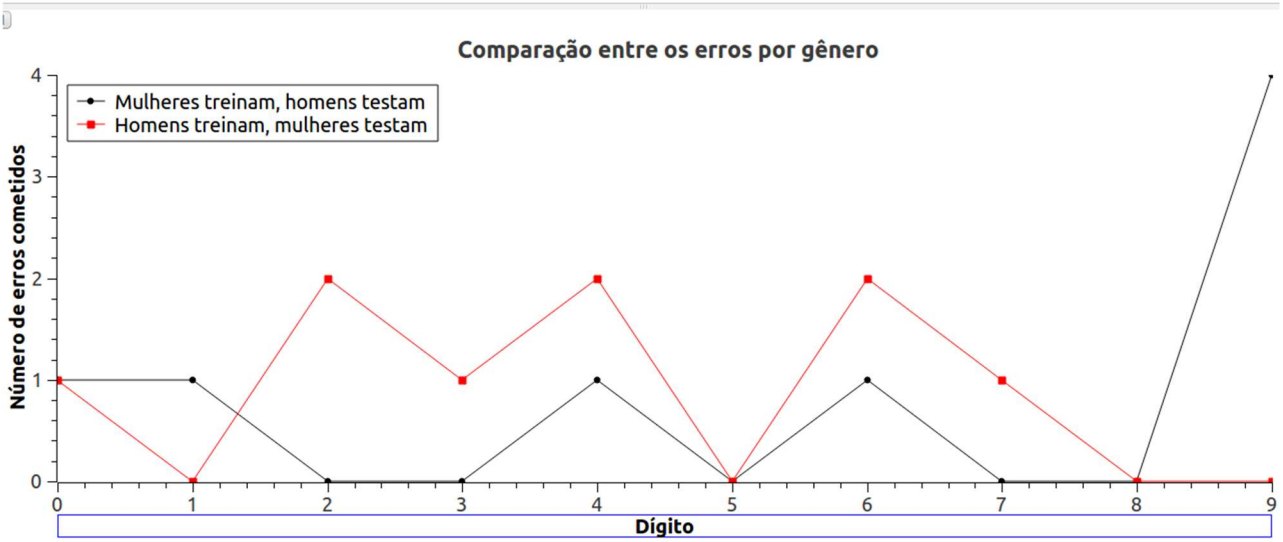

Figura 4.12 - Comparação entre os erros nos testes 1 e 2.

Da mesma forma, uma comparação foi feita envolvendo os testes 4 e 5 e o resultado é mostrado na Figura 4.13:

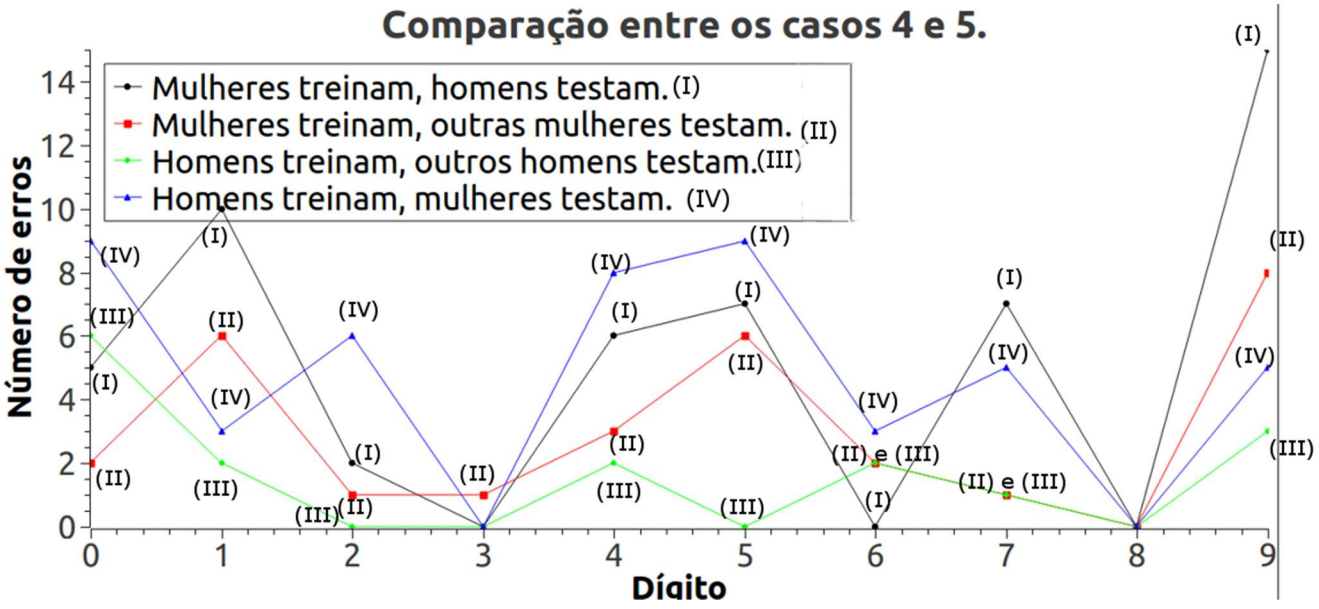

Figura 4.13 – Comparação entre os erros dos testes 4 e 5.

Observando as Figuras 4.12 e 4.13 apresentadas, pode-se perceber que, em ambos os gráficos, o dígito 9 é o que mais erros apresenta, mas também que erra-se mais os dígitos 1 e 9 quando mulheres treinam e os homens testam. Já para os dígitos 2, 4 e 6, a maior taxa de erro se dá quando os homens treinam e as mulheres testam. Como há de se esperar, os melhores resultados são obtidos quando locutores do mesmo gênero treinam e testam o reconhecimento de fala. Isso leva à conclusão de que um reconhecimento de fala, quando treinado por locutores de apenas um gênero, pode ser mais eficiente quando testado por locutores do mesmo gênero. Mas, a taxa de acertos é diretamente proporcional à quantidade de locutores que treinam o reconhecedor de fala.

Embora estas conclusões sejam válidas para este reconhecedor testado com esse banco de dados utilizado, não se pode generalizar e dizer que este é um caso geral. Para fazer afirmações deste gênero seria primeiramente necessário expandir os testes para um grupo muito maior de locutores, para verificar se uma característica amostral pode ser considerada característica da população.

4.4 COMPARAÇÃO ENTRE DIFERENTES FORMAS DE CODIFICAÇÃO

Até este momento, todos os testes executados utizaram apenas uma forma de codificação. Em linguagem do HTK, "*MFCC_0_D_A*", isso significa que foram utilizados coeficientes mel-cepstrais utilizando o primeiro elemento do vetor C_0 como componente de energia acrescentado de componentes delta relacionados às derivadas dos parâmetros e componentes de aceleração.

Um dos intuitos deste trabalho é observar o comportamento do reconhecedor de fala para diversos tipos de codificação dos parâmetros e tentar encontrar alguma evidência útil no desenvolvimento deste tipo de ferramenta.

A Tabela B.10 representa a variação da taxa de acertos em função dos diversos tipos de codificação no caso em que se testa cada um dos casos com os parâmetros encontrados para o melhor resultado do teste 0. A comparação é feita através da Figura 4.14, mostrada na sequência.

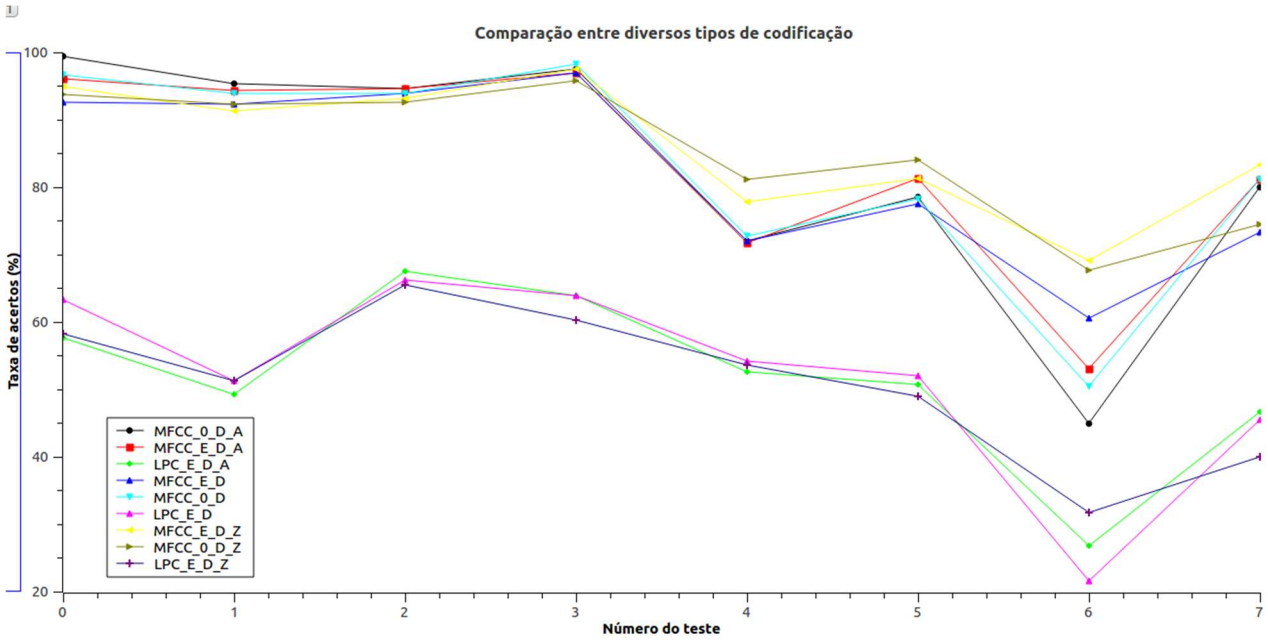

Figura 4.14 - Variação dos resultados para cada teste em função da codificação utilizada.

Estes parâmetros com letras indicadas são exemplificados na seção 4.1. Mais detalhes sobre essa formatação da codificação pode ser encontrada em [7], na seção 5.10.1. A partir dos dados representados, é possível analisar que a codificação "*MFCC_0_D_A*", padrão utilizada até aqui, é válida para os testes 0 e 1, mas não é a melhor escolha para todos os testes.

Por exemplo, para o teste 2, em que apenas locutores do gênero feminino treinam e apenas locutores do gênero masculino treinam, uma escolha melhor seria "*MFCC_E_D_A*" que, ao invés de considerar o elemento de energia como C_0, utiliza uma função logarítmica para descrever este elemento. Já para o teste 3, que representa o teste 0 com as funções trocadas, a melhor escolha seria utilizar "*MFCC_0_D*", que não leva em conta os componentes de aceleração, apenas os componentes delta.

Uma análise válida é que, para os testes em que menos locutores treinaram o HTK como nos testes 4, 5 e 6, destacam-se as codificações "*MFCC_0_D_Z*" e "*MFCC_E_D_Z*", que não consideram os componentes de aceleração, mas sim considera coeficiente estático de média nula para a análise. O que sugere que, para os casos em que há menos informação no treinamento, utilizar essa codificação é mais interessante do que considerar os coeficientes de aceleração.

Não foi possível testar o caso em que não são considerados os coeficientes delta, pois o HTK demanda diferentes arquivos protótipo para este caso e como não foi possível encontrar um modelo de exemplo, preferiu-se analisar apenas os casos descritos acima.

Outra análise possível é que, quando se utiliza LPC ao invés de MFCC, os resultados pioram bastante. Isto pode ser devido ao fato de que o processo de codificação é diferente (descrito no apêndice C). LPC utiliza um modelo preditivo linear para gerar uma representação compacta do envelope espectral, o que pode sugerir que seja mais sensível a erros do que MFCC. Mas também deve-se ter em mente que esta análise é para um caso específico utilizando parâmetros específicos. Pode ocorrer de haver parâmetros diferentes como comprimento de janela, número de reestimações, tamanho do vetor de parâmetros ou especificações de filtro que possam se adequar de forma mais precisa à codificação LPC.

4.5 COMPARAÇÃO ENTRE DIFERENTES NÚMEROS DE ESTADOS DO MODELO DE MARKOV

Da mesma forma como podem-se variar diversos parâmetros relacionados à forma como se codifica os vetores de parâmetros utilizados pelo HTK para modelar os modelos ocultos de Markov, também é possível dizer ao HTK quantos estados terá cada modelo.

Testou-se para três tamanhos distintos o comportamento da taxa percentual de acertos em função deste parâmetro, considerando-se o teste original (teste 0) com os parâmetros tidos como ideais: tamanho de janela de 36,5 ms, 280 vetores de parâmetros por segundo, 5 reestimações do modelo, codificação MFCC com componentes derivativos e de aceleração, assim como considerando o coeficiente C_0 como componente de energia do sinal.

O resultado é apresentado pelo gráfico de barras da Figura 4.15, assim como pela Tabela B.11 do apêndice B.

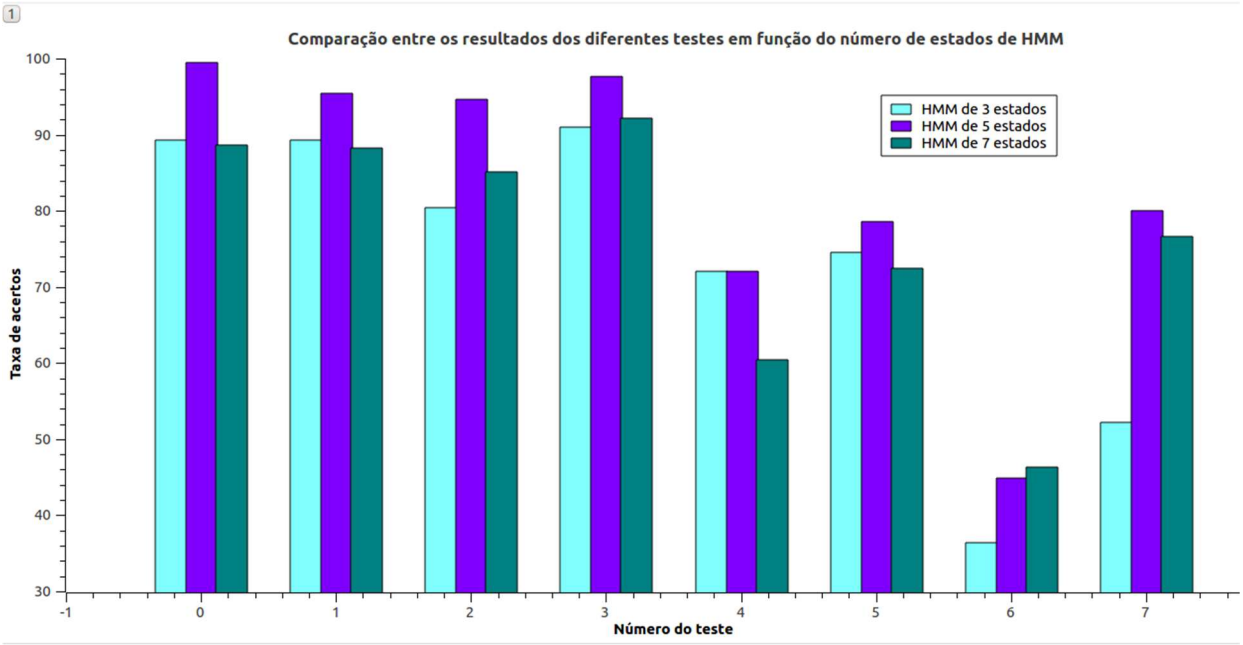

Figura 4.15 - Comparação entre a taxa percentual de acertos e a variação do número de estados de cada HMM.

O que se infere a partir da Figura 4.15 é que, para a maior parte dos casos, um modelo de 5 estados é menos susceptível a erros que um modelo de 3 estados ou 7 estados. No caso do modelo de 3 estados, ele modelará cada fone de forma menos informação possível, pois utilizará uma matriz 3x3 ao invés de uma matriz 5x5, então é esperado que seus resultados sejam inferiores aos do caso com 5 estados.

Já quando se utiliza o modelo de 7 estados, ocorreram alguns problemas para utilizar alguns poucos arquivos de treinamento, pois o HTK não foi capaz de modelar em 7 estados informações retiradas destes arquivos. Mas o modelo de 7 estados também se mostrou inferior ao modelo de 5 estados. Isso pode significar que um número elevado de estados não modelaria tão bem cada fone quanto um modelo de 5 estados no caso em que há muita diversidade de fala.

Um caso que foge desta regra é o teste número 6, em que o percentual de acertos foi melhor para um modelo de 7 estados do que para um modelo de 5 estados. Vale lembrar que o caso 6 utiliza apenas a fala de um locutor durante todas as etapas de treinamento, mas que utiliza diversas características distintas em sua fala. Isto pode sugerir que, dentro de um caso em que há pouca diversidade de fala, ou até mesmo uma grande variação na velocidade de fala, um modelo de mais estados seria mais interessante de se utilizar.

4.6 OTIMIZANDO O CASO DEPENDENTE DE LOCUTOR

É possível melhorar o desempenho do reconhecedor utilizado no teste 6 (até então o pior caso) e comparar os efeitos aplicando as mesmas configurações para os outros testes.

Analisando a Figura 4.16 nota-se que no caso do teste 6, em que apenas um único locutor treina o reconhecedor de fala, a codificação que apresenta melhor resultado é “MFCC_E_D_Z”. Se utiliza então essa codificação e, fazendo variar o comprimento da janela, busca-se encontrar o ponto em que a taxa de acertos é mais uma vez maximizada.

Primeiramente, utiliza-se como estimativa inicial: 2 reestimações, que é o valor que representava o melhor resultado quando se tinha a codificação “MFCC_0_D_A” aplicada ao teste 6 e também 280 vetores de parâmetros por segundo.

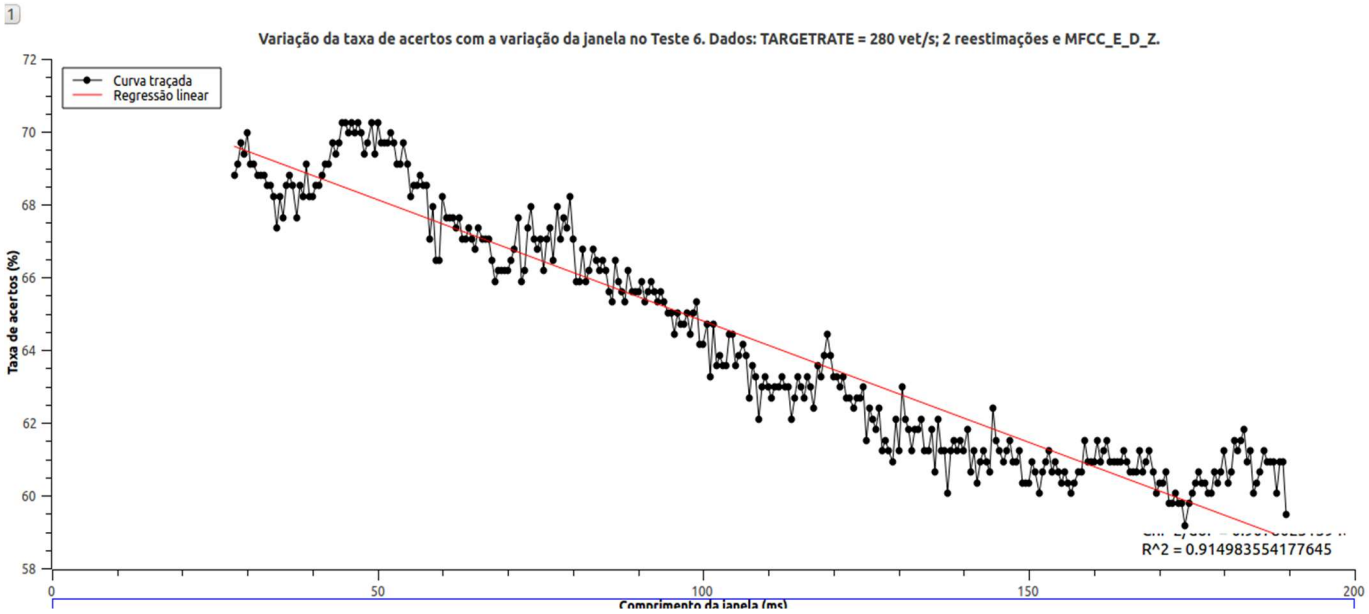

Figura 4.16 - Análise do percentual de acertos em função da variação da janela.

Nota-se pela Figura 4.16, mostrada acima e produzida a partir dos dados representados pela Tabela B.12, que, mais uma vez, existe um intervalo em que determinado comprimento de janela produz resultados melhores do que outros e quanto mais se distancia deste intervalo, menores são as chances de acerto. Neste

caso, o comprimento de janela ideal estaria em torno de 49 ms, maior do que no caso ideal do teste 0, quando se utilizava 36,5 ms.

Agora, utilizando-se tanto janela de 36,5 ms, quanto 49 ms, compara-se a taxa de acertos em função do número de reestimações com 280 vetores de parâmetros por segundo. Os resultados são apresentados na Figura 4.17, assim como na Tabela B.13.

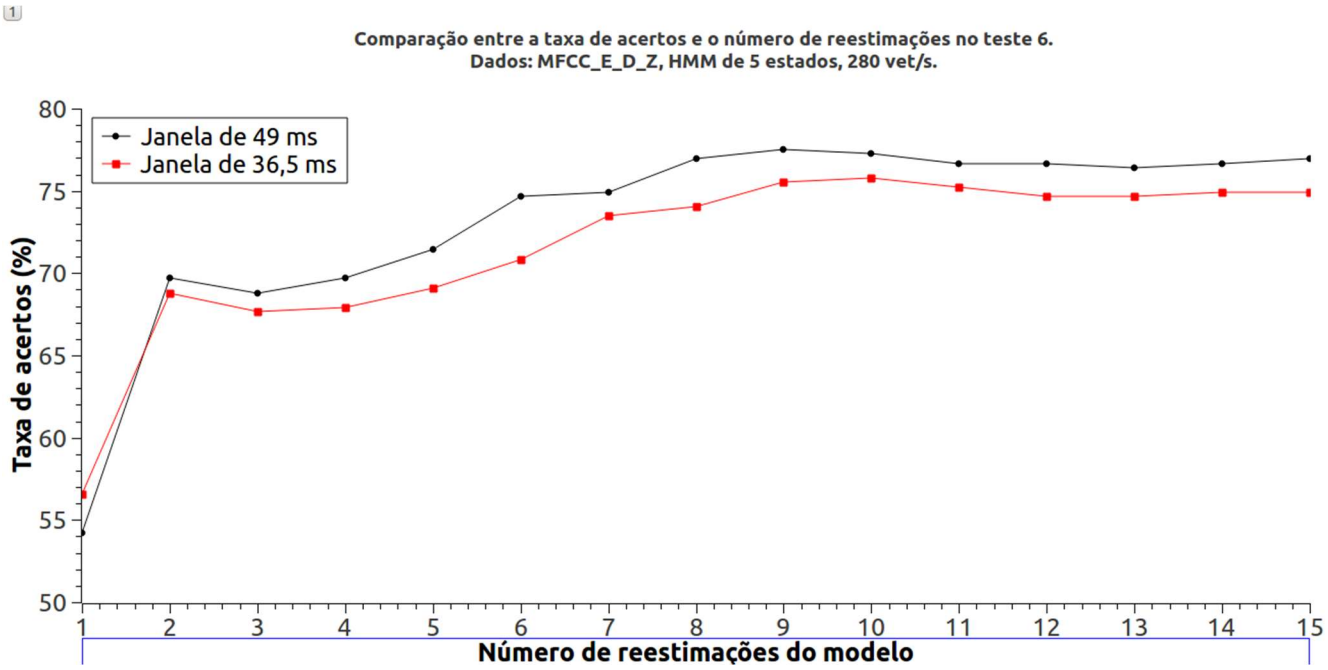

Figure 4.17 - Comparação entre a taxa de acertos e o número de reestimações no teste 6 para diferentes janelas.

Pode-se inferir que para este caso uma janela maior, de 49 ms é mais eficiente do que a janela utilizada em testes anteriormente. Isso leva a crer que cada treinamento, baseado nos arquivos de fala coletados, pode demandar uma configuração de parâmetros diferentes que se adaptam melhor para um caso específico em detrimento dos outros.

Neste caso, 9 reestimações foi o número ideal encontrado, demandando mais tempo de processamento que no teste 0, que necessitou de apenas 5 reestimações.

Para completar essa busca pelo conjunto de parâmetros ideal para o teste 6, analisa-se, então a variação da taxa de vetores de parâmetros por segundo, fixando-se o número de reestimações em 9 e o comprimento de janela em 49 m, foi possível observar os resultados mostrados pela Figura 4.18, mostrada adiante, assim como pela Tabela B.14 do apêndice B.

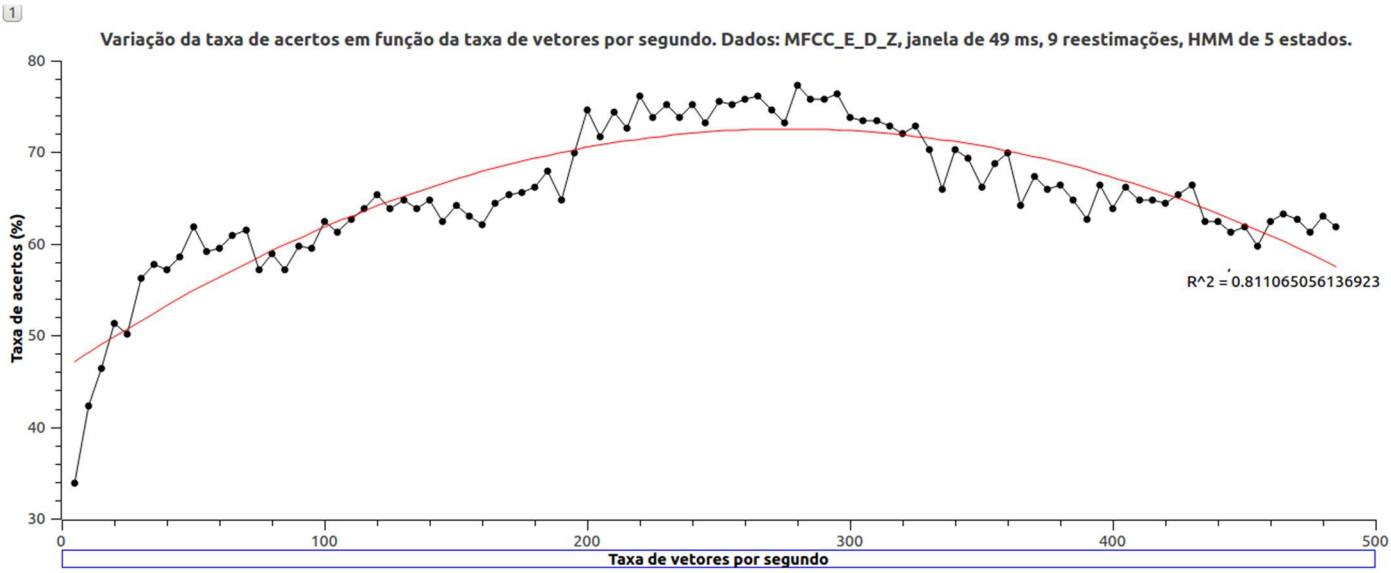

Figura 4.18 – Análise da variação da taxa de acertos em função da variação da taxa de vetores de parâmetros por segundo, teste 6, 9 reestimações, janela de 49 ms, “MFCC_E_D_Z”.

É possível novamente analisar que no caso da taxa de vetores de parâmetros por segundo, uma aproximação quadrática representa melhor o comportamento dessa curva, isto é, há um aumento da chance de acertos com o aumento dessa taxa, mas após um ponto de inflexão, a tendência é a piora dos resultados.

Uma vez fixados as características ideais para o teste 6, pode-se aplicar os mesmos valores de parâmetros para cada um dos outros 7 testes apresentados e discutir os efeitos dessa mudança no ponto de operação do reconhecedor de fala. A Figura 4.19, assim com a Tabela B15.

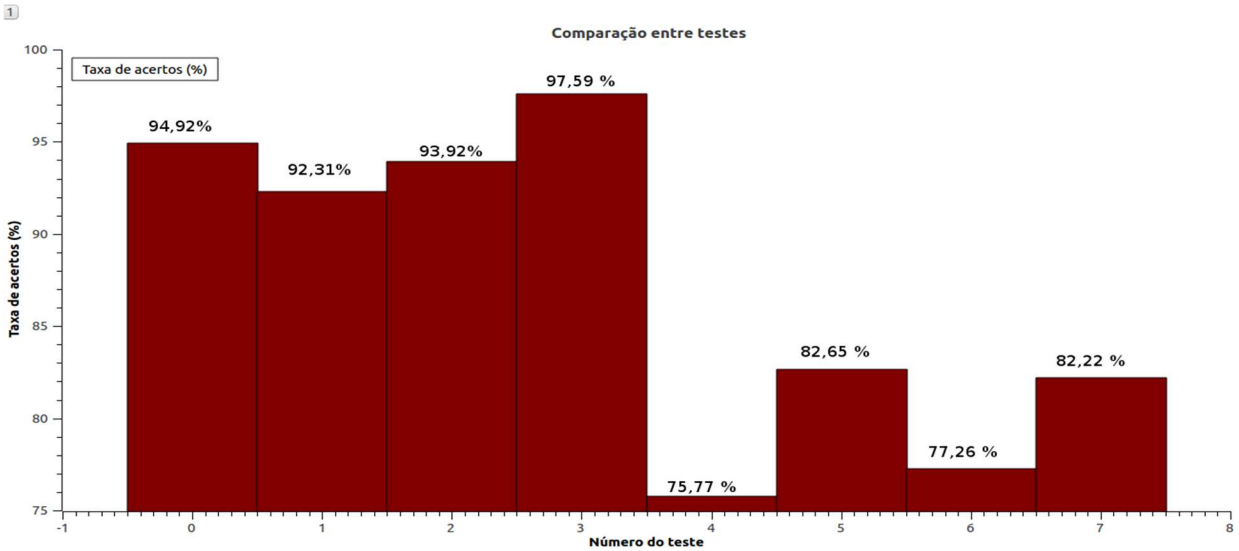

Figura 4.19 – Comparação entre testes utilizando o novo conjunto de parâmetros de modo a otimizar os acertos do teste 6.

O que se nota, a partir da Figura 4.19 (Tabela B.15) é que boa parte dos resultados melhoraram em relação à comparação feita pela Figura 4.10. O resultado de maior relevância é o do teste 6, que melhor representa o caso dependente de locutor. No resultado anterior, a taxa de acertos foi de 44,9 %, agora aumentou para 77,26 %, o que demonstra que a escolha correta do conjunto de parâmetros pode aumentar consideravelmente a taxa de acerto.

Dentre os erros cometidos durante o teste 7, onde apenas um locutor testa o reconhecedor de fala treinado pelos outros locutores, notou-se que não somente os erros diminuíram de 18 para 16, como no caso da fala lenta (que era o caso em que mais haviam erros) caíram de 6 para 2, enquanto no caso da fala rápida caíram de 5 para 4, o que leva a pensar que o aumento do comprimento da janela pode ter sido responsável por essa variação, assim como os outros parâmetros tem influência, uma janela maior consegue processar melhor trechos de fala lenta em detrimento da fala rápida.

Percebe-se também, fazendo uma comparação com o caso anterior que, os testes 0, 1, 2 pioraram seus resultados, mas ainda assim, continuam com taxas de acerto acima de 90 %. O resultado do teste 3 permaneceu inalterado e agora é o teste com melhor taxa de acertos. Os testes 4, 5, 6 e 7 melhoraram seu aproveitamento a partir das mudanças de parâmetros, embora, nenhum destes testes tenha atingido mais do que 85 % de acertos, o que leva a hipótese de que a codificação “MFCC_E_D_Z” é superior à codificação “MFCC_0_D_A” nos casos em que o número de locutores utilizados para o treinamento é reduzido.

Ainda assim, percebe-se que o número de locutores utilizados no treinamento é proporcional à qualidade do reconhecedor de fala.

5. CONCLUSÃO

O trabalho desenvolvido abordou uma análise para o reconhecimento de fala utilizando modelos ocultos de Markov gerados através da ferramenta HTK.

Algumas conclusões importantes puderam ser alcançadas, a principal, é que é sim possível utilizar um reconhecimento de fala de forma satisfatória por um locutor, independente do gênero e idade, que não tenha treinado a ferramenta. Isto vale para o caso em que o treinamento é desenvolvido por diversos tipos de vozes que consigam abranger uma grande gama de variações de suas características.

Uma forma eficiente de se treinar o reconhecedor de fala é utilizando um número parecido de locutores dos gêneros masculino e feminino. Quanto menos locutores forem utilizados durante o treinamento, menores serão as probabilidades de acerto.

Todo o estudo foi desenvolvido trabalhando-se com fala isolada, mas nada impede de expandir o sistema para um reconhecedor de frases em fala contínua. Para isso, ao invés de treinar o HTK utilizando um dígito de cada vez, pode-se utilizar frases completas, mas nesse caso o dicionário fonético e a regra gramatical devem ser mais elaborados.

As possibilidades de utilização da ferramenta são inúmeras. Para auxiliar futuros alunos que se interessem pelo tema, foi elaborado um tutorial explicativo de como foi desenvolvido o reconhecimento de fala descrito neste trabalho. As maiores dificuldades encontradas durante o período de desenvolvimento deste texto foram aprender a utilizar o HTK, uma vez que sua interface é feita apenas através de comandos no terminal e não há uma interface gráfica que seja amigável ao usuário e também a dificuldade de encontrar uma documentação que fosse explicativa e didática ao mesmo tempo. Os conceitos teóricos relativos ao reconhecimento de fala são, em sua maior parte, complexos para um estudante de engenharia, uma vez que envolvem muito conhecimento de ferramentas de processamento digital de sinais e probabilidade e estatística.

Outras conclusões relevantes são quanto aos parâmetros de configuração do HTK. Descobriu-se que o aumento da taxa de vetor de parâmetros por segundo pode aumentar a probabilidade de resultados corretos, mas com o aumento desse parâmetro, terá de ser usada uma janela de comprimento maior. Dependendo da

combinação destes dois parâmetros, o número de reestimações necessárias durante o treinamento poderá diminuir. Isto é, há um compromisso entre diversos parâmetros dentro do HTK visando encontrar o ponto ótimo do reconhecimento de fala.

A velocidade de fala é também muito importante dentro desse estudo. A probabilidade de encontrar os fones corretos depende da velocidade em que o sistema de reconhecimento é treinado, assim como a velocidade de fala utilizada durante os testes. Isso também está ligado ao conceito de enjanelamento utilizado durante a codificação dos vetores de parâmetros a partir dos dados de áudio.

A importância deste trabalho é dada quando se busca entender como modelar corretamente um reconhecedor de fala, de forma a atingir as especificações desejadas e todos estes temas abordados não são apenas úteis para pesquisadores, mas em diversas áreas da engenharia, podendo ser útil inclusive no desenvolvimento de sistemas voltados à acessibilidade de pessoas com deficiência visual e motora. Também há aplicação dentro de indústrias, onde podem-se desenvolver mecanismos comandados por voz que não necessitem de treinamento de cada locutor que os utilizará e possa substituir comandos por teclado ou botoeiras.

Algumas contribuições podem ser citadas, como o fato de poder se desenvolver reconhecedores de fala robustos o suficiente para aplicações pequenas, mas diversas, como acionamentos por voz, utilizando-se um vocabulário pequeno e poucos minutos de gravação. E isto é possível ajustando-se os parâmetros de configuração do HTK para se criar um modelo que é pouco susceptível a erros.

Outra contribuição relevante e interessante é a confirmação de que um único locutor treinando o modelo não produz resultados tão bons quando utilizado por outros locutores que o testam. Embora seja possível afirmar que o ajuste das variáveis do HTK ajuda a maximizar as chances de acerto neste caso.

Por fim, pode-se dizer que este estudo foi importante para desenvolver novos conhecimentos que serão úteis em projetos futuros e também foi importante não somente naquilo que é proposto, mas auxiliou o autor a desenvolver melhor as capacidades de expressão oral e escrita, organização de ideias, informática e programação de *scripts*. Por isso, é possível dizer que foi fundamental para um aperfeiçoamento pessoal e profissional.

REFERÊNCIAS

[1] YNOGUTI, Carlos Alberto. **Reconhecimento de fala contínua usando modelos ocultos de Markov**. Carlos Alberto Ynoguti. Campinas, SP, 1999.

[2] YNOGUTI, Carlos Alberto, pag. 16, 1999 *apud* RUDNICKY, A. I., HAUPTMANN, A. G., and LEE, K. F. **Survey of Current Speech Technology**. Disponível em: < http://www.lti.cs.cmu.edu/Research/cmt-tech-reports.html >. Acesso em 07 jul. 2014.

[3] YOMA, Néstor Becerra. **Reconhecimento automático de palavras isoladas: estudo e aplicação dos métodos determinístico e estocástico.** Campinas, SP, 1993 *apud* HUANG, X. D. ARIKI, Y. Jack, M. A. **Hidden Markov Models for Speech Recognition**. Edinburgh University Press, 1990 *et* MOORE, R. **State of the Art in Speech**. Workshop in Integrating Speeching and Natural Language. University College Dublin, julho de 1992.

[4] YOMA, N. B., pag. 1993 *apud* LEE, Kai-Fu. **Large Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System**. Tese de PhD, Department of Computer Science, Carnegie Mellon University, 1988.

[5] AULETE, Dicionário Digital. **Definição da palavra coarticulação**. Disponível em < http://www.aulete.com.br/coarticula%C3%A7%C3%A3o >. Acesso em : 25 out. 2016, 15:49

[6] YOMA, Néstor Becerra. **Reconhecimento automático de palavras isoladas: estudo e aplicação dos métodos determinístico e estocástico.** Campinas, SP, 1993.

[7] YOUNG, Steve *et al*. **The HTK Book (for HTK Version 3.4)**. Cambridge University Engineering Department. Cambridge, 2006.

[8] KELLY, ANA. **O espectrograma ajudando sua comunicação**. [S.l.]: Falarte, 2014. Disponível em: < http://falarte.com.br/?p=437 >. Acesso em: 02 nov. 2015, 14:19.

[9] LAPS - UFPA. **Produção de fala**. Belém – PA, 2008 . Disponível em: < http://laps.ufpa.br/diego/producao_2.pdf >. Acesso em: 04 nov. 2015, 14:35.

[10] MAASSEN, Ben. **Issues Contrasting Adult Acquired Versus Developmental Apraxia of Speech.** Semin Speech Lang, nov. 2002; 23(4):257-66. Disponível em: < speech-language-therapy.com/pdf/maassen2002.pdf >. Acesso em: 26 out. 2016, 16:21.

[11] AMPLIFON. **Como funciona o sistema auditivo**. Amplifon, 2013. Disponível em: < http://www.amplifon.pt/A-audi%C3%A7%C3%A3o/Como-funciona-o-sistema-auditivo/Pages/default.aspx >. Acesso em: 03 nov. 2015, 15:08.

[12] DIREITO DE OUVIR. **Como funciona o timpano no sistema auditivo**. [S.l.]. Amplifon, 2013. Disponível em: < http://www.direitodeouvir.com.br/como-funciona-o-timpano-no-sistema-auditivo/ >. Acesso em: 04 nov. 2015, 15:35.

[13] VILELA, Ana Luisa Miranda. **Percepção da Força Gravitacional e do Movimento**. Site Anatomia e Fisiologia Humanas mantido pela Profª Ana Luisa Miranda Vilela. [S.l.]: Biolo ja materiais didáticos . Disponível em: < http://www.afh.bio.br/sentidos/sentidos5.asp > Acesso em: 04 nov. 2015, 12:14.

[14] MOKRANE, Asma. **Le fonctionnement du système auditif**. [S.l.]: Lobe, 2015. Disponível em: < http://lobe.ca/audition-langage-et-parole/le-fonctionnement-de-laudition-audition-langage-et-parole/le-fonctionnement-du-systeme-auditif/le-fonctionnement-de-loreille/ >. Acesso em: 01 nov. 2015, 13:29.

[15] JNA ASSOCIATION. **Le son et le système auditif**. [S.l.]: Journée Audition, 2012. Disponível em: < http://www.journee-audition.org/l-audition/le-son-et-le-systeme-auditif.html >. Acesso em: 01 nov. 2015, 15:15.

[16] VILELA, Ana Luisa Miranda. **O Mecanismo da audição**. Site Anatomia e Fisiologia Humanas mantido pela Profª Ana Luisa Miranda Vilela. [S.l.]: Biolo ja materiais didáticos . Disponível em: < http://www.afh.bio.br/sentidos/sentidos4.asp > Acesso em: 24 out. 2016, 17:54.

[17] OUTILS RECHERCHE. **Le système auditif humain**. [S.l.]: Outils Recherche. Disponível em: < http://outilsrecherche.over-blog.com/pages/Notes_111_Le_Systeme_Auditif_Humain-3080878.html >. Acesso em: 01 nov. 2015, 15:27.

[18] STEVENS, Stanley Smith; VOLKMANN, John & NEWMAN, Edwin B. **A scale for the measurement of the psychological magnitude pitch**. Journal of the Acoustical Society of America 8 (3): páginas 185–190. 1937. Bibcode:1937ASAJ....8..185S. doi:10.1121/1.1915893.

[19] O'SHAUGHNESSY Douglas. **Speech communication: human and machine**. Addison-Wesley. p. 150. ISBN 978-0-201-16520-3. 1987.

[20] TAYLOR, Paul. **Text-to-Speech Synthesis**. Capítulo 12. Páginas 360-365. University of Cambridge. Cambridge University Press. 2009. Cambridge, Reino Unido.

[21] RABINER, Lawrence R.; SCHAFER, Ronald W. **Introduction to Digital Speech Processing**. Capítulo 5. Now Publishers Inc. ISBN 1601980701, 9781601980700. 2007.

[22] NATIONAL INSTRUMENTS. **FFTs e janelamento**. Disponível em : < http://www.ni.com/white-paper/4844/pt/ >. Acesso em: 25 out. 2016, 17h57.

[23] RUDOLPH, D. **FM Pre-emphasis and De-Emphasis**. Radio Museum, Janeiro de 2011. Disponível em: < http://www.radiomuseum.org/forum/fm_pre_emphasis_and_de_emphasis.html > Acesso em: 24 out. 2016, 13h21.

[24] JANG, Jyh-Shing Roger. **Audio Signal Processing and Recognition**. CS Dept., National Taiwan University, Taiwan. Disponível em : < http://mirlab.org/jang/books/audiosignalprocessing/speechFeatureMfcc.asp?title=12-2%20MFCC >. Acesso em: 23 nov. 2015, 14h02.

[25] GAO, Yu-Qing; CHEN, Yong-Bin; WU, Bo-Xiu. **Dynamic Adaptation of Hidden Markov Model for Robust Speech Recognition**. Laboratory of Pattern Recognition, Institute of Automation, Academy of Sinica. Pequim, China, 1989.

[26] MITRA, Vikramjit; FRANCO Horacio *et al*. **Normalized amplitude modulation features for large vocabulary noise-robust speech recognition**. Speech Technology and Research Laboratory, SRI International, Menlo Park, Califórnia, EUA.

[27] ALEXANDRIDIS, Anastasios; GRIFFIN, Anthony e MOUCHTARIS, Athanasios. **Breaking down the cocktail party: capturing and isolating sources in a soundscape**. University of Crete, Department of Computer Science, Heraklion, Creta, Grécia, 2014.

[28] YU Kai; XU Boling; DAI Mingyang *et al*. **Suppressing Cocktail Party Noise for Speech Acquisition**. National Key Laboratory of Modern Acoustics, Institute of Acoustics, Nanjing University, Nanjing, China, 2000.

[29] ANSARI, Zohreh e SALEHI, Seyyed Ali Seyyed. **Proposing Two Speaker Adaptation Methods for Deep Neural Network based Speech Recognition Systems**. Biomedical Engineering dept. Amirkabir University of Technology, Teerã, Irã, 2014.

[30] MITRA, Vikramjit; SIVARAMAN Ganesh *et al*. **Articulatory features from Deep Neural Networks and their role in Speech recognition**. Speech Technology and Research Laboratory, SRI International, Menlo Park, Califórnia, Estados Unidos, 2014.

[31] RABINER, Lawrence R. *et al*. **A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition**. Proceedings of the IEEE, Vol. 77, No. 2, Fevereiro de 1989, Fellow, IEEE.

[32] BERKELEY UNIVERSITY. Speech Signal Processing. Acervo Digital. Berkeley University. Disponível em: < http://www1.icsi.berkeley.edu/Speech/docs/HTKBook/node58.html > Acesso em: 29 out. 2016, 17:19.

APÊNDICE A – TUTORIAL PRODUZIDO PARA UTILIZAÇÃO DO HTK

A.1 INTRODUÇÃO

Durante um bom tempo houveram dificuldades para compreender como funciona o *HTK Toolkit*, ferramenta utilizada em muitas aplicações e que é baseada em HMMs (*Hidden Markov Models* ou Modelos Ocultos de Markov) para o reconhecimento de padrões.

A documentação do programa é vasta, é disponibilizado um manual de mais de 300 páginas, assim como há diversos tutoriais em diversos lugares da internet, basta digitar na ferramenta de busca: "*HTK Tutorial*" e inúmeras opções aparecerão. A questão principal é que, para quem nunca se embrenhou nessa ferramenta é bastante complicado entender como desenvolver todo o procedimento de reconhecimento visto que a ferramenta não é tão amigável em um primeiro momento.

O objetivo desse documento é servir como um tutorial em português brasileiro para explicar passo a passo como desenvolver um reconhecedor de fala que consiga reconhecer dígitos isoladamente. Entendendo bem esse caso mais simples seria possível desenvolver um reconhecimento de fala contínua com determinadas frases ou até casos mais complexos.

Também será fornecido um script ao final do tutorial desenvolvido pelo autor desse tutorial que tem como objetivo "automatizar" o procedimento de reconhecimento de fala acelerando a coleta dos resultados e visualização dos dados.

Durante todo o desenvolvimento do tutorial é necessário lembrar que foi utilizado *Ubuntu* 16.04 LTS, dependendo da versão do sistema operacional, os comandos podem não ser os mesmos.

A.2 INSTALAÇÃO DO HTK

O primeiro passo para se utilizar o HTK obviamente é instalá-lo. Este software foi desenvolvido pelo departamento de engenharia da Universidade de Cambridge na

Inglaterra. Ele é disponibilizado gratuitamente, assim como toda sua documentação pelos desenvolvedores na página:

http://htk.eng.cam.ac.uk/

Para baixá-lo é necessário ter um registro no site, basta clicar em *Register* no topo da página e a seguinte página se abrirá:

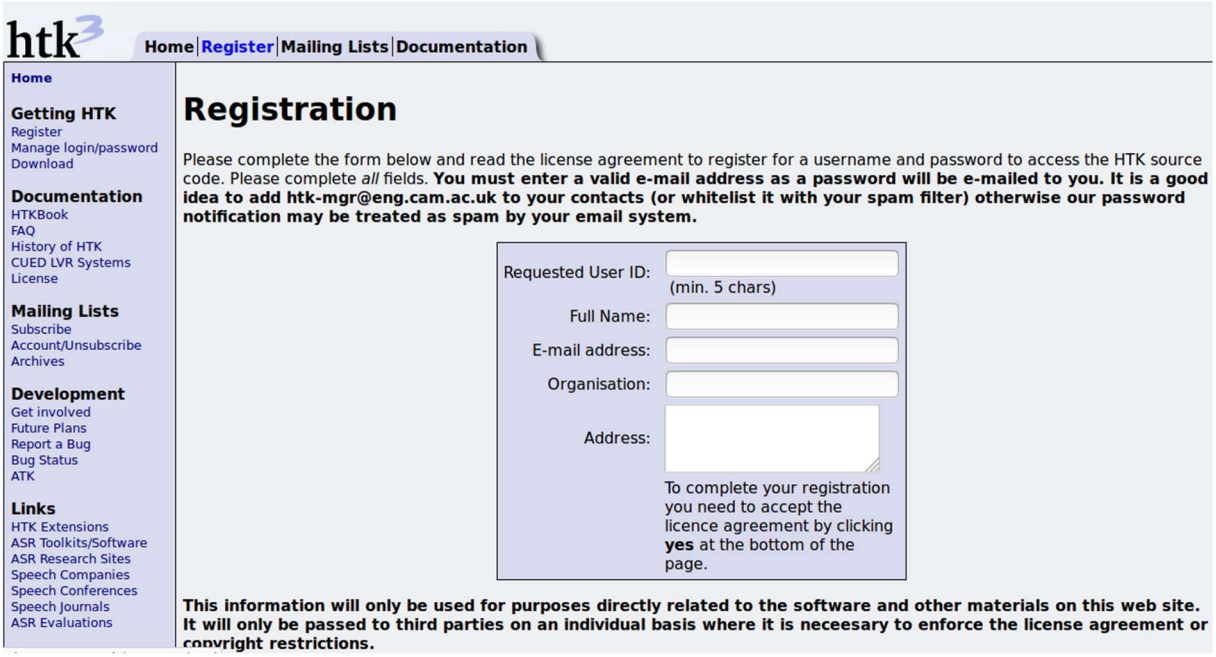

Figura A.1 - Registro para download do pacote necessário.

Nesta página, basta completar os campos requeridos, descer até o final da página e aceitar os termos de licença clicando em *Yes*.

Então um e-mail será enviado para a caixa de entrada registrada no *site* e no conteúdo deste e-mail estará a senha de acesso aos *downloads* disponíveis.

Após anotar a senha em algum lugar, vá ao site do HTK novamente e clique no campo *Download*, na parte superior esquerda da página.

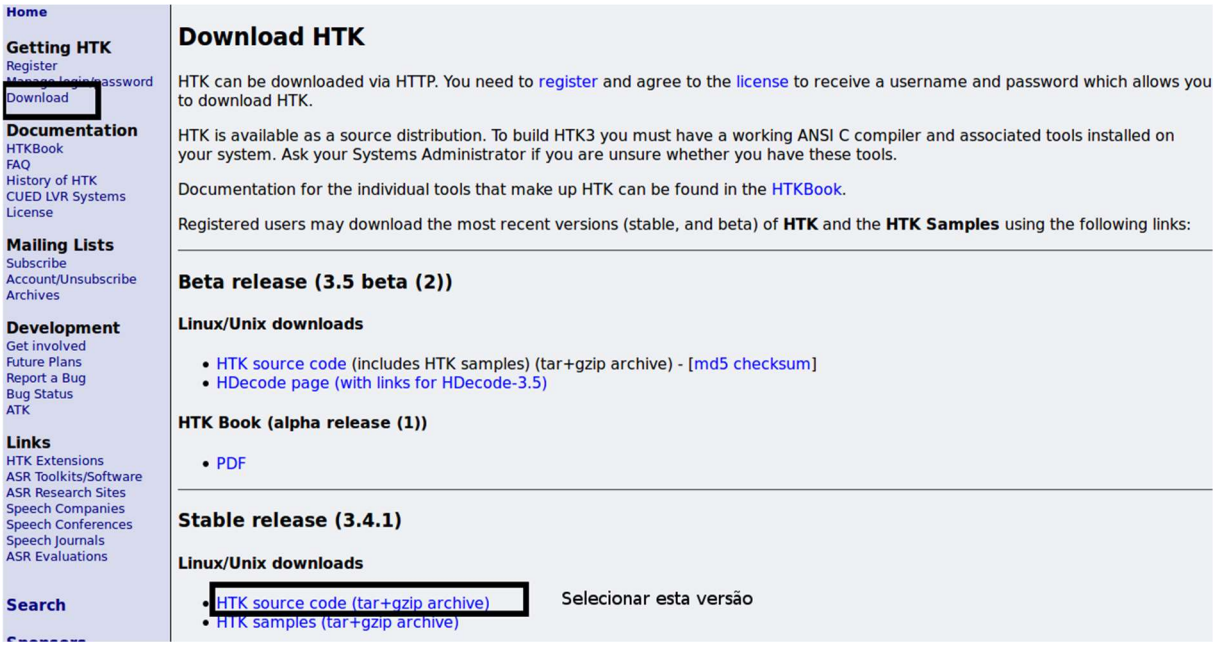

Figura A.2 - Escolhendo a versão de download do HTK.

Aparecerão nesta página algumas opções de pacotes para baixar. É altamente recomendado escolher a versão *Stable release (3.4.1).* Os desenvolvedores também disponibilizam versões *Beta*, que normalmente ainda estão sendo testadas e é possível encontrar problemas de compilação que dificultarão a utilização da ferramenta mais tarde. As versões estáveis (que já foram testadas e aprovadas) normalmente são disponibilizadas abaixo das versões *Beta*.

O pacote *HTK samples* nada mais é que um pacote com alguns arquivos extras, incluindo um tutorial e demo da utilização do HTK, este demo é interessante, mas para quem está começando a utilizar as ferramentas agora será muito difícil entender o desenvolvimento desse reconhecimento utilizando apenas esse exemplo disponibilizado na página.

Assim que clicar em *HTK Source code (tar + gzip archive)* será demandado usuário e senha que foram registrados, lembrando que a senha foi mandada para o e-mail escolhido, ou seja, foram eles que definiram. Se o usuário e senha estiverem corretos, o *download* do pacote começará instantaneamente.

Uma vez terminado de baixar o pacote (que nesse caso virá no formato “tar.gz”) será necessário extrair o conteúdo compactado para uma pasta em seu computador.

Para extrair um arquivo tar.gz siga as seguintes instruções:

Abra um terminal no Ubuntu (atalho: Ctrl+Alt+T);

Navegue até o diretório onde está o arquivo baixado utilizando os comandos:

cd [Nome da pasta]	(Se deslocará para o diretório da pasta selecionada);
ls	(Lista todo o conteúdo do diretório atual);
cd	(Volta para o diretório raiz (home))
cd ..	(Retorna ao diretório imediatamente anterior).

Ao encontrar o diretório onde está o arquivo compactado digite o seguinte comando:

```
tar –zxvf [Nome do arquivo].tar.gz
```

Isso fará com que uma nova pasta seja criada no mesmo diretório deste arquivo, contendo o conteúdo descompactado. Normalmente o nome desta pasta será htk. Não feche o terminal ainda.

Uma vez feito isso, pode-se entrar neste novo diretório utilizando o comando:

```
cd [Nome do novo diretório]
```

ou

```
cd htk
```

É a partir deste diretório que o HTK será instalado no computador.

Primeiramente é necessária configurar a instalação do HTK. No terminal, dentro da pasta htk, digite:

```
./configure
```

Isso executará alguns comandos de configuração. Quando terminar, essa mensagem deverá aparecer:

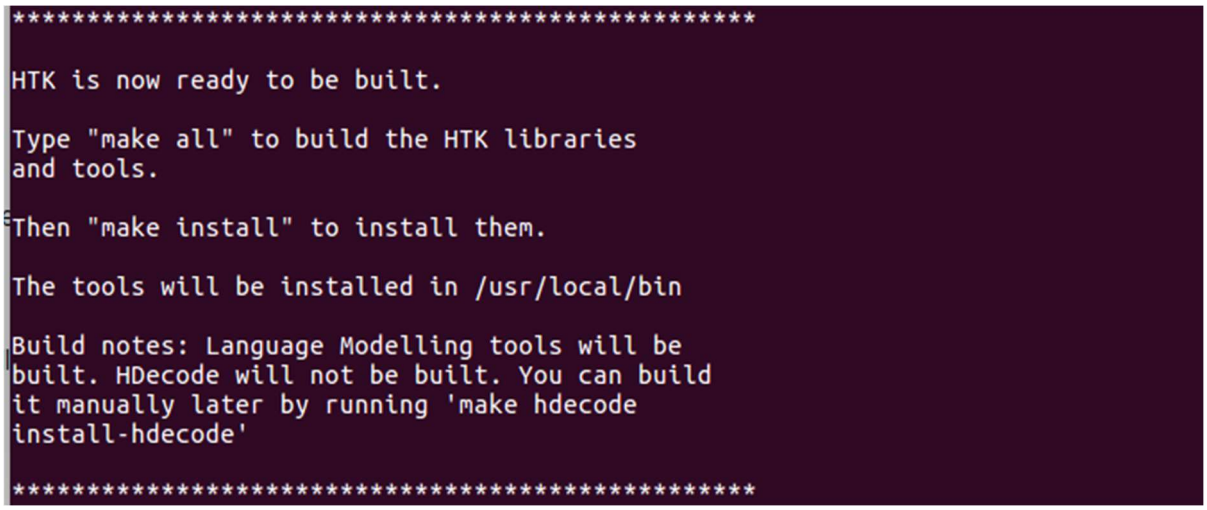

Figura A.3 - Mensagem de configuração completa no terminal.

Agora, digite no terminal:

make all

Essa etapa deverá tomar alguns minutos e vai preparar o HTK para a instalação. Observação: Algo muito comum de acontecer é um erro do tipo:

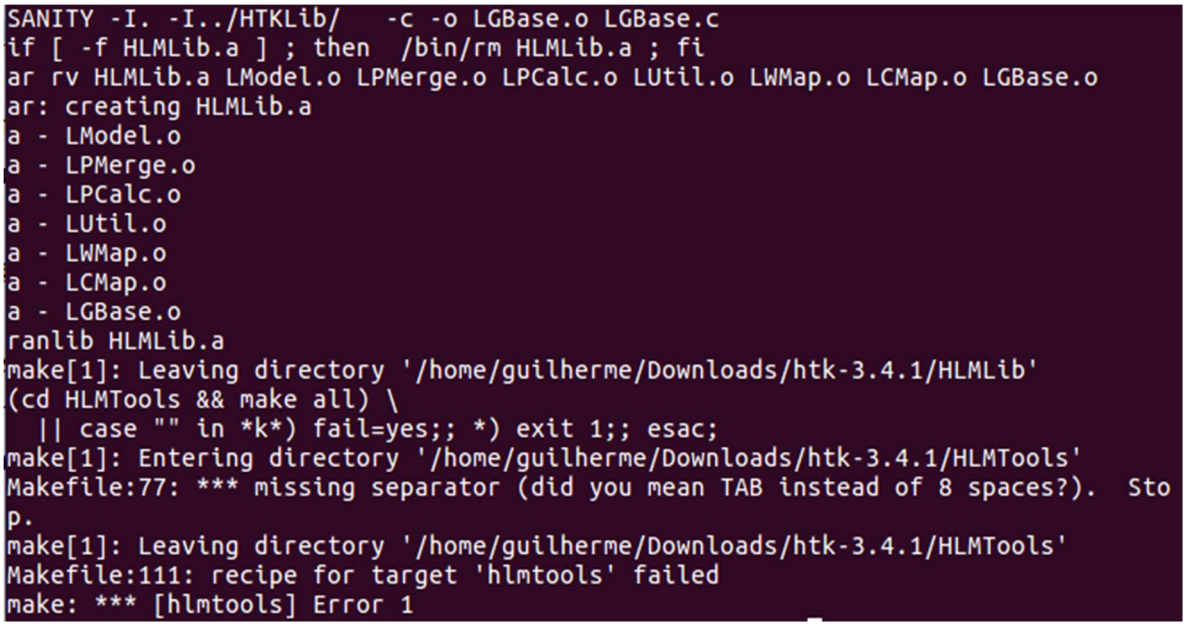

Figura A.4 - Erro de compilação do HTK.

Ao se deparar com esse erro, pode parecer algo complicado, mas na verdade é simples. Quando disponibilizaram o pacote para *download*, dentro da pasta

HLMTools, o arquivo *Makefile* está com uma falha. Use o editor de textos de sua preferência e abra o arquivo *Makefile* dentro da pasta *HLMTools*. Na linha 77:

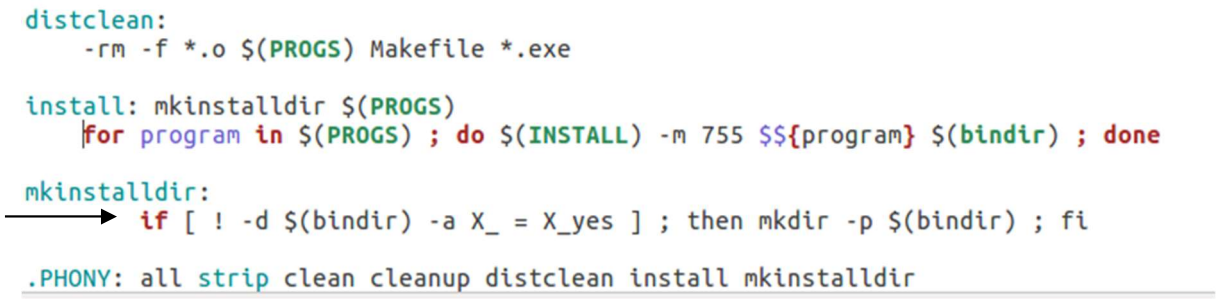

Figura A.5 - Erro encontrado no arquivo de instalação.

Exatamente antes do *if*, há 8 espaços onde deveria haver 2 tabulações. Na hora de compilar o sistema vai entender como um erro, e uma falha na instalação acontecerá. Apague todos os espaços e no lugar aperte TAB duas vezes. Então salve o arquivo e use "*make all*" no terminal mais uma vez.

Se tudo correr bem e nenhum erro aparecer no terminal, digite então:

```
sudo make install
```

A permissão será necessária, então digite a senha de super-usuário. Após isso, a instalação começará. Normalmente alguns segundos depois a instalação já estará completa.

Algo nítido após o procedimento de instalação é que não haverá nenhum ícone na área de trabalho, barras laterais ou alguma forma de executável. O HTK se quer tem uma interface gráfica com o usuário. São várias ferramentas independentes que são instaladas e para utilizá-las é normalmente através do terminal. Cada comando tem inúmeras opções e necessitam determinados argumentos. Para entender mais sobre como funciona cada ferramenta individual, pode-se baixar o HTK Book, manual de instruções do HTK na mesma página de *download* do pacote de instalação.

Ao terminar a instalação é como se uma série de "comandos de terminal" tivessem sido disponibilizados, mas para executar as tarefas de reconhecimento de fala não serão necessários todos eles, apenas uma parte, dependendo do que se está

se desenvolvendo. Por exemplo, se digitar: HVite no terminal (em qualquer diretório), algo assim deverá aparecer:

```
USAGE: HVite [options] VocabFile HMMList DataFiles...

 Option                                  Default

 -a      align from label files          off
 -b s    def s as utterance boundary word    none
 -c f    tied mixture pruning threshold      10.0
 -d s    dir to find hmm definitions         current
 -e      save direct audio rec output        off
 -f      output full state alignment         off
[…]
```

Que na verdade são as opções de configuração dessa ferramenta, caso não encontre esse comando é porque algo não correu bem durante a instalação.

A.3 DESENVOLVENDO A GRAMÁTICA E COLETANDO DADOS

Uma vez instalado o software, é hora de colocar a mão na massa. Para o desenvolvimento do reconhecimento de fala deste documento, utilizou-se como exemplo o reconhecedor de locutor dado pelo *link* a seguir:

http://www.computing.dcu.ie/~john/SpeakerRecSimple.pdf

Nesse exemplo, o autor explica como construir um sistema que reconhece entre dois locutores falando a vogal “a”, um homem e uma mulher. Um outro exemplo que segue a mesma linha de pensamento é disponibilizado no repositório do GitHub a seguir:

https://github.com/jnyryan/htk-speaker-recognition

Neste caso, há um treinamento para diferenciar entre locutor e não-locutor, ou seja, tentar saber se a pessoa que está falando é quem treinou o programa ou não.

Bom, iniciando o desenvolvimento do reconhecedor, crie uma pasta especial com o nome que desejar e dentro dela crie as seguintes pastas:

- *Configs:* Onde serão mantidos os arquivos de configuração para alguns comandos.
- Data: Onde serão mantidos os dados de voz de cada locutor. Dentro da pasta *Data* crie mais duas pastas: *train* e *test*, uma será utilizada para os arquivos de treinamento e outra para os arquivos de teste, respectivamente.
- *Dictionary* e uma sub-pasta chamada *Src*: Onde serão armazenados os arquivos de construção do dicionário e da gramática utilizados no reconhecedor.
- *Labels:* Onde ficarão os arquivos com as etiquetas correspondentes a cada fone dito.
- *Models:* Onde serão colocados os arquivos referentes aos modelos ocultos de Markov (*hmms*) necessários ao trabalho de reconhecimento.
- *Results:* Onde serão gerados os resultados para posterior análise e comparação.

Essa nomenclatura e essas pastas foram definidas simplesmente por organização e padrão, mas na verdade o reconhecimento pode perfeitamente ser realizado sem criar todas essas pastas, talvez apenas na pasta original, isso depende de cada utilizador.

Uma vez terminada a criação desses diretórios, é hora de fazer a primeira tarefa necessária ao reconhecimento. A criação de uma “regra gramatical” para o reconhecedor. A regra gramatical é um arquivo de texto qualquer que dirá ao HTK como ele deverá tentar encontrar cada unidade fonética. Alguns exemplos são disponibilizados no manual do HTK ou na *internet*:

http://www.ee.columbia.edu/ln/LabROSA/doc/HTKBook21/node131.html

O *link* acima dá uma explicação de como funciona a regra gramatical e como criar isso para diferentes casos, por exemplo:

```
$digit = one | two | three | four | five |
        six | seven | eight | nine | zero;
(
   sil < $digit > sil
)
```

Significa que a variável *digit* pode assumir qualquer valor compreendido entre *one* e *zero*, a barra vertical (|) simboliza o operador lógico OU.

O reconhecedor terá de reconhecer um silêncio (sil), em seguida um ou mais dígitos da lista e terminará a sentença com outro silêncio. Os símbolos < e > representam um laço, ou seja, a variável *digit* pode assumir mais de um valor, nesse caso, dizemos que esse reconhecedor pode trabalhar com fala contínua.

No caso deste tutorial, abra um editor de texto qualquer e escreva o seguinte texto:

```
$digit = um | dois | tres | quatro | cinco | seis | sete | oito | nove | zero;
( SENT-START ( $digit ) SENT-END )
```

Ou seja, o valor da variável *digit* pode assumir os mesmos valores, mas dessa vez em português. O SENT-START e SENT-END são similares ao “sil”, mas nesse caso não há os símbolos < e > na gramática, ou seja, ele vai esperar e tentar reconhecer apenas um dígito para cada arquivo de teste. Nesse caso a fala é isolada. Normalmente o reconhecimento de fala isolada é mais simples do que em fala contínua, mas o que diferencia os dois é apenas a forma como o usuário treina o HTK e como ele define a gramática.

Salve esse arquivo com o nome de *grammar* (a extensão é opcional) dentro da pasta Dictionary/Src.

A partir desse arquivo criado será necessário gerar um novo arquivo que dirá ao HTK a transcrição da regra gramatical. Para isso utiliza-se o comando HParse.

Usando o terminal, entre no diretório ou informe o diretório do arquivo *grammar* ao HParse e a saída será um arquivo chamado *wdnet*. Certifique-se de apontar o mesmo diretório para a saída:

HParse Dictionary/Src/grammar Dictionary/Src/wdnet

Após gerado esse arquivo, se ainda não tiver as gravações de áudio, então agora é o momento de adquiri-las. Embora o HTK possua uma ferramenta para gravação do áudio, ela é pouco intuitiva, então recomenda-se o uso do *Audacity (sudo apt-get install audacity)* para essa função. Em alguns casos, o resultado da gravação acaba um pouco travado dependendo da versão do problema, também é possível usar o *audio-recorder (sudo apt-get install audio-recorder)* no *Ubuntu* para a mesma função. Inicialmente é possível gravar em qualquer formato e taxa de amostragem, mas algumas edições podem ser feitas após.

É importante gravar várias vezes o mesmo dígito, quanto mais informação disponibilizar o HTK para modelar os modelos ocultos de Markov, melhor será a taxa de acertos.

Salve cada dígito em um áudio separado, certificando-se que no início do dígito há um tempo de “silêncio”, caso não tenha, pode adicionar silêncio artificialmente com o *Audacity* (*Generate* > *Silence*...). É importante fazer algumas conversões nessa etapa. O HTK trabalha com diversas taxas de amostragem, mas o mais comum é 16 kHz. Normalmente os áudios gravados são amostrados em 44,1 kHz ou 48 kHz.

O que é necessário é os converter para uma taxa de amostragem inferior (de 44,1 para 16 kHz), isso pode ser feito utilizando-se do software *Audacity*. Basta importar o arquivo de áudio original (ou o gravado) no software (*Ctrl+Shift+I*), selecionar o áudio (*Ctrl+A*). Clicar em *Project Rate* (Hz) (no canto inferior esquerdo da tela), selecionar 16000 e então exportar como um arquivo .*wav* de 16 bits (Ctrl+Shift+E).

É importante lembrar de transformar todas as gravações *Stereo* em *Mono*, o próprio *Audacity* tem essa função. Clique em *Tracks* > *Track Stereo* to *Mono*.

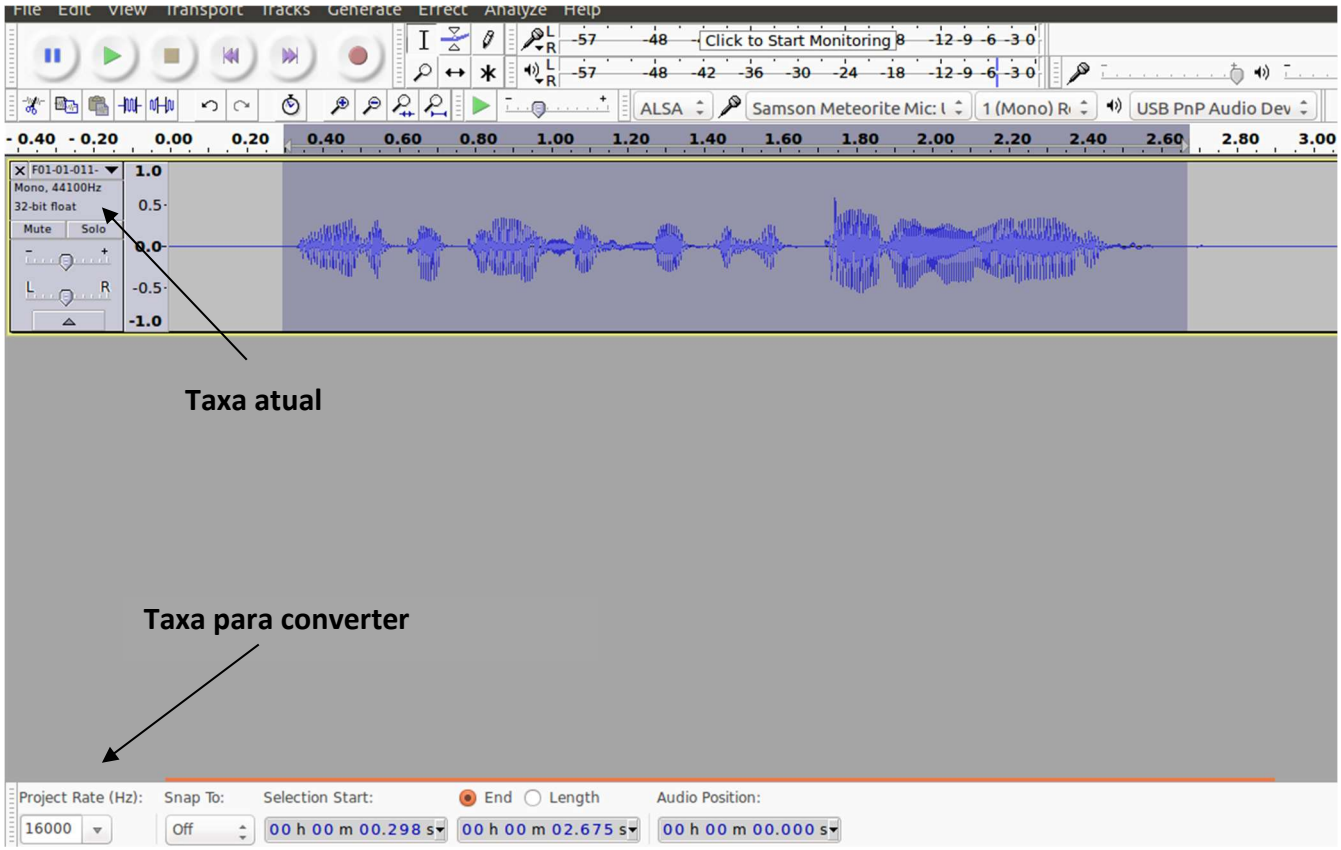

Figura A.6 - Convertendo taxa de amostragem do áudio no *Audacity*.

Se perceber na Figura A.6, representada acima, onde está apontando a taxa atual 44100 Hz, na linha imediatamente inferior está escrito: *32-bit float* dá a impressão que o áudio será salvo em *32-bit* e não em *16-bit*, mas isso não é possível, é apenas um padrão do HTK representar o formato, no momento de exportar o arquivo (*Export...* ou Ctrl+Shift+E) ele será salvo em *16-bit*.

Uma vez registrados todos os áudios, recomenda-se separar os áudios de treinamento dos áudios de teste, para isso foram criadas as duas pastas: *Data/train* e *Data/test*, a separação vai da vontade do usuário, também é possível utilizar todos para treinar e mais tarde fazer novas aquisições de áudio para teste. O ideal é não usar os mesmos de treino para teste, pode até ser o mesmo usuário, mas não os mesmos áudios porque o objetivo é fazer o HTK reconhecer um padrão de teste a partir de um padrão de treino.

A.4 PARAMETRIZAÇÃO

Essa etapa é necessária para obter informações relevantes a partir da análise dos dados de fala. Embora o HTK aceite diversos formatos de áudio, nessa etapa deve-se criar um padrão, que são os arquivos *.mfc* e estão relacionados aos coeficientes de frequência *mel-cepstra*l (MFCC).

Para esta tarefa já há um *script* pronto e a ferramenta *HCopy* do HTK tem exatamente a função de fazer essa conversão.

Abra o editor de textos, cria algo do tipo:

```
# Coding parameters
SOURCEKIND     = WAVEFORM
SOURCEFORMAT   = WAV
SOURCERATE     = 625
TARGETKIND = MFCC_0_D_A
TARGETRATE = 280000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 365000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = T
```

Caso tenha utilizado uma taxa de amostragem do áudio em 16 kHz, o parâmetro *SOURCERATE* é calculado da seguinte forma:

$$SOURCERATE = \frac{1 \cdot 10^7}{f_s}$$

Onde f_s é a frequência de amostragem do sinal em Hz. Caso utilize 16 kHz, então *SOURCERATE* = 625.

O parâmetro *TARGETKIND* é a forma de *MFCC* que o programa irá utilizar para converter o áudio, existem várias formas diferentes de se definir uma *MFCC*. Não é necessário para este tutorial entrar em detalhes sobre cada um dos tipos, utilizando "*MFCC_0_D_A"* é possível obter os resultados necessários. Mais informações podem ser obtidas na documentação do HTK.

Além desses, há outros dois parâmetros fundamentais à identificação: *WINDOWSIZE*, o comprimento da janela utilizada na conversão (vale lembrar que a unidade não é segundos, mas sim 10^{-7} segundos) e *TARGETRATE*, o tamanho (ou a taxa) de vetores de informação utilizados por segundo. Ambos influenciam no comportamento do reconhecimento e nos resultados. Cada reconhecedor tem uma taxa ideal, isso será discutido mais adiante. Por enquanto, basta conhecer o funcionamento através da documentação a seguir:

http://www1.icsi.berkeley.edu/Speech/docs/HTKBook/node58.html

O parâmetro *SOURCEKIND* revela a forma como serão obtidos os dados, no caso através de formas de onda (*WAVEFORM*) previamente gravadas em formato *WAV* (*SOURCEFORMAT*). Caso queira utilizar coeficientes de predição no lugar de um arquivo de áudio *SOURCEKIND*, troque *WAVEFORM* por LPC e se quiser que ele defina o tipo de fonte de entrada a partir do arquivo de entrada utilize *ANON*.

Sempre que usar reconhecimento *offline*, ou seja, com arquivos já salvos anteriormente ao reconhecimento, deixe o parâmetro *ENORMALIZE* em *TRUE* (T), esse parâmetro representa uma normalização da energia na conversão.

Os outros parâmetros podem ser deixados como está, é um padrão utilizado.

Salve este arquivo com o nome *config_wav2mfc* dentro da pasta *Configs*.

Agora será necessário dizer à ferramenta quais serão os arquivos a serem convertidos e sua localização em relação ao lugar onde está rodando o comando.

Digamos que nomeou os arquivos de áudio gravados da seguinte maneira:

L01N00.wav representa o locutor 01 dizendo o dígito 0 (lembrando que é o usuário quem decide a nomenclatura), então deve criar um arquivo de texto com o nome *convert.scp* com o seguinte conteúdo:

```
Data/train/L01N00.wav    Data/train/L01N00.mfc
Data/train/L01N01.wav    Data/train/L01N00.mfc
[...]
Data/test/LXXN0X.wav     Data/test/LXXN0X.mfc
```

Esta lista só funcionará se rodar o *HCopy* no diretório do projeto, caso rode o *HCopy* dentro da pasta *Data* o formato será:

```
train/L01N00.wav  train/L01N00.mfc
train/L01N01.wav  train/L01N00.mfc
[...]
test/LXXN0X.wav  test/LXXN0X.mfc
```

Nota-se que neste caso ambos os arquivos de teste quanto os arquivos de treinamento serão listados. A primeira coluna representa o arquivo no formato original e a segunda representa o arquivo depois de convertido. Também é possível direcionar os arquivos convertidos para uma pasta diferente. Caso queira separá-los em arquivos diferentes terá de rodar o *HCopy* uma vez para cada arquivo. Essa etapa de conversão nem sempre é necessária de se repetir, uma vez que já tiver todos os arquivos convertidos, mas atenção: Caso mude os parâmetros do arquivo *config_wav2mfc* então terá de reconvertê-los cada vez que fizer alterações.

Tendo os dois arquivos necessários prontos, então basta usar o seguinte comando no terminal para executar a conversão:

```
HCopy -T 1 -C config_wav2mfc -S convert.scp
```

O comando *HList* pode ser utilizado para visualizar o conteúdo dos arquivos .mfc criados, mas não interfere em nada no desempenho do reconhecedor. A Figura A.7, a seguir, detalha as etapas necessárias para o reconhecimento de fala. Perceba que há duas formas de se fazer a preparação dos dados, neste documento será abordada apenas a forma utilizando transcrições (direita).

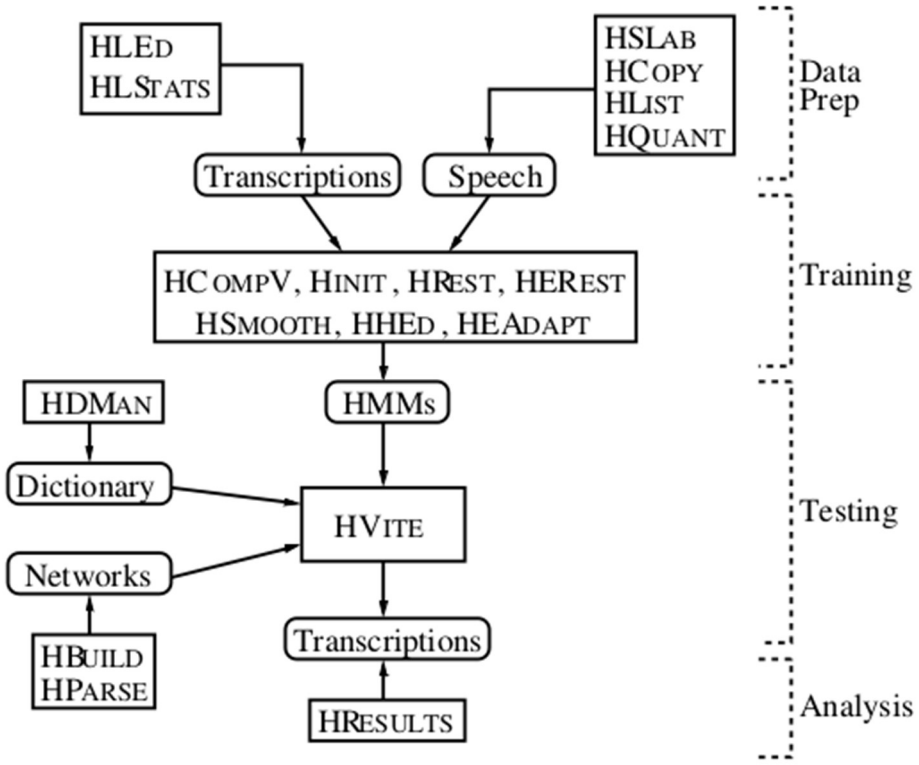

Figura A.7 - Sequência de ações a serem executadas no reconhecimento.

A.5 DEFININDO AS HMMS

Nesta etapa serão definidos os Modelos Ocultos de Markov que o HTK utilizará para associar a cada possibilidade. Para isso, inicialmente é necessário um dicionário descrevendo cada palavra e seus fones possíveis de serem reconhecidos durante o processo. Muitas são as possibilidades de se montar esse dicionário. Um exemplo é descrito a seguir. Crie uma arquivo de texto com o nome *dict* na pasta Dictionary/Src/ .

zero	[zero]	z eh r o
zero	[zero]	z e r o
um	[um]	u mm
dois	[dois]	d o ih ss
tres	[tres]	tr e ss
tres	[tres]	tr e ih ss
quatro	[quatro]	kua tr o
cinco	[cinco]	ss ih nn k o

seis	[seis]	ss e ih ss
sete	[sete]	ss eh t e
sete	[sete]	ss eh t ih
oito	[oito]	o ih t o
nove	[nove]	nn oh v e
nove	[nove]	nn oh v ih
SENT-START	[]	sil
SENT-END	[]	sil

No exemplo percebe-se que cada dígito pode ser representado mais de uma vez dependendo de sua pronúncia, as vezes um usuário fala “seti” e outro fala “sete”, que é uma variação muito comum, então é possível especificar ao HTK cada um dos dois separadamente.

A segunda coluna representa como o HTK vai representar a saída quando esse dígito for encontrado, por exemplo, se quiser que o HTK mostre quando achou que era “seti” ao invés de “sete” basta colocar [seti] na segunda coluna do dígito sete.

E na última coluna são os fones correspondentes a cada palavra, nesse caso foi inventada essa forma de representar os fones, caso queira um padrão há dicionários que mostram a fonética de cada palavra na *internet*.

O *SENT-START* e *SENT-END* são necessários caso tenha os usado na regra gramatical, sua saída é nula e o fone é apenas um silêncio.

Para adiantar o trabalho, crie um novo arquivo de texto usando o editor de sua preferência e nele coloque cada HMM em uma linha com aspas no início e no final. Uma HMM no caso deste tutorial é um fone, não uma palavra, utilizando o dicionário mostrado há pouco teríamos o seguinte:

```
"z"
"eh"
"r"
"o"
"u"
"mm"
"d"
"ih"
"ss"
"tr"
"e"
"kua"
"nn"
"k"
"t"
"oh"
"v"
"sil"
```

Caso tivesse um reconhecedor de frases teria de colocar cada fone presente na frase e possíveis variações, assim como no dicionário. Salve esse arquivo com o nome de *HmmList* na pasta *Configs*. Lembre de colocar o "*sil*", pois o silêncio também deve ser modelado como uma HMM. No caso de um reconhecedor de frases teríamos o "*sp*" que representa uma pausa curta entre os fones.

Já aproveitando este arquivo, copie-o e então chame a cópia de *phonemodels0.* Recorte-o e cole-o na pasta *Labels*. Antes de prosseguir, retire as aspas. O arquivo *phonemodels0* será semelhante ao exemplo:

z
eh
r
o
u
mm
d
ih
ss
tr
e
kua
nn
k
t
oh
v
sil

Uma vez criado o dicionário, podemos passar a próxima etapa: criar um arquivo de configuração para a etapa de estimação.

Nesta etapa se utilizará o comando *HCompV*, ele demanda como parâmetros de entrada três arquivos: um arquivo de configuração, a lista de todos as vozes que serão usadas apenas no treinamento e um arquivo “protótipo” para definir como será calculado os arquivos de saída.

O arquivo de configuração é muito similar ao arquivo *config_wav2mfc* criado anteriormente. Crie um novo arquivo de texto na pasta *Configs* com o nome *config_mfc*, em seu conteúdo coloque:

```
# Coding parameters
TARGETKIND = MFCC_0_D_A
TARGETRATE = 280000
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 365000
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = T
```

Lembre de colocar os parâmetros semelhantes aos usados na configuração dos arquivos .mfc.

Ainda na pasta *Configs*, crie um novo arquivo de texto com o nome *proto*, nele cole o seguinte:

```
~o <VecSize> 39 <MFCC_0_D_A>
~h "proto"
<BeginHMM>
<NumStates> 5
<State> 2
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 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 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0
```

```
<State> 3
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 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 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0
<State> 4
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 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 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0
<TransP> 5
0.0 1.0 0.0 0.0 0.0
0.0 0.6 0.4 0.0 0.0
0.0 0.0 0.6 0.4 0.0
0.0 0.0 0.0 0.7 0.3
0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Este código é uma forma do HTK inicializar a primeira HMM e a partir dela teremos de dizer como se comportarão todas as outras e então a cada reestimação será atualizado o valor dessa matriz. Por isso diz-se que esse arquivo é apenas um protótipo.

O último arquivo necessário para utilizar o *HCompV* é listar todos os arquivos .mfc a serem utilizados durante o treinamento.

Crie um novo arquivo de texto chamado *training.scp* dentro da pasta *Configs*. Seu conteúdo deve ser similar ao exemplo seguinte:

```
Data/train/L1N00-01-011-001-01.mfc
Data/train/L1N01-01-011-001-01.mfc
Data/train/L1N02-01-011-001-01.mfc
Data/train/L1N03-01-011-001-01.mfc
Data/train/L1N04-01-011-001-01.mfc
[...]
```

Onde a localização de cada arquivo utilizado no treinamento deverá ser especificada. Neste caso supõe-se que *HCompV* será chamado na pasta do projeto e não dentro das pastas inerentes (*Configs*, *Data, Labels*, *Models*, etc.), esse é o caminho lógico a se seguir.

Salvando esses arquivos necessários é hora de rodar o comando para criar os próximos arquivos necessários. Antes de prosseguir será necessário criar uma pasta chamada hmm0 dentro do diretório *Models*.

```
HCompV -C Configs/config_mfc -f 0.01 -m -S Configs/training.scp -M Models/hmm0 Configs/proto
```

Dois arquivos serão então gerados dentro da pasta hmm0: proto e vFloors. Ambos serão necessários para se gerar outros dois arquivos que serão utilizados nas etapas de reestimação.

O arquivo *proto* é da forma:

```
~o
<STREAMINFO> 1 39
<VECSIZE> 39<NULLD><MFCC_0_D_A><DIAGC>
~h "proto"
<BEGINHMM>
```

```
<NUMSTATES> 5
<STATE> 2
<MEAN> 39
 -9.021510e+00 1.475264e+00 -3.056515e-01 -2.347598e+00 -3.787058e+00 -2.801918e+00 -3.151892e+00 -2.780075e+00 -2.021620e+00 -2.057544e+00 -1.566315e+00 -1.686084e+00 6.509219e+01 1.910519e-02 2.391953e-02 3.212003e-03 1.042973e-02 6.046844e-03 1.607360e-02 8.954209e-03 6.728815e-04 -3.989456e-03 2.263297e-03 -1.093109e-02 -9.731252e-03 -3.012684e-03 1.386369e-03 -3.054424e-03 -1.019288e-03 -6.460799e-05 -1.275310e-03 -2.903126e-04 -2.246951e-03 4.901755e-03 2.685908e-03 1.127640e-03 1.299447e-03 2.243091e-03 -6.091754e-03
<VARIANCE> 39
 6.591405e+01 4.255153e+01 5.676612e+01 6.595497e+01 6.922560e+01 5.657383e+01 5.178168e+01 6.259002e+01 4.091607e+01 3.830684e+01 3.633432e+01 3.051450e+01 1.278141e+02 3.723217e+00 2.542111e+00 3.788159e+00 4.018138e+00 4.016970e+00 3.662831e+00 3.608129e+00 3.787151e+00 3.034247e+00 2.883769e+00 2.651216e+00 2.275900e+00 4.602840e+00 5.763927e-01 4.310517e-01 6.197264e-01 6.623716e-01 6.715016e-01 6.579852e-01 6.426996e-01 6.360610e-01 5.598588e-01 5.281521e-01 4.793243e-01 4.173611e-01 6.408510e-01
<GCONST> 1.319196e+02
<STATE> 3
[...]
<TRANSP> 5
 0.000000e+00 1.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 0.000000e+00 6.000000e-01 4.000000e-01 0.000000e+00 0.000000e+00
 0.000000e+00 0.000000e+00 6.000000e-01 4.000000e-01 0.000000e+00
 0.000000e+00 0.000000e+00 0.000000e+00 7.000000e-01 3.000000e-01
 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
<ENDHMM>
```

Ele lembra muito o "*proto*" original que foi criado na pasta *Configs*. Na verdade, o que aconteceu é que a matriz de inicialização foi atualizada para os valores mostrados a partir dos dados disponibilizados no treinamento.

O arquivo *vFloors* terá o formato:

```
~v varFloor1
<Variance> 39
 6.591405e-01 4.255153e-01 5.676612e-01 6.595497e-01 6.922560e-01
5.657383e-01 5.178168e-01 6.259001e-01 4.091607e-01 3.830684e-01 3.633432e-
01 3.051450e-01 1.278141e+00 3.723216e-02 2.542111e-02 3.788159e-02
4.018138e-02 4.016970e-02 3.662831e-02 3.608128e-02 3.787151e-02 3.034247e-
02 2.883768e-02 2.651216e-02 2.275900e-02 4.602840e-02 5.763927e-03
4.310517e-03 6.197263e-03 6.623716e-03 6.715016e-03 6.579852e-03 6.426996e-
03 6.360610e-03 5.598588e-03 5.281521e-03 4.793243e-03 4.173611e-03
6.408510e-03
```

Copie esse arquivo *vFloors* e chame o novo arquivo de *macros* no mesmo local onde estão salvos *proto* e *vFloors*. As três primeiras linhas de *macros* serão idênticas à *proto*:

```
~o
<STREAMINFO> 1 39
<VECSIZE> 39<NULLD><MFCC_0_D_A><DIAGC>
```

Atenção: O que ocorre em geral é que na criação de proto, onde está escrito MFCC_0_D_A está na verdade MFCC_D_A_0, isso sempre vai acontecer e será necessário corrigir apenas no arquivo *macros*.

Agora, copie o arquivo *proto* para um novo arquivo chamado *hmmdefs* no mesmo diretório, retire as três primeiras linhas desse arquivo (que são justamente as mesmas linhas que serão colocadas em *macros*).

Agora vem uma etapa importante. Abra o arquivo *HmmList* e editaremos o *hmmdefs* criado anteriormente e, para cada fone do *HmmList*, terá de substituir o ~h “proto” no arquivo *hmmdefs* pelo ~h “fone da lista” e então copiar toda a matriz uma vez para cada fone. Esse procedimento parece ser braçal quando se tem um

dicionário muito vasto de fones, mas é possível automatizar com uma solução proposta no fim do tutorial.

Por exemplo, para o primeiro fone da lista:

~h "proto" se tornará ~h “z”, copie toda a matriz entre <BEGINHMM> e <ENDHMM>

Então abaixo do <ENDHMM> substitua o ~h "proto" por ~h “eh” e mais uma vez copie e cole tudo entre <BEGINHMM> e <ENDHMM>. Obs.: a matriz será a mesma para todos os fones, ela só vai variar durante a reestimação.

Repita o procedimento para cada HMM listada.

É importante lembrar que essa regra de utilizar fones como HMM não é global, em muitos casos se utilizam difones, trifones ou outras unidades fonéticas, isso pode ser variado como desejar.

Salvando o arquivo *hmmdefs* já com todas as matrizes (uma para cada HMM) é hora de passar à etapa de reestimação.

A.6 REESTIMANDO O MODELO DE MARKOV

Essa etapa é quando realmente será realizado o treinamento a partir do algoritmo de Baum-Welch. Será utilizada a ferramenta *HERest* para essa etapa. Além do *phonemodels0* criado anteriormente será necessária a criação de um novo arquivo.

Gere um novo arquivo de texto com o nome *speakertrainmodels0* dentro da pasta *Labels* (nesses arquivos não é necessária extensão (por exemplo .txt), basta salvá-los dessa forma mesmo. No conteúdo desse arquivo terá de transcrever o que cada arquivo usado no treinamento (apenas no treinamento) está representando. Algo do tipo:

```
#!MLF!#
"*Data/train/L1N00-01-011-001-01.lab"
zero
.
"*Data/train/L1N01-01-011-001-01.lab"
um
.
```

```
"*Data/train/L1N02-01-011-001-01.lab"
dois
.
"*Data/train/L1N03-01-011-001-01.lab"
tres
.
"*Data/train/L1N04-01-011-001-01.lab"
quatro
.
[...]
```

É sempre necessário manter esse formato com um ponto (.) no final, o endereço entre aspas e um asterisco (*) antes do endereço. Também é necessário #!MLF!# no início do arquivo.

Por enquanto não é necessário colocar cada fone, bastam as palavras, há uma ferramenta que fará isso. Utilizando *HLEd* podemos acessar o dicionário e substituir cada palavra por sua transcrição fonética equivalente.

Para usar essa ferramenta é necessário criar um novo arquivo de texto que pode ser chamado de *mkphones.led* e salvo dentro do diretório *Configs*. Neste arquivo devem estar as instruções que o *HLEd* usará para fazer essa conversão de palavras em fones. No caso do reconhecimento de dígitos em fala isolada basta inserir:

```
EX
IS sil sil
DE sp
```

O que na verdade ele faz é dizer ao *HLEd* para substituir cada palavra pelos fones equivalentes definidos no arquivo *dict* e então adicionar “*sil*” antes e depois de cada palavra, assim como apagar todos os “*sp*” presentes. Após salvar utilize o seguinte comando no terminal:

```
HLEd -d Dictionary/Src/dict -i Labels/words.mlf Configs/mkphones.led Labels/speakertrainmodels0.mlf
```

Um novo arquivo será gerado na pasta *Labels* com o nome *words.mlf*. Esse arquivo será do tipo:

```
#!MLF!#
"*Data/train/L1N00-01-011-001-01.lab"
sil
z
eh
r
o
sil
.
[...]
```

Ou seja, ele contém as substituições necessárias durante a etapa de reestimação.

Uma colocação importante neste ponto deste passo a passo é que há duas formas distintas de se dizer ao HTK como interpretar cada fone.

Uma é através de *labels* (ou etiquetas) onde o usuário diz: “Para este arquivo de áudio, no instante de tempo X começará o fone ‘TAL’, este fone terminará exatamente em X + t” e assim vai para cada fone de cada arquivo de áudio. Essa “etiquetagem” é muitas vezes trabalhosa de ser feita. O próprio HTKDemo utiliza esse método. Um exemplo é:

```
0  1383750 S
1383750  1725000 sil
1725000  2583125 z
2583125  3005000 eh
3568750  4543750 r
4543750  5170000 o
5170000  5636250 sil
```

Recordando que as unidades estão em 10^{-7} segundos. Essa abordagem não será utilizada neste documento. Para mais detalhes, acesse:

http://www.ee.columbia.edu/ln/LabROSA/doc/HTKBook21/node79.html

A Figura A.8, na sequência, mostra as duas possibilidades. No caso de utilizar as etiquetas não será utilizado o comando *HCompV* e sim os comandos *HInit* e *HRest*. Mais informações sobre o que fazem esses comandos podem ser encontradas na documentação oficial do HTK.

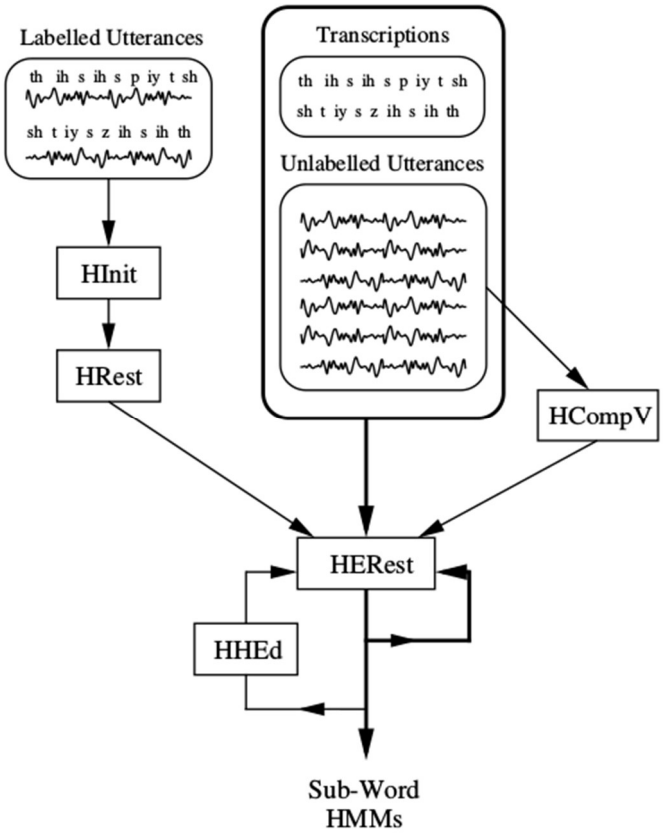

Figura A.8 - Diferentes formas de dizer ao HTK onde estão as HMMs.

Antes de utilizar a reestimação será necessário criar uma nova pasta para cada reestimação, por exemplo, se quisermos fazer três reestimações será necessário criar as pastas *hmm1*, *hmm2* e *hmm3* dentro do diretório *Models*.

Feito isso, faz-se a primeira reestimação utilizando o terminal.

```
HERest -C Configs/config_mfc -I Labels/words.mlf -t 250.0 150.0 1000.0 -S Configs/training.scp -H Models/hmm0/macros -H Models/hmm0/hmmdefs -M Models/hmm1 Labels/phonemodels0
```

Isso utilizará todos os dados disponíveis para reestimar o modelo e na pasta *hmm1* serão geradas as versões atualizadas de *macros* e *hmmdefs*.

Um erro comum de se aparecer nessa etapa é quando ele diz que não há muita informação sobre determinado modelo, ou seja, os dados utilizados no treinamento são insuficientes para se modelar corretamente determinada HMM, neste caso será necessário utilizar mais informação no treinamento, talvez gravando mais áudio.

Para a segunda reestimação utiliza-se comando semelhante:

```
HERest -C Configs/config_mfc -I Labels/words.mlf -t 250.0 150.0 1000.0 -S Configs/training.scp -H Models/hmm1/macros -H Models/hmm1/hmmdefs -M Models/hmm2 Labels/phonemodels0
```

Ou seja, dessa vez ele utilizará *macros* e *hmmdefs* atualizados na pasta *hmm1* e salvará a saída na pasta *hmm2* e as reestimações seguintes utilizarão a mesma lógica. O parâmetro -t 250.0 150.0 1000.0 representa algo como “fator de poda”, não sei explicar ao certo sua influência. De acordo com a documentação buscada:

> *“-t X Y Z : beam search pruning for alignment. First the pruning factor is set to X. If the alignment fails, x is incremented by Y and retried. This is done until Z is reached. Then an error message is issued.”*

Se observar na Figura A.8, existe um comando chamado *HHED* utilizado em *feedback* entre a saída e entrada do bloco de reestimação *HERest*. Essa ferramenta é uma das mais poderosas do pacote HTK pois ela é responsável pela manipulação de HMMs podendo clonar os fones para bi ou trifones, manipular transições, compactar as definições de cada HMM, etc.

Não será dado ênfase a esse comando, pois nessa identificação tenta-se fazer algo não tão aprofundado. Mas pode-se estudar mais sobre a utilização deste comando e implementar uma reestimação mais eficiente ou específica.

A.7 TESTANDO O RECONHECEDOR

Uma vez terminado o treinamento do reconhecedor, é o momento de testar seu funcionamento e extrair resultados da ferramenta. Já temos o dicionário pronto, o que falta é dizer quais serão os dados que o HTK deverá utilizar para testar durante o reconhecimento.

Essa função é dada pela ferramenta *HVite*, que como o próprio nome já sugere, usará o algoritmo de Viterbi para realizar essa ação.

Abra o editor de texto e no conteúdo deverá incluir uma lista semelhante à lista criada em *training.scp,* onde são listados quais os arquivos *.mfc* que serão utilizados nesta etapa, lembre que os arquivos tanto de treino quanto teste são convertidos para *.mfc* através da ferramenta *HCopy* (discutida anteriormente), caso não tenha convertido todos os arquivos esse é o momento de fazê-lo.

O arquivo criado terá um formato parecido com:

```
Data/test/L18N00-01-011-001-01.mfc
Data/test/L18N01-01-011-001-01.mfc
Data/test/L18N02-01-011-001-01.mfc
Data/test/L18N03-01-011-001-01.mfc
Data/test/L18N04-01-011-001-01.mfc
Data/test/L18N05-01-011-001-01.mfc
[...]
```

Basta lembrar-se da localização destes arquivos e de onde está chamando o comando no terminal. Salve esse arquivo com o nome *testing.scp* dentro da pasta *Configs*.

Também será necessário um arquivo semelhante ao *speakertrainmodels0.mlf*, mas dessa vez listando as palavras do teste, algo como:

```
#!MLF!#
"*Data/test/L18N00-01-011-001-01.lab"
zero
.
"*Data/test/L18N01-01-011-001-01.lab"
um
.
"*Data/test/L18N02-01-011-001-01.lab"
dois
.
"*Data/test/L18N03-01-011-001-01.lab"
tres
.
"*Data/test/L18N04-01-011-001-01.lab"
quatro
.
"*Data/test/L18N05-01-011-001-01.lab"
cinco
.
```

Atenção: Neste caso não é necessário utilizar a ferramenta *HLEd* para converter as palavras em fones, o próprio HTK irá confrontar os resultados com as palavras listadas nesse arquivo. Salve esse arquivo com a nomenclatura *speakertestmodels0.mlf* dentro da pasta *Labels*.

Assim temos todos os arquivos necessários para testar o reconhecedor treinado. Utilize o comando *HVite* no terminal com os seguintes parâmetros:

```
HVite -H Models/hmm3/macros -H Models/hmm3/hmmdefs -S Configs/testing.scp -i Results/recout.mlf -w Dictionary/Src/wdnet -p 0.0 -s 5.0 Dictionary/Src/dict Configs/HmmList
```

Os arquivos *hmmdefs* e *macros* nesse caso deverão ser baseados na última reestimação feita, por isso foi usado a pasta *hmm3*, caso tivesse feito 10 reestimações teria que declarar a pasta *hmm10*. Olhando na descrição das opções do *HVite*, o *–p* representa:

“-p f inter model trans penalty (log) 0.0”

E o *–s* representa um fator de escala da gramática:

“-s f grammar scale factor 1.0”

Rodando esse comando com sucesso deverá gerar um arquivo chamado *recoult.mlf* dentro da pasta *Results*.

Este arquivo conterá uma listagem parecida com *speakertestmodels0.mlf*, mas dessa vez dizendo ao usuário quais foram as conclusões do HTK sobre cada um dos arquivos de teste, lá é possível saber quais foram os erros e analisar em quais casos o HTK errou e em que casos acertou.

A.8 ANALISANDO OS RESULTADOS

Para finalizar o reconhecimento de fala, o HTK disponibiliza uma ferramenta chamada *HResults* que tem por função comparar o arquivo de saída do HVite chamado *recoult.mlf* com o arquivo com as transcrições corretas, chamado *speakertestmodels0.mlf*. Basta dar o comando:

HResults -I Labels/speakertestmodels.mlf Configs/HmmList Results/recout.mlf >> Results/ResultLog

No caso, esse comando irá salvar o resultado em um novo arquivo de texto chamado *ResultLog* dentro da pasta *Results*. Caso queira que o resultado apareça no terminal ao invés de ser salvo em arquivo utilize:

HResults -I Labels/speakertestmodels.mlf Configs/HmmList Results/recout.mlf

Algo assim deverá aparecer:

```
====================== HTK Results Analysis =======================
Date: Sun Oct 16 16:33:19 2016
Ref : Labels/speakertestmodels0.mlf
Rec : Results/recout.mlf
---------------------- Overall Results -------------------------
SENT: %Correct=99.44 [H=176, S=1, N=177]
WORD: %Corr=99.44, Acc=99.44 [H=176, D=0, S=1, I=0, N=177]
```

Neste caso utilizou-se reconhecimento em fala isolada então a taxa de frases corretas (*SENT*) deve ser igual à taxa de acertos de palavras (*WORD*), caso utilizasse reconhecimento em fala contínua esses dados seriam independentes.

Neste exemplo obteve-se uma taxa de acertos muito alta (com 176 dígitos corretos e apenas 1 errado), mas essa taxa varia dependendo de inúmeros fatores.

A.9 UTILIZANDO O SCRIPT

Uma vez terminado todo o reconhecimento pode-se notar que todas essas etapas demandam certo tempo e que seria muito difícil utilizar esse método “manual” a cada vez que for executar uma verificação, em muitos casos será necessário executar uma varredura, por exemplo, como se comportará a taxa de acertos em função de determinado parâmetro escolhido e que varie a cada novo teste. Se tivermos que abrir o arquivo, modificar um valor, salvar e então executar comando por comando no terminal levaremos dias para terminar a tarefa.

Visando resolver este problema, é possível gerar *scripts* que automatizem esse processo, deixando o controle do usuário sobre o teste muito mais rápido e simples.

Crie uma nova pasta no diretório em que está o projeto do HTK chamada *HTK_Scripts*. Nesta pasta ficarão localizados os scripts criados relacionados ao projeto.

Desenvolvi um script utilizando comandos de *shell script* (*bash*) no Linux, este *script* é chamado de *scriptHTK.sh* e salvo na pasta criada há pouco, a função desse script é realizar todo o reconhecimento de fala descrito nesse tutorial, sendo necessário chamar apenas o script e não comando por comando. Também é possível usá-lo para fazer varreduras, ou seja, fazer um ou mais parâmetros variando a cada iteração.

Uma aplicação é saber como se portará a taxa de acertos enquanto variamos o tamanho da janela ou o número de iterações, por exemplo, também é possível criar um "*loop*" dentro de outro, podendo assim otimizar e encontrar os parâmetros ideais do reconhecedor.

Para utilizá-lo a partir do tutorial descrito neste documento será necessário fazer algumas alterações na nomenclatura de alguns arquivos utilizados anteriormente.

Na pasta *Configs*:

- Altere o nome do arquivo *training.scp* para *training0.scp*;
- Altere o nome do arquivo *testing.scp* para *testing0.scp*;

Na pasta *Results*:

- Apague ou salve todo o conteúdo de *ResultLog* em outro arquivo, mantendo o arquivo atual sem utilizar o script pode gerar um erro de interpretação no segundo *script* (visto na sequência).

```
#! /bin/bash

#
# Usar chmod +x scriptHTK.sh no terminal antes de usar pela primeira vez o script
# Usar chmod +x ResultAnalyzer.sh no terminal antes de usar pela primeira vez o script
# Ambos os scripts devem estar no mesmo diretório
# Para usar digite sudo ./scriptHTK.sh no terminal e dê enter
#
```

```
cd ..

T=0;                    #(T=0 : Grupo original, T=1 : homens treinam, mulheres testam, T=2 : mulheres treinam, homens testam, T=3 : Grupo original com funções invertidas, T=4 : Grupo pequeno de mulheres, os outros locutores testam, T=5 : Poucos homens treinam, os outros locutores testam, T=6 : Apenas um locutor treina, todos os outros testam, T=7, todos treinam, apenas um testa)
nReest=5;               #número de reestimações, ideal 5
WINDOWSIZE=365000;      #Duração da janela, original 300000, ideal 365000 = 36,5 ms
TARGETRATE=280000;      #Tamanho do vetor de parâmetros, original 100000, ideal 280000 = 280 vetores, $TARGETRATE não pode ser maior que $WINDOWSIZE
FinalValue=8;           #Quebra o loop quando atingir esse valor
Passo=1;                #Passo de alteração dentro do loop
CONVERT="TRUE";         #Converter os arquivos para mfc? TRUE se a variável $WINDOWSIZE ou $TARGETRATE mudar, se variar só o $nReest não é preciso

COUNTER=0; #Não alterar
COUNTER2=0; #Não alterar

#while [  $T -lt $FinalValue ]; do

        sed -i.bak -e '6d' Configs/config_mfc
        sed -i.bak 's/USE/WINDOWSIZE = '$WINDOWSIZE'\nUSE/g' Configs/config_mfc
        sed -i.bak -e '3d' Configs/config_mfc
        sed -i.bak 's/SAVECOM/TARGETRATE = '$TARGETRATE'\nSAVECOM/g' Configs/config_mfc

        sed -i.bak -e '9d' Configs/config_wav2mfc
        sed -i.bak 's/USE/WINDOWSIZE = '$WINDOWSIZE'.0\nUSE/g' Configs/config_wav2mfc
        sed -i.bak -e '6d' Configs/config_wav2mfc
```

```
		sed -i.bak 's/SAVECOM/TARGETRATE = '$TARGETRATE'.0\nSAVECOM/g' Configs/config_wav2mfc

		HParse Dictionary/Src/grammar Dictionary/Src/wdnet

		HLEd -d Dictionary/Src/dict -i Labels/words.mlf Configs/mkphones.led Labels/speakertrainmodels$T.mlf

		if [ "$CONVERT" == "TRUE" ]; then
			HCopy -T 1 -C Configs/config_wav2mfc -S Configs/convert.scp
		fi

		HCompV -C Configs/config_mfc -f 0.01 -m -S Configs/training$T.scp -M Models/hmm0 Configs/proto

		perl -pi -e 's/<MFCC_D_A_0>/<MFCC_0_D_A>/g' Models/hmm0/proto
		cat Models/hmm0/proto | tail -28 | head -31> Models/hmm0/proto2 #Apaga o cabeçalho

		cat Models/hmm0/proto | tail -31 | head -3 > Models/hmm0/header #Deixa só o cabeçalho
		cat Models/hmm0/header Models/hmm0/vFloors > Models/hmm0/macros #Cria o arquivo macros

		cat Models/hmm0/proto2 > Models/hmm0/hmmdefs #Cria o hmmdefs

		while IFS='' read -r line || [[ -n "$line" ]]; do
			perl -pi -e 's/~h "proto"/~h '$line'/g' Models/hmm0/hmmdefs #substitui o "proto" pelo equivalente do dicionário
			cat Models/hmm0/proto2 >> Models/hmm0/hmmdefs #Cria uma nova cópia do proto a cada iteração no hmmdefs
		done < "Configs/HmmList"

		while [  $COUNTER -lt $nReest ]; do
```

```
                                    let COUNTER2=COUNTER+1
                                    rm -rf Models/hmm$COUNTER2
                                    mkdir Models/hmm$COUNTER2
                                    HERest -C Configs/config_mfc -I Labels/words.mlf -t
250.0 150.0 1000.0 -S Configs/training$T.scp -H Models/hmm$COUNTER/macros -H
Models/hmm$COUNTER/hmmdefs -M Models/hmm$COUNTER2
Labels/phonemodels0
                                  let COUNTER=COUNTER+1
                                done

        HVite -H Models/hmm$COUNTER2/macros -H
Models/hmm$COUNTER2/hmmdefs -S Configs/testing$T.scp -i Results/recout.mlf -w
Dictionary/Src/wdnet -p 0.0 -s 5.0 Dictionary/Src/dict Configs/HmmList

        echo 'Teste = '$T'; Reestimações = '$nReest'; Tamanho da janela =
'$WINDOWSIZE'; Tamanho do vetor = '$TARGETRATE';' >> Results/ResultLog

        HResults -I Labels/speakertestmodels$T.mlf Configs/HmmList
Results/recout.mlf >> Results/ResultLog
        COUNTER=0;
        COUNTER2=0;

#let T=T+Passo
#                       done
cd HTK_Scripts
sh ./ResultAnalyzer.sh
```

Copie todo o texto acima e cole no script. Crie um novo script com o nome *ResultAnalyzer.sh* no mesmo diretório do outro, copie e cole o seguinte código:

```
#! /bin/bash

cd ..
cp Results/ResultLog Results/ResultMatrix

sed -i.bak -e 's/WORD: %Corr=\(.*\),/\1/' -e 's/Tamanho do vetor =\(.*\);/\1/' -e 's/Tamanho da janela =\(.*\);/\1/' -e 's/\(Acc=\).*\(]\)/\1\2/' -e 's/Reestimações =\(.*\);/\1/' -e 's/Teste =\(.*\);/\1/' -e 's/, Acc=]//g' -e 's/==========//g' -e 's/========== HTK Results Analysis ==========//g' -e 's/---------- Overall Results ----------//g' -e 's/\( Date:\).*\(2016\)/\1\2/' -e 's/ Date:2016//g' -e 's/\(SENT:\).*\(]\)/\1\2/' -e 's/SENT:]//g' -e 's/\( Ref :\).*\(.mlf\)/\1\2/' -e 's/\( Rec :\).*\(.mlf\)/\1\2/' -e 's/ Ref :.mlf//g' -e 's/ Rec :.mlf//g' -e 's/ / /g' -e '/^\s*$/d' Results/ResultMatrix

sed -i.bak ':a;N;$!ba;s/\n/ /g' Results/ResultMatrix
egrep -o '\S+\s+\S+\s+\S+\s+\S+\s+\S+' Results/ResultMatrix > i.bak
mv i.bak Results/ResultMatrix

#sed -i.bak '1i\\' Results/ResultMatrix

awk '{$3 = $3/10000; print}' Results/ResultMatrix > i.bak
mv i.bak Results/ResultMatrix
awk '{$4 = $4/1000; print}' Results/ResultMatrix > i.bak
mv i.bak Results/ResultMatrix
sed -i.bak '1i\N_Teste Reest. Janela(ms) Vetor Acertos(%)' Results/ResultMatrix

scidavis Results/ResultMatrix
```

Esse segundo script será chamado dentro do primeiro e ele dependerá do aplicativo *SciDavis*, disponível para Ubuntu pelo comando no terminal:

sudo apt-get install scidavis

Todo o projeto utilizado como base para este tutorial está disponível em:

https://drive.google.com/open?id=0ByZzV_kRgeveZkZaMVAySWtlZVE

O tamanho aproximado é 19,1 MB. Todo o conteúdo deste pacote é *Open Source*, podendo ser usado com fins educacionais.

Para extrair, abra uma nova janela do terminal e se desloque ao diretório onde foi baixado o arquivo. Digite o comando:

tar -zxvf ReconhecimentodeDigitos.tar.gz

E uma nova pasta *ReconhecimentodeDigitos* será criada no mesmo diretório.

O primeiro *script* tem a função de executar todas as tarefas necessárias à identificação de fala discutidas no tutorial e tem a possibilidade de controle de algumas variáveis como o número de reestimações (*nReest*), o tamanho da janela (*WINDOWSIZE*), a taxa de vetores por segundo (*TARGETRATE*), o tipo de teste (observe que neste caso serão necessários mais arquivos para executar os outros testes, esses arquivos podem ser baixados no link disponibilizado há pouco.

Também há a possibilidade de incrementar mais funções utilizando a mesma lógica de programação, na verdade as possibilidades são muitas.

Caso queira usar a função de varredura, para fazer variar uma das variáveis dentro de várias iterações, será preciso descomentar as linhas 22, 73 e 74.

Na linha 22:

while [$T -*lt* $*FinalValue*]; do

Substitua o T pelo nome da variável que deseja controlar. A variável *FinalValue* dirá o valor final do *loop*, lembre-se que –lt, significa *less than* ou menor que... ou seja, se quiser trocar para menor ou igual que... substitua o –lt por –le.

Na linha 73:

let T=T+Passo

Substitua o T pela variável que deseja controlar mais uma vez. O Passo regula de quanto em quanto ela variará a cada iteração. Se estiver trabalhando com *WINDOWSIZE*, cujos valores são na unidade de centenas de nano-segundo é preferível que o passo seja de pelo menos 1000 para variar de 0,1 ms a cada iteração.

Caso não queira fazer uma varredura e sim apenas um teste simples, basta comentar novamente as linhas anteriormente citadas.

O segundo *script* terá como função importar os resultados do arquivo *ResultLog* gerado através do primeiro *script* e transformar o resultado em uma matriz contendo as variáveis de interesse, na verdade tudo o que esse *script* faz é apagar as linhas de texto e deixar os números que interessam à análise, além de converter o tamanho da janela para unidade de ms e o número de vetores para sua unidade usual e no final chamará o *scidavis* importando essa matriz, o que facilita a plotagem dos dados.

Uma observação importante: a matriz gerada divide as colunas com espaços, o que não representa o padrão do *scidavis*. Para que a importação funcione corretamente, é necessário abrir o *software*, ir nas propriedades, propriedades de tabelas e trocar o padrão para *Space*.

Caso não queira usar o *SciDavis* e sim o Excel ou o *Matlab* para análise dos dados basta remover ou comentar a linha que chama o programa dentro do script *ResultAnalyzer.sh*.

Se quiser criar novos testes, baseados nesse modelo, basta seguir a mesma lógica:

Adicione os arquivos novos de áudio a serem convertidos no arquivo *convert.scp*;

Crie uma nova lista de arquivos para treinamento e teste (*trainingX.scp* e *testingX.scp*) onde X representa o número do teste;

Liste as palavras ditas por cada locutor nos arquivos *speakertrainmodelsX.mlf* e *speakertestmodelsX.mlf*;

Substitua o valor da variável T pelo número do teste.

APÊNDICE B – TABELAS UTILIZADAS PARA PLOTAR OS RESULTADOS

As colunas marcadas em amarelo representam o eixo das abcissas dos gráficos plotados na seção de resultados, enquanto as colunas em azul representam o eixo das ordenadas dos gráficos plotados. As linhas marcadas em verde representam o melhor resultado encontrado durante a análise dos dados mostrados nas tabelas.

Tabela B.1 - Variação da taxa de acertos em função do número de reestimações, com janela de 20 ms e 150 vetores/s.

n° do Teste	n° de Reest,	Janela (ms)	Vetores/s	Acertos (%)
0	1	20	150	78,53
0	2	20	150	92,66
0	3	20	150	94,34
0	4	20	150	96,61
0	5	20	150	96,61
0	6	20	150	97,74
0	7	20	150	98,31
0	8	20	150	98,31
0	9	20	150	98,31
0	10	20	150	97,74
0	11	20	150	98,31
0	12	20	150	97,18
0	13	20	150	97,18
0	14	20	150	97,18

Tabela B.2 - Taxa de acertos em função da janela. 150 Vetores, 7 reest., teste 0.

Ja	Ac	Ja	Ac	Ja	Ac	Ja	Ac	Ja	Ac	Ja	Ac
		38	96,05	60,5	97,18	83	96,61	105,5	96,05	128	95,48
16	96,05	38,5	94,92	61	96,61	83,5	96,61	106	96,05	128,5	95,48
16,5	94,92	39	95,48	61,5	96,61	84	96,61	106,5	96,05	129	96,05
17	96,05	39,5	96,05	62	96,05	84,5	96,61	107	95,48	129,5	96,05
17,5	96,05	40	96,05	62,5	96,05	85	96,61	107,5	96,05	130	95,48
18	96,61	40,5	96,05	63	96,61	85,5	96,05	108	95,48	130,5	95,48
18,5	97,74	41	95,48	63,5	96,61	86	96,05	108,5	95,48	131	94,92
19	98,31	41,5	95,48	64	96,05	86,5	96,61	109	96,05	131,5	94,92
19,5	98,31	42	95,48	64,5	96,61	87	96,61	109,5	95,48	132	95,48
20	98,31	42,5	95,48	65	96,61	87,5	96,61	110	96,05	132,5	94,92
20,5	97,74	43	94,35	65,5	96,61	88	96,05	110,5	96,05	133	94,92
21	97,74	43,5	95,48	66	96,61	88,5	96,61	111	96,05	133,5	94,92
21,5	97,74	44	94,92	66,5	96,61	89	96,05	111,5	96,05	134	94,92
22	96,61	44,5	93,79	67	97,18	89,5	96,61	112	95,48	134,5	94,92
22,5	94,92	45	94,35	67,5	96,61	90	97,18	112,5	96,05	135	94,92
23	96,61	45,5	94,35	68	96,61	90,5	96,61	113	96,05	135,5	94,92
23,5	95,48	46	94,35	68,5	97,18	91	96,61	113,5	96,05	136	94,92
24	96,05	46,5	94,35	69	97,18	91,5	96,61	114	96,05	136,5	95,48
24,5	94,92	47	94,92	69,5	97,18	92	96,61	114,5	96,05	137	94,92
25	95,48	47,5	94,92	70	97,18	92,5	96,61	115	96,05	137,5	94,92
25,5	94,35	48	95,48	70,5	97,18	93	96,61	115,5	96,05	138	95,48
26	95,48	48,5	95,48	71	97,18	93,5	96,61	116	96,61	138,5	94,92
26,5	95,48	49	94,92	71,5	97,18	94	96,61	116,5	96,61	139	94,92
27	95,48	49,5	97,18	72	96,61	94,5	96,61	117	96,05	139,5	95,48
27,5	96,05	50	96,61	72,5	96,61	95	96,05	117,5	96,05	140	96,05
28	95,48	50,5	96,61	73	96,61	95,5	96,05	118	96,05	140,5	96,05
28,5	93,79	51	96,61	73,5	96,61	96	96,05	118,5	96,61	141	95,48
29	93,79	51,5	97,18	74	96,61	96,5	96,05	119	96,05	141,5	95,48
29,5	93,79	52	96,61	74,5	97,18	97	96,61	119,5	96,05	142	94,92
30	95,48	52,5	96,61	75	96,61	97,5	96,05	120	96,05	142,5	94,92
30,5	94,92	53	96,61	75,5	97,18	98	96,61	120,5	96,05	143	94,92
31	94,35	53,5	96,05	76	97,18	98,5	96,61	121	96,05	143,5	94,92
31,5	93,79	54	96,61	76,5	96,61	99	96,05	121,5	96,05	144	94,92
32	94,92	54,5	96,61	77	96,61	99,5	95,48	122	96,05	144,5	94,92
32,5	93,79	55	96,61	77,5	96,61	100	95,48	122,5	96,05	145	96,05
33	94,35	55,5	97,18	78	96,05	100,5	96,05	123	96,05	145,5	95,48
33,5	93,79	56	96,61	78,5	96,05	101	95,48	123,5	96,05	146	96,05
34	94,92	56,5	96,61	79	96,05	101,5	95,48	124	96,05	146,5	94,92
34,5	94,92	57	96,61	79,5	96,61	102	96,05	124,5	96,05	147	94,92
35	96,05	57,5	96,61	80	96,61	102,5	96,05	125	96,05	147,5	96,05
35,5	96,61	58	96,05	80,5	95,48	103	96,05	125,5	96,05	148	96,05
36	97,18	58,5	97,18	81	96,61	103,5	96,05	126	96,05	148,5	96,05
36,5	97,18	59	96,61	81,5	97,18	104	96,05	126,5	95,48	149	96,05
37	96,61	59,5	96,05	82	96,05	104,5	96,05	127	95,48	149,5	96,05
37,5	94,92	60	96,61	82,5	96,05	105	96,05	127,5	95,48		

Tabela B.3 - Análise da taxa de acertos com a variação da taxa de vetores de parâmetros para janela de 20 ms e 7 reestimações. Teste 0.

n° do Teste	n° de Reest,	Janela (ms)	Vetores/s	Acertos (%)
0	7	20	20	77,4
0	7	20	25	81,36
0	7	20	30	82,49
0	7	20	35	81,36
0	7	20	40	80,79
0	7	20	45	84,75
0	7	20	50	83,62
0	7	20	55	84,18
0	7	20	60	86,44
0	7	20	65	87,57
0	7	20	70	92,66
0	7	20	75	89,83
0	7	20	80	92,09
0	7	20	85	91,53
0	7	20	90	91,53
0	7	20	95	88,14
0	7	20	100	91,53
0	7	20	105	92,66
0	7	20	110	93,22
0	7	20	115	92,09
0	7	20	120	93,22
0	7	20	125	93,22
0	7	20	130	92,66
0	7	20	135	95,48
0	7	20	140	94,35
0	7	20	145	95,48
0	7	20	150	98,31
0	7	20	155	96,05
0	7	20	160	94,92
0	7	20	165	96,61
0	7	20	170	94,35
0	7	20	175	94,92
0	7	20	180	96,61
0	7	20	185	94,35
0	7	20	190	94,35
0	7	20	195	94,35
0	7	20	200	97,18

Tabela B.4 - Variação da taxa de acertos de acordo com a variação do número de vetores por segundo. Dados: Janela de 35 ms, 7 reestimações e teste 0.

Vetores/s	Acertos (%)	Vetores/s	Acertos (%)
5	49,72	180	94,92
10	74,01	185	94,92
15	77,4	190	96,05
20	84,18	195	96,05
25	83,05	200	96,61
30	87,01	205	97,18
35	86,44	210	96,05
40	86,44	215	97,74
45	85,88	220	97,18
50	87,57	225	97,18
55	87,57	230	97,74
60	88,14	235	96,61
65	90,96	240	97,74
70	92,66	245	96,05
75	91,53	250	97,74
80	92,66	255	97,18
85	91,53	260	97,74
90	92,09	265	98,31
95	88,7	270	97,74
100	92,09	275	97,74
105	93,79	280	98,87
110	92,66	285	98,31
115	92,66	290	98,31
120	94,35	295	98,31
125	94,92	300	98,31
130	95,48	305	96,61
135	95,48	310	97,74
140	95,48	315	96,61
145	94,35	320	96,05
150	96,05	325	96,05
155	95,48	330	96,05
160	96,61	335	97,18
165	94,92	340	97,18
170	94,92	345	95,48
175	95,48		

Tabela B.5 - Taxa de acertos com variação do número de reestimações. Janela de 35 ms e 280 vet./s

n° do Teste	n° de Reest,	Janela (ms)	Vetores/s	Acertos (%)
0	1	35	280	82,49
0	2	35	280	96,05
0	3	35	280	98,31
0	4	35	280	98,31
0	5	35	280	98,87
0	6	35	280	98,87
0	7	35	280	98,87
0	8	35	280	97,74
0	9	35	280	97,74

Observação: Na tabela B.6 mostrada abaixo, “Jan” representa o comprimento da janela em ms, enquando “Ac” representa o percentual de acertos encontrado.

Tabela B.6 - Taxa de acertos em função do tamanho da janela para 280 vet., 5 Reest., T= 0.

Jan	Ac	Jan	Ac	Jan	Ac	Jan	Ac	Jan	Ac	Jan	Ac	Jan	Ac	Jan	Ac
28	96,61	50,5	98,31	73	95,48	95,5	94,92	118	96,05	140,5	95,48	163	94,92	185,5	92,66
28,5	96,61	51	98,31	73,5	94,92	96	94,92	118,5	96,05	141	95,48	163,5	94,92	186	93,22
29	96,61	51,5	97,74	74	96,05	96,5	96,05	119	96,61	141,5	94,92	164	94,92	186,5	93,79
29,5	96,61	52	98,31	74,5	96,05	97	96,05	119,5	96,61	142	94,92	164,5	94,92	187	93,22
30	97,18	52,5	98,31	75	96,61	97,5	95,48	120	95,48	142,5	94,92	165	94,92	187,5	93,22
30,5	97,18	53	98,31	75,5	97,18	98	95,48	120,5	96,05	143	94,92	165,5	94,92	188	93,22
31	97,18	53,5	97,74	76	97,18	98,5	95,48	121	96,05	143,5	95,48	166	94,92	188,5	93,22
31,5	97,74	54	97,74	76,5	97,18	99	95,48	121,5	95,48	144	95,48	166,5	94,92	189	93,22
32	98,87	54,5	97,74	77	96,61	99,5	96,05	122	96,61	144,5	94,92	167	94,92	189,5	93,22
32,5	97,74	55	97,74	77,5	97,18	100	96,05	122,5	95,48	145	94,92	167,5	94,92	190	93,22
33	97,74	55,5	98,31	78	97,18	100,5	96,05	123	96,05	145,5	94,92	168	94,92	190,5	93,22
33,5	97,74	56	98,31	78,5	97,18	101	96,05	123,5	96,05	146	94,92	168,5	94,92	191	93,22
34	97,74	56,5	97,74	79	97,18	101,5	95,48	124	95,48	146,5	94,92	169	94,92	191,5	93,79
34,5	97,74	57	98,31	79,5	96,05	102	96,05	124,5	96,05	147	96,05	169,5	94,35	192	93,79
35	98,87	57,5	97,18	80	97,18	102,5	96,61	125	95,48	147,5	95,48	170	94,92	192,5	93,79
35,5	98,31	58	98,31	80,5	97,18	103	96,05	125,5	94,92	148	94,92	170,5	94,92	193	93,79
36	98,31	58,5	97,18	81	97,18	103,5	96,05	126	95,48	148,5	95,48	171	94,35	193,5	93,79
36,5	99,44	59	97,74	81,5	97,18	104	96,61	126,5	94,92	149	94,35	171,5	94,35	194	93,79
37	98,31	59,5	97,74	82	97,18	104,5	96,61	127	95,48	149,5	95,48	172	93,22	194,5	93,79
37,5	98,87	60	97,74	82,5	97,18	105	96,61	127,5	95,48	150	96,61	172,5	93,22	195	93,79
38	98,31	60,5	98,31	83	96,61	105,5	96,61	128	95,48	150,5	96,05	173	93,22	195,5	93,79
38,5	98,31	61	98,31	83,5	97,74	106	96,61	128,5	94,92	151	96,05	173,5	93,79	196	93,79
39	98,31	61,5	98,31	84	97,74	106,5	96,05	129	95,48	151,5	95,48	174	93,79	196,5	93,79
39,5	98,31	62	98,31	84,5	97,18	107	96,05	129,5	95,48	152	95,48	174,5	93,22	197	93,22
40	98,31	62,5	97,74	85	97,74	107,5	96,61	130	94,92	152,5	95,48	175	93,22	197,5	92,66
40,5	98,31	63	98,31	85,5	97,74	108	96,61	130,5	94,92	153	95,48	175,5	93,22	198	92,66
41	97,74	63,5	98,31	86	96,61	108,5	96,05	131	94,35	153,5	95,48	176	93,22	198,5	93,22
41,5	97,18	64	97,74	86,5	96,05	109	96,05	131,5	94,92	154	95,48	176,5	93,22	199	93,79
42	97,18	64,5	97,74	87	96,61	109,5	95,48	132	94,92	154,5	95,48	177	93,22	199,5	93,22
42,5	98,31	65	97,18	87,5	96,61	110	96,05	132,5	94,92	155	94,92	177,5	93,79		
43	97,18	65,5	97,74	88	96,05	110,5	96,61	133	94,35	155,5	95,48	178	93,79		
43,5	97,74	66	97,74	88,5	96,05	111	96,61	133,5	94,35	156	95,48	178,5	93,22		
44	97,74	66,5	97,74	89	96,61	111,5	96,61	134	94,92	156,5	95,48	179	93,79		
44,5	97,74	67	97,74	89,5	96,05	112	96,61	134,5	94,92	157	94,92	179,5	93,22		
45	97,74	67,5	97,18	90	97,18	112,5	96,61	135	94,92	157,5	94,92	180	93,22		
45,5	97,74	68	96,61	90,5	96,61	113	96,61	135,5	94,92	158	95,48	180,5	93,22		
46	97,74	68,5	96,61	91	97,18	113,5	95,48	136	94,92	158,5	94,92	181	93,79		
46,5	97,18	69	96,61	91,5	96,05	114	96,05	136,5	95,48	159	94,92	181,5	93,79		
47	97,74	69,5	96,05	92	96,61	114,5	96,05	137	95,48	159,5	95,48	182	93,79		
47,5	98,31	70	95,48	92,5	96,05	115	96,05	137,5	95,48	160	95,48	182,5	93,79		
48	98,31	70,5	96,61	93	96,05	115,5	95,48	138	95,48	160,5	95,48	183	92,66		
48,5	98,31	71	96,05	93,5	96,05	116	95,48	138,5	95,48	161	94,92	183,5	93,22		
49	97,74	71,5	96,61	94	96,05	116,5	95,48	139	95,48	161,5	94,92	184	93,22		
49,5	98,31	72	95,48	94,5	96,61	117	94,92	139,5	95,48	162	94,92	184,5	92,66		
50	98,31	72,5	95,48	95	96,05	117,5	95,48	140	95,48	162,5	94,92	185	93,22		

Tabela B.7 - Variação da taxa de acertos em relação à variação do n° de reestimações para 280 Vet./s; janela de 36,5 ms e teste 0.

n° do Teste	n° de Reest,	Janela (ms)	Vetores/s	Acertos (%)
0	1	36,5	280	82,49
0	2	36,5	280	96,05
0	3	36,5	280	98,31
0	4	36,5	280	97,74
0	5	36,5	280	99,44
0	6	36,5	280	99,44
0	7	36,5	280	99,44
0	8	36,5	280	98,87
0	9	36,5	280	97,74
0	10	36,5	280	97,74
0	11	36,5	280	97,74
0	12	36,5	280	98,87
0	13	36,5	280	98,87
0	14	36,5	280	98,31
0	15	36,5	280	97,74

Tabela B.8 - Comparação entre os testes para janela de 36,5 ms, 280 vet./s e 5 reestimações.

n° do Teste	n° de Reest,	Janela (ms)	Vetores/s	Acertos (%)
0	5	36,5	280	99,44
1	5	36,5	280	95,38
2	5	36,5	280	94,59
3	5	36,5	280	97,59
4	5	36,5	280	72,01
5	5	36,5	280	78,57
6	5	36,5	280	44,9
7	5	36,5	280	80

Tabela B.9 - Análise relativa à variação da taxa de vetores/s, com janela de 36,5 ms e 5 reestimações.

Vetores/s	Acertos (%)	Vetores/s	Acertos (%)	Vetores/s	Acertos (%)	Vetores/s	Acertos (%)
20	81,92	120	94,35	220	97,18	320	96,05
25	84,18	125	94,35	225	97,18	325	94,92
30	87,57	130	96,61	230	97,74	330	95,48
35	88,7	135	96,05	235	96,61	335	96,61
40	87,57	140	96,61	240	97,18	340	96,61
45	87,57	145	94,92	245	96,61	345	94,92
50	89,27	150	96,61	250	97,18	350	95,48
55	89,83	155	97,74	255	96,61	355	94,92
60	92,66	160	96,05	260	97,74		
65	90,96	165	94,92	265	97,18		
70	89,83	170	96,05	270	97,74		
75	89,83	175	94,35	275	98,31		
80	90,4	180	95,48	280	99,44		
85	89,27	185	94,92	285	97,74		
90	92,66	190	94,92	290	98,31		
95	91,53	195	95,48	295	97,18		
100	92,66	200	96,05	300	98,87		
105	92,09	205	97,18	305	94,35		
110	94,92	210	96,05	310	97,74		
115	93,79	215	95,48	315	96,61		

Tabela B.10 - Comparação entre os diferentes tipos de codificação e seus percentuais de acerto para cada um dos testes.

Teste	MFCC_0_D_A	MFCC_E_D_A	LPC_E_D_A	MFCC_E_D	MFCC_0_D	LPC_E_D	MFCC_E_D_Z	MFCC_0_D_Z	LPC_E_D_Z
0	99,44	96,05	57,63	92,66	96,61	63,28	94,92	93,79	58,19
1	95,38	94,36	49,23	92,31	93,85	51,28	91,28	92,31	51,28
2	94,59	94,59	67,57	93,92	93,92	66,22	93,24	92,57	65,54
3	97,59	96,99	63,86	96,99	98,19	63,86	97,59	95,78	60,24
4	72,01	71,67	52,56	72,01	72,7	54,27	77,82	81,23	53,58
5	78,57	81,29	50,68	77,55	78,23	52,04	81,29	84,01	48,98
6	44,9	53,06	26,82	60,64	50,44	21,57	69,1	67,64	31,78
7	80	81,11	46,67	73,33	81,11	45,56	83,33	74,44	40

Tabela B.11 - Percentual de acertos para cada teste, considerando HMMs de 3, 5 e 7 estados, com 5 reestimações, 280 vetores de parâmetros por segundo e um comprimento de janela de 36,5 ms.

n° do teste	Acertos (%) 3 estados	Acertos (%) 5 estados	Acertos (%) 7 estados
0	89,27	99,44	88,7
1	89,23	95,38	88,21
2	80,41	94,59	85,14
3	90,96	97,59	92,17
4	72,01	72,01	60,41
5	74,49	78,57	72,45
6	36,44	44,9	46,36
7	52,22	80	76,67

Tabela B.12 – Comparação entre a taxa de acertos (%) (Acc) em função da variação do comprimento de janela (ms) (Jan), para 2 reestimações, “MFCC_E_D_Z”, 280 vetores de parâmetros por segundo.

Jan.	Acc	Jan.	Acc	Jan.	Acc	Jan.	Acc	Jan.	Acc	Jan.	Acc	Jan.	Acc	Jan.	Acc
28	68,8	48,5	69,68	69	66,18	89,5	65,6	110	62,97	130,5	62,97	151	60,64	171,5	59,77
28,5	69,1	49	70,26	69,5	66,18	90	65,6	110,5	62,68	131	62,1	151,5	60,06	172	59,77
29	69,68	49,5	69,39	70	66,18	90,5	65,89	111	62,97	131,5	61,81	152	60,64	172,5	60,06
29,5	69,39	50	70,26	70,5	66,47	91	65,31	111,5	62,97	132	61,22	152,5	60,93	173	59,77
30	69,97	50,5	69,68	71	66,76	91,5	65,6	112	63,27	132,5	61,81	153	61,22	173,5	59,77
30,5	69,1	51	69,68	71,5	67,64	92	65,89	112,5	62,97	133	61,81	153,5	60,64	174	59,18
31	69,1	51,5	69,68	72	65,89	92,5	65,6	113	62,97	133,5	62,1	154	60,93	174,5	59,77
31,5	68,8	52	69,97	72,5	66,18	93	65,31	113,5	62,1	134	61,22	154,5	60,64	175	60,06
32	68,8	52,5	69,68	73	67,35	93,5	65,6	114	62,68	134,5	61,22	155	60,35	175,5	60,35
32,5	68,8	53	69,1	73,5	67,93	94	65,31	114,5	63,27	135	61,81	155,5	60,64	176	60,64
33	68,51	53,5	69,1	74	67,06	94,5	65,01	115	62,97	135,5	60,64	156	60,35	176,5	60,35
33,5	68,51	54	69,68	74,5	66,76	95	65,01	115,5	62,68	136	62,1	156,5	60,06	177	60,35
34	68,22	54,5	69,1	75	67,06	95,5	64,43	116	63,27	136,5	61,22	157	60,35	177,5	60,06
34,5	67,35	55	68,22	75,5	66,18	96	65,01	116,5	62,97	137	61,22	157,5	60,64	178	60,06
35	68,22	55,5	68,51	76	67,06	96,5	64,72	117	62,39	137,5	60,06	158	60,64	178,5	60,64
35,5	67,64	56	68,51	76,5	67,35	97	64,72	117,5	63,56	138	61,22	158,5	61,52	179	60,35
36	68,51	56,5	68,8	77	66,47	97,5	65,01	118	63,27	138,5	61,52	159	60,93	179,5	60,64
36,5	68,8	57	68,51	77,5	67,93	98	64,43	118,5	63,85	139	61,22	159,5	60,93	180	61,22
37	68,51	57,5	68,51	78	67,06	98,5	65,01	119	64,43	139,5	61,52	160	60,93	180,5	60,35
37,5	67,64	58	67,06	78,5	67,64	99	65,31	119,5	63,85	140	61,22	160,5	61,52	181	60,64
38	68,51	58,5	67,93	79	67,35	99,5	64,14	120	63,27	140,5	61,81	161	60,93	181,5	61,52
38,5	68,22	59	66,47	79,5	68,22	100	64,14	120,5	63,27	141	60,64	161,5	61,22	182	61,22
39	69,1	59,5	66,47	80	67,06	100,5	64,72	121	62,97	141,5	61,22	162	61,52	182,5	61,52
39,5	68,22	60	68,22	80,5	65,89	101	63,27	121,5	63,27	142	60,35	162,5	60,93	183	61,81
40	68,22	60,5	67,64	81	65,89	101,5	64,72	122	62,68	142,5	60,93	163	60,93	183,5	60,93
40,5	68,51	61	67,64	81,5	66,76	102	63,56	122,5	62,68	143	61,22	163,5	60,93	184	61,22
41	68,51	61,5	67,64	82	65,89	102,5	63,85	123	62,39	143,5	60,93	164	60,93	184,5	60,06
41,5	68,8	62	67,35	82,5	66,18	103	63,56	123,5	62,68	144	60,64	164,5	61,22	185	60,35
42	69,1	62,5	67,64	83	66,76	103,5	63,56	124	62,68	144,5	62,39	165	60,93	185,5	60,64
42,5	69,1	63	67,06	83,5	66,47	104	64,43	124,5	62,97	145	61,52	165,5	60,64	186	61,22
43	69,68	63,5	67,06	84	66,18	104,5	64,43	125	61,52	145,5	61,22	166	60,64	186,5	60,93
43,5	69,39	64	67,35	84,5	66,47	105	63,56	125,5	62,39	146	60,93	166,5	60,64	187	60,93
44	69,68	64,5	67,06	85	66,18	105,5	63,85	126	62,1	146,5	61,22	167	61,22	187,5	60,93
44,5	70,26	65	66,76	85,5	65,6	106	64,14	126,5	61,81	147	61,52	167,5	60,64	188	60,06
45	70,26	65,5	67,35	86	65,31	106,5	63,85	127	62,39	147,5	60,93	168	60,93	188,5	60,93
45,5	69,97	66	67,06	86,5	66,47	107	62,68	127,5	61,22	148	60,93	168,5	61,22	189	60,93
46	70,26	66,5	67,06	87	65,89	107,5	63,56	128	61,52	148,5	61,22	169	60,64	189,5	59,48
46,5	69,97	67	67,06	87,5	65,6	108	63,27	128,5	61,22	149	60,35	169,5	60,06		
47	70,26	67,5	66,47	88	65,31	108,5	62,1	129	60,93	149,5	60,35	170	60,35		
47,5	69,97	68	65,89	88,5	66,18	109	62,97	129,5	62,1	150	60,35	170,5	60,35		
48	69,39	68,5	66,18	89	65,6	109,5	63,27	130	61,22	150,5	60,93	171	60,64		

Tabela B.13 – Variação do número de reestimações, com diferentes tamanhos de janela.

Número de reest.	Taxa de vetores/seg	Acertos (%) para janela de 36,5 ms	Acertos (%) para janela de 49 ms
1	280	56,56	54,23
2	280	68,8	69,68
3	280	67,64	68,8
4	280	67,93	69,68
5	280	69,1	71,43
6	280	70,85	74,64
7	280	73,47	74,93
8	280	74,05	76,97
9	280	75,51	77,55
10	280	75,8	77,26
11	280	75,22	76,68
12	280	74,64	76,68
13	280	74,64	76,38
14	280	74,93	76,68
15	280	74,93	76,97

Tabela B.14 – Variação da taxa de acertos em função da taxa de vetores por segundo no teste 6, para janela de 49 ms.

Vetores/s	Acertos (%)	Vetores/s	Acertos (%)	Vetores/s	Acertos (%)	Vetores/s	Acertos (%)	Vetores/s	Acertos (%)
5	33,82	110	62,68	215	72,59	320	72,01	425	65,31
10	42,27	115	63,85	220	76,09	325	72,89	430	66,47
15	46,36	120	65,31	225	73,76	330	70,26	435	62,39
20	51,31	125	63,85	230	75,22	335	65,89	440	62,39
25	50,15	130	64,72	235	73,76	340	70,26	445	61,22
30	56,27	135	63,85	240	75,22	345	69,39	450	61,81
35	57,73	140	64,72	245	73,18	350	66,18	455	59,77
40	57,14	145	62,39	250	75,51	355	68,8	460	62,39
45	58,6	150	64,14	255	75,22	360	69,97	465	63,27
50	61,81	155	62,97	260	75,8	365	64,14	470	62,68
55	59,18	160	62,1	265	76,09	370	67,35	475	61,22
60	59,48	165	64,43	270	74,64	375	65,89	480	62,97
65	60,93	170	65,31	275	73,18	380	66,47	485	61,81
70	61,52	175	65,6	280	77,26	385	64,72		
75	57,14	180	66,18	285	75,8	390	62,68		
80	58,89	185	67,93	290	75,8	395	66,47		
85	57,14	190	64,72	295	76,38	400	63,85		
90	59,77	195	69,97	300	73,76	405	66,18		
95	59,48	200	74,64	305	73,47	410	64,72		
100	62,39	205	71,72	310	73,47	415	64,72		
105	61,22	210	74,34	315	72,89	420	64,43		

Tabela B.15 – Variação do percentual de acertos em função do teste executado, considerando-se 9 reestimações, 280 vetores de parâmetros/s, janela de 49 ms, HMM de 5 estados, codificação “MFCC_E_D_Z”.

Comparação entre testes	
Número do teste	**Percentual de acertos**
0	94,92
1	92,31
2	93,92
3	97,59
4	75,77
5	82,65
6	77,26
7	82,22

APÊNDICE C – DETALHES ADICIONAIS SOBRE HMMs

C.1 TIPOS DE HMM

Algumas definições quanto aos tipos de HMMs devem ser descritas antes de prosseguir ao próximo capítulo.

Até o momento só foram consideradas HMMs ergódigas, ou seja, onde a partir de um estado, qualquer outro estado pode ser acessado através de uma transição para determinado estado. Este tipo de modelo poderia gerar uma matriz de probabilidades de transições do tipo:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \quad \text{(C.1)}$$

Para algumas aplicações, há modelos de HMM que se encaixem melhor do que o ergódigo, como por exemplo, o modelo abaixo, representado na Figura C.1:

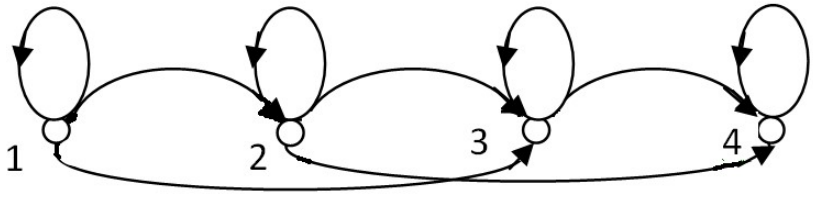

Figura C.1 - Modelo direita-esquerda de 4 estados.

Este modelo é chamado direita-esquerda ou modelo de Bakis e a sequência de estados associada a esse modelo tem a propriedade de aumentar o índice conforme o tempo passa (ou permanece inalterado), isto é, a sequência de estados se desloca da direita para a esquerda. O modelo de Bakis tem a propriedade de modelar facilmente sinais cujas propriedades mudam com o tempo como sinais de fala.

$$a_{ij} = 0, \quad j < i \tag{C.2}$$

No caso do modelo da Figura C.1, a matriz de transições possíveis seria da forma:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \tag{C.3}$$

É possível também considerar modelos nos quais as observações são associadas com os arcos do modelo. É útil em certas situações permitir transições que não produzem saídas (saídas vazias ø).

Há também a possibilidade de atribuir uma duração específica para cada um dos estados do modelo. A transição, então, é feita somente após um número suficiente de observações acontecerem neste estado. Assume-se que um total de *r* estados foram visitados durante o tempo *t* e denota-se estes estados como $q_1, q_2, \ldots, q$, com durações associadas a cada estado como sendo $\alpha_1, \alpha_2, \ldots, \alpha_t$. A duração de cada variável em HMM pode ser equivalente à HMM padrão se igualar $p_i(d)$ à:

$$p_i(d) = \left(a_{ij}\right)^{d-1}\left(1 - a_{ij}\right) \tag{C.4}$$
$$= probabilidade\ de\ observações\ consecutivas\ no\ estad \quad S_i$$

Define-se a variável $\alpha_t(i)$ como sendo:

$$\alpha_t(i) = P(O_1\ O_2\ O_3 \ldots O_t, Si\ terminar\ em\ t|\lambda) \tag{C.5}$$

O maior problema com os modelos direita-esquerda é que não se pode utilizar uma única sequência de observações para a reestimação paramétrica do modelo. Isso se deve à natureza transitiva dos estados no modelo que só permite um pequeno

número de observações para cada estado (até que uma transição leve a um estado sucessor).

Outra questão chave é decidir a estimação inicial dos parâmetros da HMM, de forma que o máximo local seja também o máximo global da função de verossimilhança. Esta função serve para maximizar o valor da probabilidade de encontrar uma sequência de símbolos observáveis, dado o modelo λ. De acordo com [31], a partir de experimentos, tanto para estimações aleatórias como "chutando" valores iniciais, a utilização de constantes uniformes para π e A é suficiente para uma boa reestimação desses parâmetros em quase todos os casos. No entanto, para os parâmetros B, experimentos mostraram que são importantes boas estimações iniciais no caso de símbolos discretos e essenciais no caso de distribuições contínuas.

Quando os dados são insuficientes para o treinamento, segundo [31], o aumento do tamanho do conjunto de treinamento de dados observáveis poderia solucionar o problema, mas normalmente isto se torna impraticável. Uma segunda maneira de solucionar esse caso seria reduzir o tamanho do modelo (redução do número de estados, número de observações na saída por estado, etc.), no entanto, mesmo que houvesse a possibilidade, há certas razões físicas que justificam a escolha do modelo, isto é, o fato de uma HMM ter tantos estados é devido à certa característica demandada. Um reconhecedor de fala normalmente utiliza HMMs de 5 estados, diminuir ou aumentar esse número pode alterar significativamente os resultados. A terceira possibilidade seria interpolar a estimação de um conjunto de parâmetros com um modelo no qual existe quantidade suficiente de dados.

Tendo estimativas para o modelo $\lambda = (A,B,\pi)$ e reduzindo-o para $\lambda' = (A',B',\pi')$, logo o modelo interpolado $\tilde{\lambda} = (\tilde{A}, \tilde{B}, \tilde{\pi})$ é obtido através de:

$$\tilde{\lambda} = \varepsilon \cdot \lambda + (1 - \varepsilon) \cdot \lambda' \qquad \text{(C.6)}$$

Onde ε representa uma ponderação dos parâmetros do modelo completo e (1 – ε) representa a ponderação dos parâmetros do modelo reduzido.

C.2 ANÁLISE LPC-*CEPSTRAL*

Uma das formas de obter um vetor de dados observados, a partir das amostras de fala utilizadas na entrada de um reconhecedor, é assumir que estamos processando apenas amostras correspondentes às palavras faladas, isto é, todo fundo anterior e posterior à palavra dita é eliminado por um algoritmo de detecção de palavras. O tipo de análise espectral utilizada neste caso é chamado *linear predictive coding* (LPC), ou codificação preditiva linear.

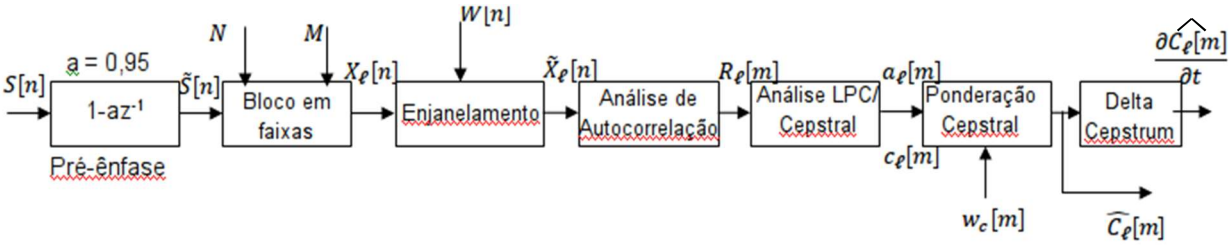

Figura C.2 - Diagrama de blocos representando a análise LPC/*Cepstral*.

O sinal de fala digitalizado *S[n]* é processado por uma rede digital de primeira ordem reduzir a largura de banda espectral do sinal, em seguida, seções de N_A amostras de fala consecutivas (N_A = 300 corresponde a 45 ms do sinal) são utilizadas como faixas individuais. Cada uma dessas pequenas faixas amostradas é multiplicada por uma janela de Hamming *w[n]* de forma a minimizar os efeitos adversos do cortar uma amostra fora do sinal de fala.

Em seguida, cada saída do enjanelamento é autocorrelacionado para fornecer um conjunto de coeficientes. Então, para cada uma desses quadros, um vetor de *(p+1)* coeficientes LPC/*Cepstral* é calculado usando métodos recursivos, onde *p* é a ordem da análise LPC desejada. Um vetor *Cepstral* derivado então é computado até o *Q*-ésimo componente, onde *Q* > *p*.

O vetor *Cepstral* de *Q* coeficientes então será ponderado por uma janela w_c[m]:

$$w_c(m) = 1 + \frac{Q}{2} \cdot sen(\frac{\pi \cdot m}{Q}), \quad 1 \leq m \leq Q \tag{C.7}$$

Na última etapa, a derivada temporal da sequência de vetores cepstrais é aproximada por um polinômio ortogonal de 1ª ordem sobre uma janela de comprimento finito de 2*K*+1 faixas centradas nas proximidades do vetor atual [31].