Universidade Federal do Pará
Centro Tecnológico
Programa de Pós-Graduação em Engenharia Elétrica

Chadia Nadim Aboul Hosn

CONVERSÃO GRAFEMA-FONE PARA UM SISTEMA DE RECONHECIMENTO DE VOZ COM SUPORTE A GRANDES VOCABULÁRIOS PARA O PORTUGUÊS BRASILEIRO

TM - 18 / 2006

UFPA – CT - PPGEE
Campus Universitário do Guamá
66.075-900 – Belém – Pará – Brasil

Universidade Federal do Pará
Centro Tecnológico
Programa de Pós-Graduação em Engenharia Elétrica

Chadia Nadim Aboul Hosn

CONVERSÃO GRAFEMA-FONE PARA UM SISTEMA DE RECONHECIMENTO DE VOZ COM SUPORTE A GRANDES VOCABULÁRIOS PARA O PORTUGUÊS BRASILEIRO

Dissertação submetida à Banca Examinadora do Programa de Pós-Graduação em Engenharia Elétrica da UFPA para a obtenção do Grau de Mestre em Engenharia Elétrica

UFPA – CT - PPGEE
Campus Universitário do Guamá
66.075-900 – Belém – Pará – Brasil

Universidade Federal do Pará
Centro Tecnológico
Programa de Pós-Graduação em Engenharia Elétrica

CONVERSÃO GRAFEMA-FONE PARA UM SISTEMA DE RECONHECIMENTO DE VOZ COM SUPORTE A GRANDES VOCABULÁRIOS PARA O PORTUGUÊS BRASILEIRO

AUTORA : CHADIA NADIM ABOUL HOSN

DISSERTAÇÃO SUBMETIDA À AVALIAÇÃO DA BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA DA UNIVERSIDADE FEDERAL DO PARÁ E JULGADA ADEQUADA PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA COM ÊNFASE EM TELECOMUNICAÇÕES.

APROVADA EM 12 / 06 / 2006

BANCA EXAMINADORA:

_____
Prof. Dr. Aldebaro Barreto da Rocha Klautau Júnior
(ORIENTADOR - UFPA)

_____
Prof. Dr. Evaldo Gonçalves Pelaes
(EXAMINADOR – UFPA)

_____
Profa. Dra. Valquíria Gusmão Macêdo
(EXAMINADOR – UFPA)

_____
Prof. Dr. Antônio Marcos de Lima Araújo
(EXAMINADOR - IESAM)

VISTO:

_____
Prof. Dr. Evaldo Gonçalves Pelaes
(COORDENADOR DO PPGEE - UFPA)

UFPA – CT - PPGEE
Campus Universitário do Guamá
66.075-900 – Belém – Pará – Brasil

A morte do homem começa no instante em que ele desiste de aprender.

(Albino Teixeira)

Dedicatória:

Ao meu querido avô Mohamad, por ser sempre a minha fonte de inspiração e aos meus pais, pelo amor, incentivo e educação ao longo de minha vida.

AGRADECIMENTOS

A Deus, pelos caminhos guiados, pelas conquistas e derrotas, pelas pessoas que pôs em minha vida e pelo que sou hoje.

Ao meu orientador Prof. Dr. Aldebaro Klautau Jr., meu grande agradecimento pela orientação, dedicação, paciência, encorajamento, compreensão e discussões técnicas sem as quais não seria possível a realização deste trabalho.

Aos meus pais, Nadim e Afaf, e irmãos Raid e Nadia, mesmo eu não possuindo palavras suficientes para expressar o fiel reconhecimento, agradeço pela criação, dedicação, carinho, diálogo, amor, respeito, correções, conselhos e incentivo ao longo de minha vida.

Ao meu noivo Karim, companheiro, amigo, confidente e que, mesmo de longe, me apoiou e incentivou para a conclusão deste trabalho.

Aos amigos Luiz Alberto Novaes e Tales Imbiriba pela paciência em esclarecer minhas dúvidas.

Aos amigos e companheiros de Laboratório de Processamento de Sinais (LaPS) Yomara, Adalbery, Ênio, Gabrielle, Fabiola e a todos os outros, pela saudável convivência, apoio em todas as fases do projeto.

Aos professores do DEEC.

A todos aqueles que não foram citados aqui, mas que contribuíram de alguma forma para a conclusão deste trabalho e aquisição deste título, venho aqui expressar o meu muito obrigada.

RESUMO

O processamento de voz tornou-se uma tecnologia cada vez mais baseada na modelagem automática de vasta quantidade de dados. Desta forma, o sucesso das pesquisas nesta área está diretamente ligado a existência de corpora de domínio público e outros recursos específicos, tal como um dicionário fonético. No Brasil, ao contrário do que acontece para a língua inglesa, por exemplo, não existe atualmente em domínio público um sistema de Reconhecimento Automático de Voz (RAV) para o Português Brasileiro com suporte a grandes vocabulários. Frente a este cenário, o trabalho tem como principal objetivo discutir esforços dentro da iniciativa FalaBrasil [1], criada pelo Laboratório de Processamento de Sinais (LaPS) da UFPA, apresentando pesquisas e softwares na área de RAV para o Português do Brasil. Mais especificamente, o presente trabalho discute a implementação de um sistema de reconhecimento de voz com suporte a grandes vocabulários para o Português do Brasil, utilizando a ferramenta HTK baseada em modelo oculto de Markov (HMM) e a criação de um módulo de conversão grafema-fone, utilizando técnicas de aprendizado de máquina.

Palavras chave: Conversão Grafema-Fone, Reconhecimento Automático de Voz, Modelos Ocultos de Markov, Aprendizado de Máquina.

ABSTRACT

Speech processing has become a data-driven technology. Hence, the success of research in this area is linked to the existence of public corpora and associated resources, as a phonetic dictionary. In contrast to other languages such as English, one cannot find, in public domain, a Large Vocabulary Continuos Speech Recognition (LVCSR) System for Brazilian Portuguese. This work discusses some efforts within the FalaBrasil initiative [1], developed by researchers, teachers and students of the Signal Processing Laboratory (LaPS) at UFPA, providing an overview of the research and softwares related to Automatic Speech Recognition (ASR) for Brazilian Portuguese. More specifically, the present work discusses the implementation of a large vocabulary ASR for Brazilian Portuguese using the HTK software, which is based on hidden Markov models (HMM). Besides, the work discusses the implementation of a grapheme-phoneme conversion module using machine learning techniques.

Keywords: Grapheme-Phoneme Conversion, Automatic Speech Recognition, Hidden Markov Models, Machine Learning.

SUMÁRIO

RESUMO ..... **iv**
ABSTRACT ..... **v**
SUMÁRIO ..... **vi**
LISTA DE FIGURAS ..... **viii**
LISTA DE TABELAS ..... **x**

1. INTRODUÇÃO **1**
1.1. Considerações Iniciais 1
1.2. Objetivos do Trabalho 2
1.3. Estrutura da Tese 3

2. RECONHECIMENTO AUTOMÁTICO DE VOZ (RAV) **4**
2.1. Introdução 4
2.2. Breve Histórico das Pesquisas em Reconhecimento de Voz 4
2.3. O Problema do Reconhecimento de Voz 7
2.4. Aplicações de sistemas RAV 10
2.4.1. Interface para Computadores Pessoais 10
2.4.2. Sistemas de Ditado de Grande Vocabulário 11
2.4.3. Sistemas Baseados em Rede Telefônica 12
2.4.4. Aplicações Industriais e Sistemas Integrados 13
2.5. O Futuro do RAV 13
2.5.1. O Futuro de Acordo com Ray Kurzweil 13
2.5.2. Opinião da Comunidade 14

3. CONVERSÃO GRAFEMA-FONE **20**
3.1. Introdução 20
3.2. Breve Histórico 20
3.3. Conceitos importantes 21
3.3.1. Fonética e Fonologia 21
3.3.2 Alfabeto Fonético 22
3.3.3. Representação de Caracteres e Símbolos 23

3.4 Considerações Sobre a Língua Portuguesa 23
3.5. Sistema de Aprendizado de Pronúncias 24
3.5.1 Weka 25
3.5.2. Dicionário UFPAdic1.0 28
3.5.3. Módulos do conversor grafema-fone 28
3.6. Resultado das simulações 37
3.7. Construção do Dicionário Fonético 39

4. LVCSR PARA O PORTUGUÊS BRASILEIRO 40

4.1. Introdução 40
4.2. Blocos Componentes de um sistema de RAV 40
4.3. Implementação do Sistema LVCSR para o Português Brasileiro 42
4.3.1. Descrição do Corpus Spoltech Versão 1.0 43
4.3.2. Dicionário Fonético 45
4.4. HTK - The Hidden Markov Model Toolkit 45
4.4.1. Análise do Sinal 47
4.4.2. Treinamento do Sistema 49
4.4.3 Reconhecimento e Medidas de Desempenho 53
4.5 Resultado das Simulações 57

5. CONCLUSÃO 60

5.1. Discussão Geral 60
5.2. Sugestões para Trabalhos Futuros 61

REFERÊNCIAS BIBLIOGRÁFICAS 63

Apêndice 1 - Alfabeto Fonético Internacional **67**

Apêndice 2 - Tabela ASC II e Descrição **68**

Apêndice 3 – Passos Realizados para Conversão Grafema-Fone **70**

Apêndice 4 – Mapeamento UFPAdic1.0 - SAMPA **71**

Apêndice 5 – Passos da implementação do LVCSR para o PB, usando o *corpus* Spoltech **..74**

LISTA DE FIGURAS

Figura 2.1 - Progresso no RAV. **6**
Figura 2.2 - Precisão do reconhecimento feito por máquina x homem. **7**
Figura 2.3 - Respostas à declaração "A maioria dos telefones com resposta interativa a voz aceita a voz como entrada (além de apenas dígitos)". **15**
Figura 2.4 - Respostas ao evento "Reconhecimento de Voz está geralmente disponível em casa (por exemplo, na televisão interativa, no controle de eletrodomésticos e em sistemas de gerenciamento da casa)". **16**
Figura 2.5 - Respostas à declaração "Telefones tradutores permitem duas pessoas pelo globo conversar sem a necessidade de falarem o mesmo idioma". **17**
Figura 2.6 - Respostas à afirmação "Interações com computador feita por gestos e comunicação natural em duplo sentido". **17**
Figura 2.7 - Respostas ao evento "Não haverá mais necessidade de pesquisas em reconhecimento de voz". **18**
Figura 2.8 - Taxa de erro sugerida com o aumento da quantidade de dados de treino. **19**

Figura 3.1 - Matriz de 'associação' B, indexada por letras da palavra soletrada "phase" e fonemas de sua pronunciação "feIz". **30**
Figura 3.2 - Superposição das matrizes C e D. Contém valores acumulados pelas associações através da PD, juntamente com os ponteiros indicando o movimento percorido para chegar à célula atual, tal que maximize o valor da associação. **31**
Figura 3.3 – Variação de contexto, arquivo ARFF. **33**
Figura 3.4 - Esquema de treino do conversor grafema-fone. **36**
Figura 3.5 - Esquema de transcrição realizada pelo conversor grafema-fone. **37**

Figura 4.1 - Blocos componentes de um sistema de reconhecimento de voz (RAV). **41**
Figura 4.2 - Processo de parametrização. **41**
Figura 4.3 - Estrutura left-right. para um HMM com 3 estados. **46**
Figura 4.4 - Fundamentos do HTK. **47**
Figura 4 5 - Funcionamento do HCopy. **47**
Figura 4 6 - Inicialização e treino dos modelos HMMs. **50**

Figura 4 7 - Funcionamento do HERest. **.....53**
Figura 4.8 - Funcionamento do HVite e HResults. **.....55**
Figura 4.9 - Consertando o modelo de silêncio. **.....56**
Figura 4.10 - Caracterização do HHEd. **.....56**
Figura 4.11 - Taxa de erro por palavra WER (%) variando o número de gaussianas por mixtura para treino das HMMs. **.....58**

LISTA DE TABELAS

Tabela 2.1 - Parâmetros que caracterizam a capacidade de um sistema RAV. **.....08**

Tabela 3.1 - Número de fonemas e palavras usadas para treino e teste dos dicionários. **.....32**

Tabela 3.2 - Taxa de erros do dicionário UFPAdic 1.0 variando o contexto de expansão**.....38**

Tabela 3.3 - Taxa de erros obtidas para o dicionário NetTalk.**.....38**

Tabela 3.4 - Precisão da transcrição obtida pelo Brulex. **.....38**

Tabela 3.5 - Precisão da tanscrição obtida pelo BEEP 1.0.**.....38**

Tabela 4.1 – Desempenho do LVCSR para o PB baseado em monofones de acordo com o número de reestimações e variação do número de gaussianas por mixtura. **.....58**

1. INTRODUÇÃO

1.1. Considerações Iniciais

A linguagem falada é a forma mais usada de comunicação entre os seres humanos. Devido à capacidade do nosso cérebro em interpretar informações complexas, podemos, de forma praticamente inconsciente, reconhecer quem está falando (emissor da mensagem), sua posição no espaço físico, seu estado emocional e outros dados que podem estar escondidos no tom de voz usado (ironia, seriedade ou tristeza, por exemplo). Neste cenário se insere o Reconhecimento Automático de Voz (RAV) com o objetivo de desenvolver máquinas que tenham um desempenho similar ao sistema auditivo humano.

Nos últimos anos temos assistido a um desenvolvimento significativo de sistemas RAV, tanto pelos principais institutos de pesquisas quanto pelas empresas que têm colocado no mercado um conjunto de produtos extremamente interessantes. A utilização da fala nos sistemas computacionais segue uma tendência natural que visa tornar a interação homem-máquina mais direta e efetiva. Tais sistemas proporcionam aplicações como: discagem através da fala em telefones celulares, sistemas de tradução automática, acesso remoto a base de dados, automação doméstica, comunicações *hands-free* e *eyes-free* com computadores e outras máquinas, entre outros.

Há muitos grupos de pesquisa nos cinco continentes abordando o reconhecimento de voz de diferentes maneiras. No Brasil, ao contrário do que acontece para a língua inglesa, por exemplo, não existe atualmente em domínio público um sistema de reconhecimento automático de voz para o Português Brasileiro (PB), com suporte a grandes vocabulários, ou seja, com mais de 30 mil palavras. As dificuldades para o desenvolvimento destes sistemas podem ser aglutinadas ao redor de duas lacunas: um *corpus* de voz digitalizado e transcrito, grande o suficiente para o treinamento de modelos acústicos e de recursos específicos, tal como um dicionário fonético de grande vocabulário. Sistemas de reconhecimento de voz com suporte a grandes vocabulários

são conhecidos na literatura como LVCSR do inglês *Large Vocabulary Continuos Speech Recognition*.

A criação de um dicionário fonético pode ser feita manualmente com auxílio de lingüistas, mas à medida que os dicionários tornam-se maiores a transcrição de palavras de forma manual torna-se um método não sustentável, pois além de consumir tempo requer um bom conhecimento na área de lingüística. Desta forma, surge a necessidade de realizar transcrições de forma automática, e para tal finalidade são usados os sistemas de conversão grafema-fone.

A fonetização, i.e., a conversão de seqüências de caracteres em seqüências de fones comporta algumas dificuldades, pois o mesmo símbolo ortográfico pode corresponder a vários fones ou simplesmente não ter realização oral, assim como o mesmo som é, muitas vezes, representado na escrita de forma diversa.

O grau de facilidade em desenvolver um algoritmo preciso para conversão grafema-fone é diretamente proporcional a regularidade existente entre a grafia e fonia, assim como a complexidade alofônica de uma determinada língua. Para línguas como Inglês e Francês, por exemplo, esta conversão chega a ser um grande desafio. Devido a pouca regularidade existente entre a escrita e a pronúncia, a transcrição fonética torna-se estritamente dependente do léxico e cheia de “exceções”.

1.2. Objetivos do Trabalho

Este trabalho discute esforços dentro da iniciativa FalaBrasil [1], que tem como principal objetivo apresentar pesquisas e *softwares* na área de RAV para o Português do Brasil e, desta forma, estabelecer um ponto de partida para estes sistemas permitindo a reprodução dos resultados em diferentes localidades. Mais especificamente, este trabalho tem como objetivo a implementação de um sistema LVCSR para o Português Brasileiro e a criação do módulo de conversão grafema-fone, o qual será utilizado para obtenção do dicionário fonético de grande vocabulário do PB.

Para processamento da primeira tarefa será utilizado o *corpus* Spoltech, resultado de um projeto de cooperação entre o Brasil e os Estados Unidos para o avanço da tecnologia da linguagem falada, dentro da área de reconhecimento e síntese de voz. O *software* utilizado para desenvolver o sistema foi o HTK, baseado em modelo oculto de Markov (HMM) e capaz de processar rotinas desde a análise do sinal de voz até a fase de cálculo do desempenho do sistema.

Quanto às técnicas usadas para criação do módulo de conversão grafema-fone, optou-se por utilizar o método de aprendizado de máquina, capaz de extrair regras de transcrição a partir de um léxico. Em particular, serão usadas técnicas de aprendizagem por indução (algoritmo J48) e *Naïve Bayes*, disponíveis no *software* Weka. A criação deste sistema permitirá a transcrição de forma automática de um grande número de palavras, e desta forma, será possível criarmos o dicionário fonético de grande vocabulário para o PB, necessário para desenvolvimento do sistema LVCSR proposto.

1.3. Estrutura da Tese

O presente trabalho está organizado em 5 capítulos, divididos da seguinte forma: o Capítulo 1 corresponde às considerações iniciais e os objetivos do trabalho. No Capítulo 2 é feito um breve histórico das pesquisas em RAV; abordam-se os principais problemas na construção destes sistemas e são apresentadas algumas aplicações e esperanças futuras para a área. No Capítulo 3 é apresentado o estudo sobre sistemas de conversão grafema-fone; e a metodologia usada para desenvolvimento de um sistema de conversão garfema-fone, utilizando técnicas de aprendizado de máquina. O Capítulo 4 apresenta os blocos necessários para o desenvolvimento de sistemas RAV; a metodologia usada para contrução de um sistema LVCSR para o Português Brasileiro, utilizando o *software Hidden Markov Model Toolkit* (HTK) para modelagem do sistema. Finalmente, o Capítulo 5 apresenta as conclusões e sugestões para trabalhos futuros.

2. RECONHECIMENTO AUTOMÁTICO DE VOZ (RAV)

2.1. Introdução

O reconhecimento automático de voz (*Automatic Speech Recognition* – ASR) é a tarefa de traduzir um sinal acústico (analógico) representando a fala de uma pessoa em uma representação textual que corresponda àqueles sons. O progresso desta tecnologia tem permitido alcançar níveis de desempenho satisfatórios em uma ampla gama de aplicações, envolvendo desde o emprego de pequenos vocabulários para o reconhecimento através de linhas telefônicas até a compreensão espontânea da fala [2].

Neste capítulo é apresentado um breve histórico das pesquisas em reconhecimento de voz; os principais problemas para construção destes sistemas; algumas aplicações e esperanças futuras para esta área.

2.2. Breve Histórico das Pesquisas em Reconhecimento de Voz

Pesquisas no reconhecimento de voz por máquinas têm sido feitas por mais de quatro décadas e a sua história está inteiramente relacionada com a história dos microcomputadores e evolução do processamento digital de sinais. Para ter noção do tamanho do progresso obtido, vale a pena fazer um breve resumo de algumas das pesquisas mais importantes.

No início dos anos 50 ocorreu o primeiro esforço em arquitetar um sistema de reconhecimento de voz. Nos laboratórios Bell, em 1952, Davis, Biddulph, e Balashek construíram um sistema para reconhecimentos de dígitos isolados para apenas um falante. Em 1959, no colégio universitário na Inglaterra, Fry e Denes, tentaram construir um reconhecedor de fonema para reconhecer 4 vogais e 9 consoantes. Eles usaram um analisador espectral e um verificador de padrão para fazer a decisão de reconhecimento.

Outro equipamento interessante criado nesta época foi o reconhecedor vocálico de Forgie & Forgie, construído nos laboratórios da MIT (*Massachusetts Institute of Technology*) em 1959, no qual 10 vogais (língua inglesa) eram reconhecidas em um estilo independente de fala.

Nos anos 60 foram publicadas idéias fundamentais no reconhecimento da voz. Em 1963 houve a criação do hardware reconhecedor de dígito de Nagata e companheiros nos laboratórios NEC (Nippon Eletric Corporation) do Japão que foi, talvez, o mais notável da época e tornou possível um programa de pesquisa longo e altamente produtivo.

Outros projetos importantes ocorreram como o trabalho de Martin e seus colegas nos laboratórios RCA, em Nova Jersey, buscavam desenvolver soluções realísticas para problemas associados com sinais não uniformes no tempo, o caso da voz. No mesmo período, na União Soviética, Vintsyuk, propôs o uso de métodos de programação dinâmica para alinhamento no tempo de um par de expressões vocais (declarações), técnica esta não muito estudada antes da década de 80.

Nos anos 70, a área de reconhecimento de palavras isoladas ou pronúncias discretas tornou-se uma tecnologia viável e usável baseada em estudos fundamentais de Velichko e Zagoruyko na Rússia, Sakoe e Chiba, no Japão, e Itakura, nos Estados Unidos. Estudos russos ajudaram a avançar o uso de idéias de reconhecimento de padrão no reconhecimento de voz. Reddy (1975), através de uma revista da IEEE, publicou uma série de artigos acadêmicos a respeito do tema. Idéia seguida por Hanton (1982), que publicou um livro intitulado "*Automatic Speech Analysis and Recognition*". Entretanto, muitas das propostas eram quase impraticáveis, nem tanto pelas idéias, mas pela tecnologia disponível na época e pelo seu difícil acesso.

Os anos foram passando e muitos problemas tecnológicos foram sendo superados. No final da década de 80, o mundo, finalmente, conheceria os microcomputadores, os disquetes, os *winchesters* e, também, a conversão analógico-digital, tecnologia proporcionada por placas dedicadas e recurso especial para o processamento digital de sinais. Essa revolução não foi diferente para o reconhecimento

de voz, de modo que, a globalização e a popularização dos computadores aumentaram significativamente o número de pessoas pesquisando técnicas e tecnologias diversas.

Da mesma maneira que o reconhecimento de uma palavra isolada foi um foco chave da pesquisa na década de 70, o problema de reconhecimento de palavra conectada foi um foco da pesquisa da década de 80, onde pesquisadores iniciaram uma série de experimentos visando a produção de sistema de reconhecimento de voz que foram verdadeiros *speakers* independentes.

O final da década de 80 foi caracterizado por uma mudança tecnológica para formas apropriadas de métodos de modelos estatísticos, especialmente o hidden Markov model, usado para modelar sinais variantes no tempo, como a voz. Outra tecnologia reintroduzida na década de 80 foi a idéia de aplicações de redes neurais para problemas de reconhecimento da voz. Redes neurais foram introduzidas na década de 50, mas não foram aprovadas, inicialmente, por terem muitos problemas práticos. Na década de 80, entretanto, uma compreensão mais profunda do poder e das limitações desta tecnologia foi obtida, assim como, a relação desta tecnologia com os métodos clássicos de classificação de sinal.

Os progressos na área de voz são evidentes e diferentes tipos de sistema foram e estão sendo implementados para reconhecimento automárico de voz. A Figura 2.1 mostra a variação da taxa de erro por palavra com a evolução da tecnologia.

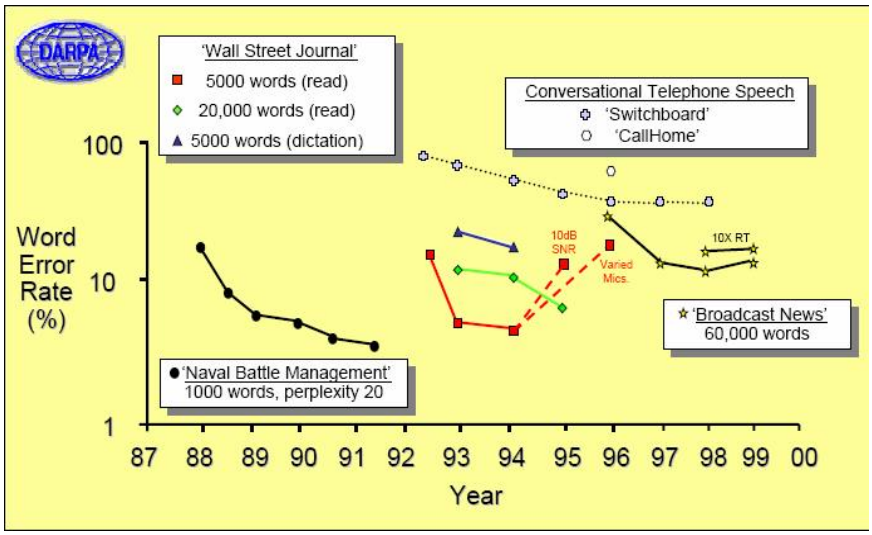


Figura 2.1 - Progresso no RAV. Prof. Roger K Moore, Keynote Talk, SPECOM, Patras, 2005.

Uma pesquisa realizada por Richard Lippmann em 1997 comparava-se a precisão do reconhecimento feito por máquinas (ASR) com o feito por humanos (Human Speech Recognition - HSR) sobre alguns tipos de sistemas RAV. O resultado indicando a taxa de erro por palavra mostrou o reconhecimento feito por máquina com desempenho limitado comaparado ao sistema auditivo, Figura 2.2.

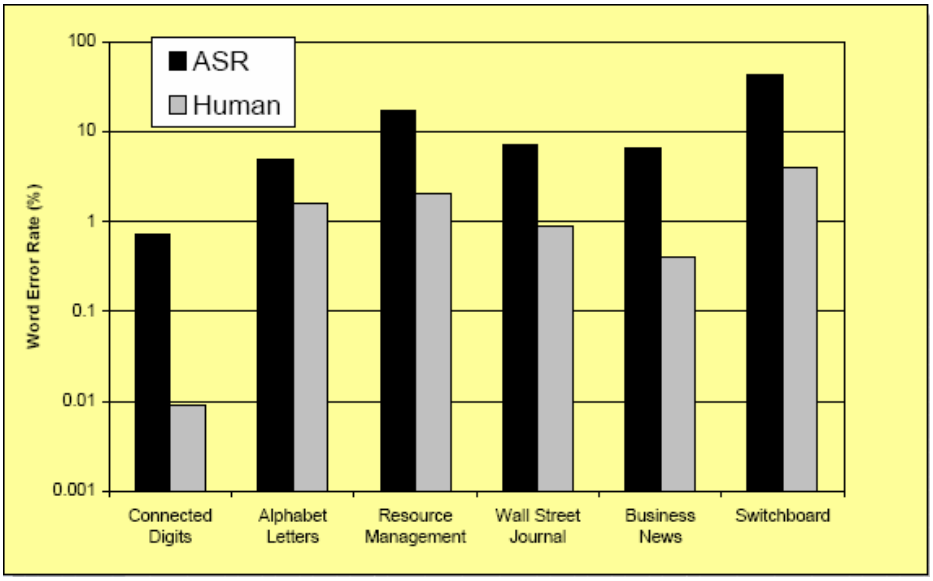


Figura 2.2 - Precisão do reconhecimento feito por máquina x homem. Tirada do texto '*Speech Recognition by Machines and Humans'*, R. Lippmann, Speech Communication, 1997.

2.3. O Problema do Reconhecimento de Voz

Para os seres humanos reconhecer a fala é uma tarefa considerada "simples" e "natural". No entanto, não se pode dizer o mesmo para os computadores. Fazer um computador responder a um comando falado é uma tarefa extremamente difícil e complexa. As principais dificuldades no reconhecimento de voz podem ser resumidas nos seguintes aspectos:

- A mesma palavra pronunciada pode apresentar diferentes formas de onda devido à articulação dos órgãos do aparelho fonador. Por exemplo, o fonema /t/ em tatu tem uma articulação puramente oclusiva, e em tia, dependendo do locutor, pode

ter uma articulação africada, onde à oclusão se segue um ruído fricativo semelhante ao do início da palavra “chuva”;

- As dificuldades na segmentação da fala: não se tem precisamente uma forma de limitação dos fonemas (menor unidade da fala), dificultando o reconhecimento de voz contínua. Esta imprecisão do limite advém da grande variação dos sinais de voz e a iteração mútua entre eles. Por exemplo, nas fronteiras entre palavras, as variações contextuais podem tornar-se bem mais acentuadas fazendo, por exemplo, com que a frase ‘a justiça é ...’ seja pronunciada como ‘ajusticé...’;
- As variações nas características da fala: pode haver diferenças acústicas não-lineares no tempo (ritmo), em freqüência (timbre), e em amplitude (intensidade);
- Com insuficiente uso do conhecimento lingüístico: a fala pode não conter toda a informação lingüística (como por exemplo: erros de Português e sotaque).

A Tabela 2.1 mostra os parâmetros mais importantes que caracterizam a capacidade de um sistema RAV.

Parâmetros	**Faixa**
Modo de pronúncia	De palavras isoladas a fala continua
Estilo de pronúncia	De leitura a fala espontânea
Treinamento	De dependente de locutor a independente de locutor
Vocabulário	De pequeno (< 20 palavras) a grande (> 20000 palavras)
Modelo de linguagem	De estados finitos a sensível a contexto
Perplexidade	De pequena (< 10) a grande (> 100)
Razão Sinal Ruído (SNR)	De alta (> 30 dB) a baixa (< 10 dB)
Transdutor	De microfone com cancelamento de ruído a telefone celular

Tabela 2.1 - Parâmetros que caracterizam a capacidade de um sistema RAV.

- Modo de Pronúncia: é possível reconhecer palavras isoladas ou fala contínua. No primeiro caso é necessário um período mínimo de silêncio entre as palavras pronunciadas e no segundo esta restrição não é aplicada.

- Estilo de Pronúncia: sentenças lidas são adequadas ao treinamento de sistemas de reconhecimento de voz bem articulada e pronunciada sem hesitações. Sistemas visando o reconhecimento de voz espontânea devem ser treinados com o uso de bases de fala espontânea.
- Treinamento: se o sistema somente reconhece a voz dos locutores para o qual foi treinado, tal sistema é dependente do locutor. O sistema é independente do locutor quando é capaz de reconhecer qualquer locutor que não tenha sido treinado.
- Vocabulário: o tamanho do vocabulário influencia a precisão do sistema de reconhecimento. Isto ocorre devido a possível ambigüidade das palavras (palavras semelhantes para o algoritmo classificador). O reconhecimento torna-se mais difícil à medida que o vocabulário cresce, ou apresenta palavras parecidas.
- Modelo de Linguagem: quando a fala é produzida em seqüências de palavras, são usados modelos de linguagem para restringir as possibilidades de seqüências de palavras. O modelo mais simples pode ser definido como uma máquina de estados finita, onde são explicitadas as palavras que podem seguir uma dada palavra. Os modelos de linguagem mais gerais, que se aproximam da linguagem natural, são definidos em termos de gramáticas sensíveis a contexto.
- Perplexidade: uma medida popular da dificuldade da tarefa, que combina o tamanho do vocabulário e o modelo de linguagem, é a perplexidade, grosseiramente definida como a média do número de palavras que pode seguir uma palavra depois que o modelo de linguagem foi aplicado.
- SNR (*Signal-to-Noise-Ratio*) e Transdutor: indicam parâmetros externos que podem afetar o desempenho de um sistema de reconhecimento de voz como, características do ruído ambiente e o tipo e posição do microfone. Trabalhando-se fora do ambiente de laboratório a influência do ruído torna-se inevitável, o qual pode aparecer de diversas formas possíveis, desde vozes de outros locutores, sons de equipamentos, ar condicionado, luz fluorescente e, até mesmo, provocado pelo próprio locutor, tais como tosses, espirros, estalo dos lábios, suspiro, respiração forte, etc. A falta de robustez dos sistemas RAV a

perturbações que afetam o sinal acústico continua a ser uma limitação muito grave, o que tem sugerido abordagens diversas [3].

2.4. Aplicações de sistemas RAV

Sendo em casa, no trabalho ou num jogo, a maioria dos seres humanos administra suas atividades diárias interagindo com outras pessoas por meio da fala, desde uma simples fofoca a transações comerciais complexas, de negociações empresariais a diplomacia internacional. O surgimento desta tecnologia, aliada a pressão econômica para redução de custos envolvidos com atendimento ao consumidor tem dado origem ao desenvolvimento de um grande mercado para a tecnologia de fala. Algumas áreas de aplicação comercial são: interfaces para computadores pessoais, ditados, serviços de telefonia automáticos e aplicações industriais especiais [4]. O principal motivo do sucesso comercial tem sido o aumento na produtividade proporcionado por estes sistemas que auxiliam ou substituem operadores humanos.

2.4.1. Interface para Computadores Pessoais

A fala tende a se tornar uma componente importante na interface com os computadores. Algumas das possíveis aplicações poderiam ser:

- Fala como atalho: ao invés de abrir um arquivo através de vários níveis de hierarquia, o usuário apenas diz “Abra o relatório”.
- . Recuperação de informação: interfaces gráficas são inconvenientes para especificar recuperação de informações baseada em restrições (“encontre todos os documentos de Carlos criados depois de abril”)
- Computadores de bolso: à medida que o tamanho dos computadores diminui (hoje existem palm-tops minúsculos), teclados e mouses tornam-se cada vez mais difíceis de usar, tornando a fala uma alternativa bastante atraente.

Embora o reconhecimento de voz em computadores seja uma alternativa bastante atraente, as interfaces atuais, teclado e mouse, representam uma alternativa madura e extremamente eficiente. É improvável que a fala possa substituir completamente estes dispositivos. Ao invés disso, a nova interface deve combinar estes dispositivos e permitir que o usuário defina qual combinação de dispositivos é a mais adequada para determinada tarefa.

O uso apropriado da fala nos computadores pessoais irá provavelmente requerer o desenvolvimento de um novo conceito de interação com o usuário ao invés de simplesmente modificar as interfaces gráficas existentes.

Uma questão social também está envolvida neste tipo de interface: a dos deficientes físicos. Com interface via voz, pessoas impossibilitadas de usar o computador, por causa de deficiências, seriam capazes de utilizá-lo normalmente, permitindo seu ingresso ao mercado de trabalho e uma competição em pé de igualdade com as outras pessoas.

2.4.2. Sistemas de Ditado de Grande Vocabulário

Os sistemas de ditado de vocabulário extenso podem ser de dois tipos: ditado irrestrito (por exemplo, cartas de negócios ou artigos de jornais) e geração de documentos estruturados (por exemplo, receitas médicas, apólices de seguro, relatórios radiológicos, etc).

Até bem pouco tempo atrás, os sistemas de palavras isoladas predominaram no mercado. Agora, sistemas de reconhecimento de voz contínua começam a aparecer. Os vocabulários são de aproximadamente 60000 palavras. Estes sistemas são projetados para operar em condições favoráveis (por exemplo, em escritórios, com microfones fixos na cabeça do operador e com cancelamento de ruído).

Para aumentar a taxa de acertos, os sistemas de ditado irrestrito contam com modelos de linguagem estatísticos para favorecer palavras ou seqüências de palavras

mais freqüentes. Os sistemas de domínio específico podem aumentar o desempenho incorporando um padrão de documento estruturado para gerar um relatório completo, embora muitas vezes isto exija um processo de planejamento bastante minucioso.

Um sistema de ditado torna-se mais poderoso se possui a habilidade de se adaptar à voz de um determinado usuário (adaptação ao locutor), vocabulário (aprendizagem de novas palavras), e tarefas (adaptação do modelo de linguagem).

2.4.3. Sistemas Baseados em Rede Telefônica

O reconhecimento de voz baseado na rede telefônica oferece um potencial enorme por ser um meio de comunicação extremamente difundido. É também a área tecnicamente mais difícil para o reconhecimento devido à impossibilidade de controle sobre as condições de uso.

Além do pouco controle sobre a qualidade do sinal, o reconhecimento através da linha telefônica apresenta problemas devido à expectativa dos usuários que o sistema se comporte como um interlocutor humano. Dois exemplos clássicos seriam:

- Usuário fala enquanto o sistema ainda está formulando as questões (intromissão), de modo que na hora em que o sistema entra em modo de gravação para coletar a resposta, o usuário já está no meio da resposta ou já terminou de falar.
- Usuário adiciona palavras à resposta, que não estão no vocabulário do sistema (“sim, por favor”). Neste caso podem ser usadas técnicas de identificação de palavras para conseguir taxas de reconhecimento aceitáveis.

Estes serviços de operação envolvem vocabulários pequenos, diálogo interativo e avisos. As possíveis aplicações seriam: reservas para hotéis, restaurantes, teatros, passagens aéreas, consultas a telefones e etc.

2.4.4. Aplicações Industriais e Sistemas Integrados

Os sistemas de reconhecimento de voz também podem ser utilizados em aplicações mais simples de vocabulário restrito, como o controle de máquinas e dispositivos, abertura e fechamento de portas e válvulas, acendimento de luzes, operações financeiras e outros. 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.

Por outro lado, 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.

Os exemplos de aplicações, mencionados anteriormente, significam uma nova era na interação homem-máquina, onde a tecnologia procura criar interfaces, cada vez mais, amigáveis ao homem.

2.5. O Futuro do RAV

2.5.1. O Futuro de Acordo com Ray Kurzweil

Talvez a tentativa mais aventureira de predição das capacidades e técnicas a serem alcançadas com a automatização da fala tenha sido feita por Ray Kurzweil [6], o qual indica as seguintes previsões com o decorrer do tempo.

- Início dos anos 2000: “telefones tradutores permitem que duas pessoas sobre o globo possam conversar mesmo que não falem o mesmo idioma; máquinas de reconhecimento de voz transformam fala em uma exibição visual para o surdo; telefones são atendidos por uma secretária eletrônica inteligente, a qual conversa com a pessoa que originou a chamada determinando a natureza e a prioridade da chamada”;

- 2009: “a maioria de textos é criada através de RAV; transações empresariais rotineiras ocorrem entre um humano e uma máquina virtual (com visual animado imitando uma face humana); mini-máquinas para leitura usadas por deficientes visuais; telefones tradutores em comum para muitas línguas”;
- 2019: “realidade virtual com *displays* tridimensionais, embutidos em óculos e lentes de contato; interações com o computador, feita por gestos e linguagem natural em duplo sentido; as pessoas surdas leriam o que outras pessoas estariam dizendo pelas exibições de suas lente; a maior parte das transações realizada entre humanos e máquinas capazes de simular o homem”;
- 2029: “implantes permanentes ou removíveis permitem troca de informações entre humanos e a rede mundial de computadores; agentes automatizados realizando auto-aprendizado; a maioria das comunicações envolvendo humanos seria entre homens e máquinas”.
- 2049 – 2099: “Não haverá mais distinções perceptíveis entre humanos e computadores”.

“Um PC terá o poder computacional de uma mente humana até
2019, e equivalente a 1000 mentes humana até 2029”.
Ray Kurzweil [6]

Além de serem aplicações estimulantes para a imaginação, fica claro, pelas datas preditas que a tecnologia pode estar atrasada quanto ao pensamento original de Kurzweil e desafios consideráveis precisam ser superados para satisfazer algumas exigências destas aplicações.

2.5.2. Opinião da Comunidade

Em 1997 Roger K. Moore, Prof. da Universidade de Sheffield na Inglaterra, comandou um seminário no Workshop da IEEE e como parte de suas pesquisas na área de “Reconhecimento e Compreensão Automática de voz” (ASRU – *Automatic Speech Recognition and Understanding*) pediu a vários participantes, presentes neste Workshop, que atribuíssem previsões de datas a um conjunto de possíveis

acontecimentos para o progresso da tecnologia RAV. No final da pesquisa, percebeu que era possível construir distribuições sobre as respostas e obter informações úteis como a média, desvio padrão (DP) e valores mínimo e máximo dos anos previstos para cada possível evento [7].

Alguns eventos sugeridos:

1. A maioria dos telefones com resposta interativa a voz aceita a voz como entrada (além de apenas dígitos).
2. Reconhecimento de voz é comumente encontrado em casa (por exemplo, na TV interativa, no controle de eletrodomésticos e sistemas de gerenciamento da casa).
3. Não haverá mais necessidade de pesquisas na área de voz.
4. Telefones tradutores permitem que duas pessoas pelo globo conversem sem a necessidade de falarem o mesmo idioma.
5. Interações com computador feita por gestos e comunicação natural em duplo sentido.

As Figuras 2.3 a 2.7 mostram a variação das respostas obtidas sobre alguns dos eventos citados acima. O eixo das abscissas indica os anos previstos para cada evento e o eixo das ordenadas referencia o número de pessoas que preveram determinada data na pesquisa sobre os possíveis acontecimentos para o progresso da tecnologia RAV.

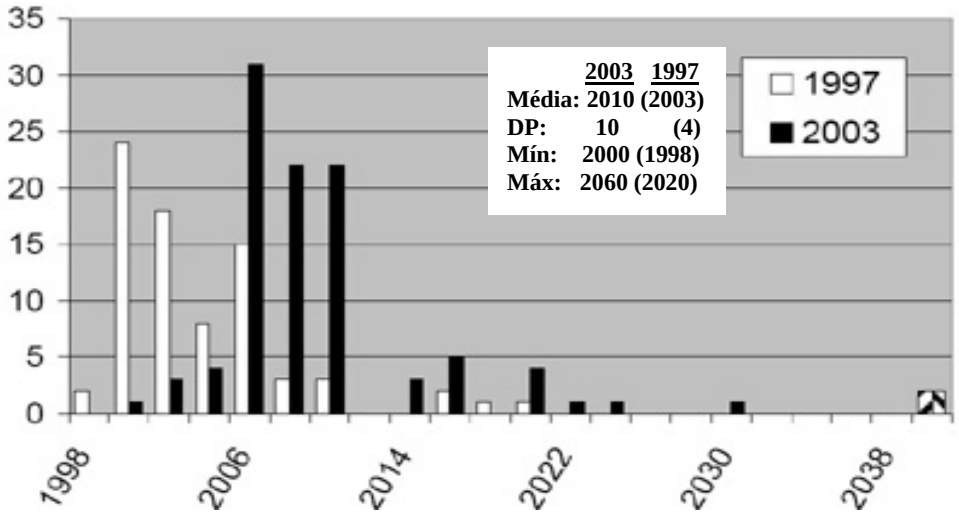


Figura 2.3 - Respostas à declaração "A maioria dos telefones com resposta interativa a voz aceita a voz como entrada (além de apenas dígitos)". Adaptado de Moore [7].

A Figura 2.4 mostra as respostas à declaração “Reconhecimento de Voz é comumente encontrado em casa (por exemplo, na televisão interativa, no controle de eletrodomésticos e sistemas de gerenciamento da casa)”. A média das datas para realização desta aplicação era de 2010 para pesquisa em 1997 e de 2016 para pesquisa em 2003. Percebe-se que, a diferença entre as médias foi de 6 anos, mostrando que a dificuldade em alcançar o evento declarado se manteve constante.

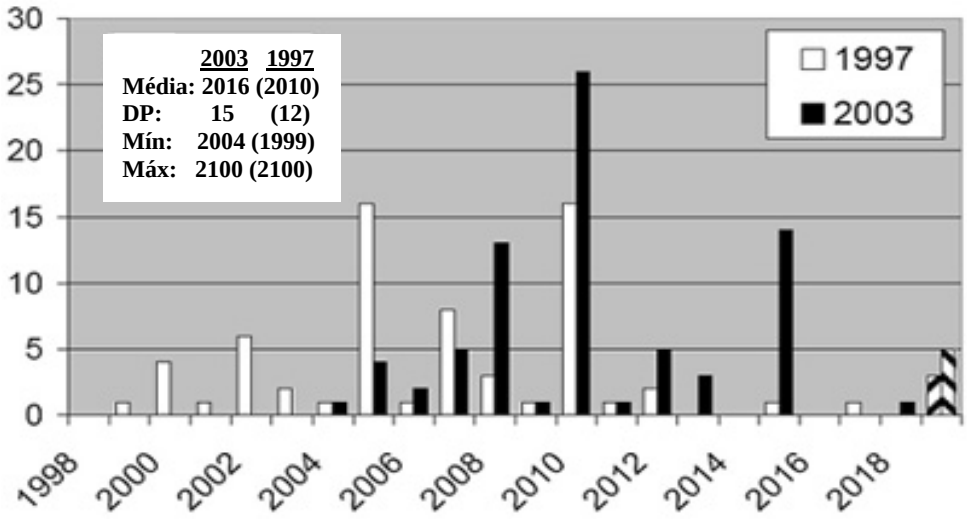


Figura 2.4 – Respostas ao evento “Reconhecimento de voz comumente encontrado em casa (por exemplo, televisão interativa, controle de eletrodomésticos e sistemas de gerenciamento da casa)”. Adaptado de Moore [7].

A Figura 2.5 mostra a resposta à declaração “Telefones tradutores permitem duas pessoas pelo globo conversarem entre si sem falarem o mesmo idioma”. Kurzweil sugeriu que tal acntecimento ocorresse antes do ano 2009 e muitos participantes da pesquisa preveram algo semelhante. Porém, a média das respostas foi do ano de 2057.

A Figura 2.6 mostra as respostas para a declaração “Interações com computador feita por gestos e comunicação natural em duplo sentido”. Embora Kurzweil tenha sugerido o ano de 2019 para tal acontecimento, a maioria dos entrevistados respondeu que isto nunca ocorreria.

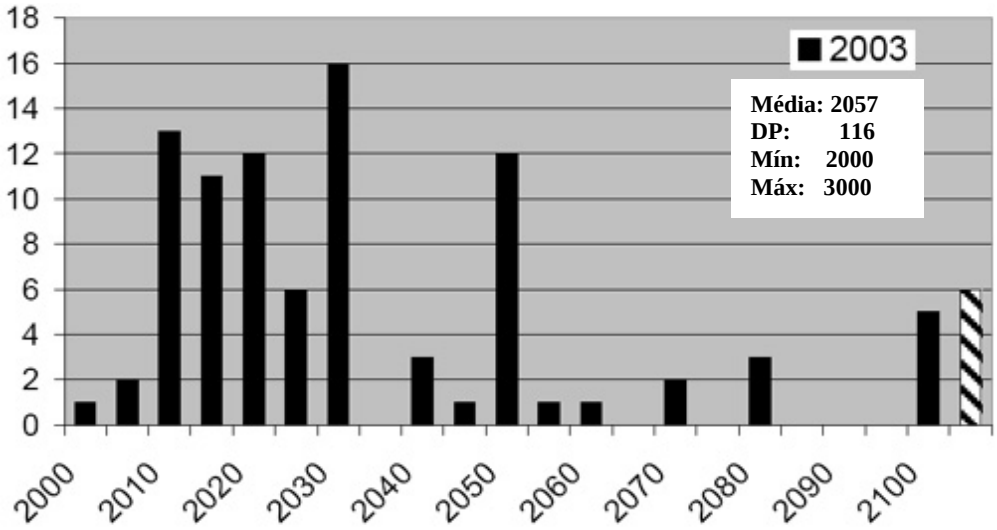


Figura 2.5 - Respostas à declaração "Telefones tradutores permitem duas pessoas pelo globo conversem sem a necessidade de falarem o mesmo idioma". Adaptado de Moore [7].

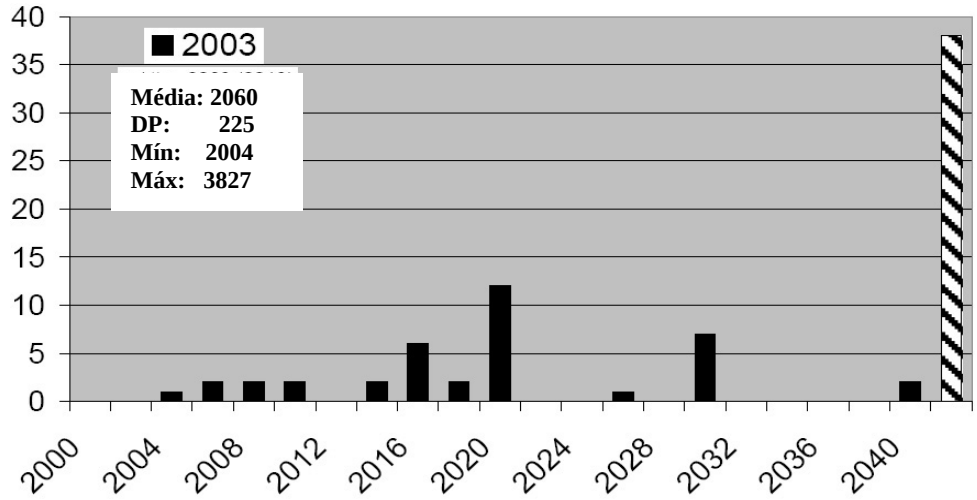


Figura 2.6 - Responstas à afirmação "Interações com computador feita por gestos e comunicação natural em duplo sentido". Adaptado de Moore [7].

Finalmente, a declaração que mais supreendeu os entrevistados foi: "Não haverá mais necessidade de pesquisas em reconhecimento de voz". A Figura 2.7 mostra que uma grande parte das respostas previu um futuro bem distante para a possível realização da declaração, além do grande índice de respostas 'nunca'. Vale a pena citar respostas interessantes como a da pesquisa de 1997, onde um participante atribuiu o ano de 1984 para tal acontecimento. Esta resposta, segundo o autor da pesquisa, indicava que algumas pessoas julgavam existir pesquisas suficientes na área de voz.

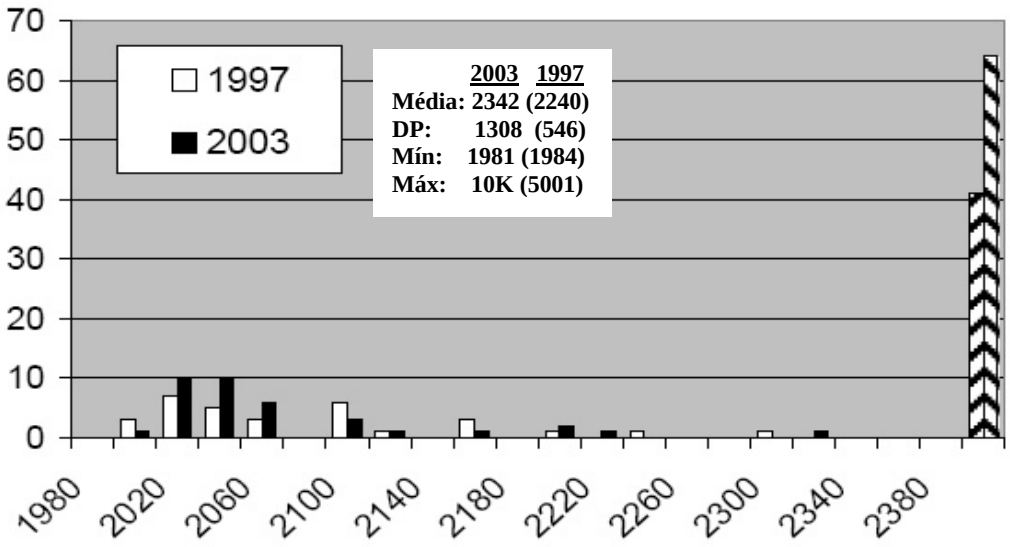


Figura 2.7 - Respostas ao evento "Não haverá mais necessidade de pesquisas em reconhecimento de voz". Adaptado de Moore [7].

Segundo o autor das pesquisas os resultados foram, de modo geral, consistentes. A pesquisa feita em 2003 não foi nem mais otimista e nem mais pessimista se comparada à de 1997. Vários participantes também aproveitaram a oportunidade para sugerir possíveis eventos a serem declarados em pesquisas futuras como, por exemplo, "Computadores realizariam auto-pesquisas sobre reconhecimento feito por homens, modelando a melhoria humana".

De acordo com o atual modelo *data-driven* da tecnologia de voz, Moore em [8] estima que 100.000 a 1.000.000 de horas de voz seriam necessários para treinar sistemas RAV e, deste modo, reduzir por um fator de 5 a taxa de erro por palavra, Figura 2.8.

De forma similar, para sistemas de síntese de voz, o autor em [9] afirma que o tamanho de uma base de dados necessária para captar 100 diferentes estilos de voz e 10.000 diferentes vozes seria de 5.000 Gbytes, mas infelizmente a tecnologia ainda não é capaz de gerar tal banco de dados automaticamente. Este fato traz a tona um dos principais desafios de pesquisadores, que é escapar da necessidade de ter uma grande quantidade de dados para treino visando confiabilidade nos resultados.

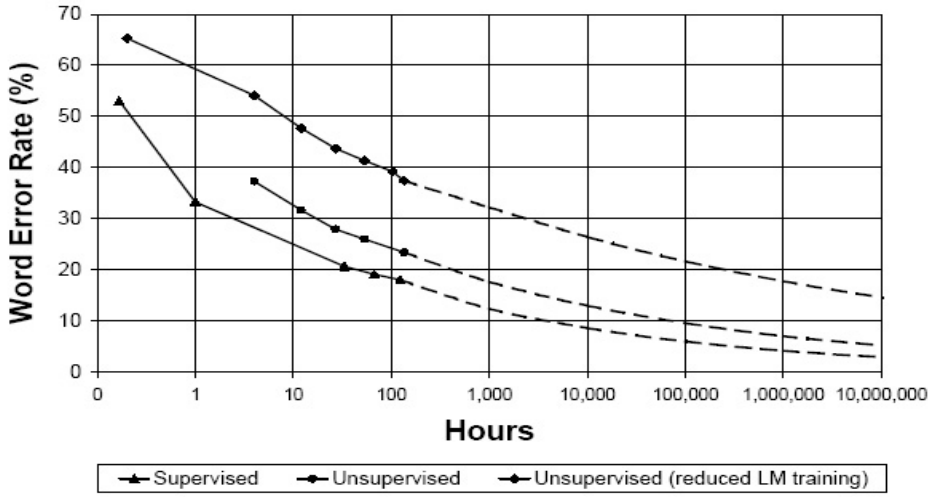


Figura 2.8 - Taxa de erro por palavra com o aumento da quantidade de dados de treino. (Moore [8]).

Os progressos obtidos na área de voz são aparentes e a cada ano mais de 1000 papers são publicados na área [46]. Mas continua a dúvida quanto a possibilidade de alcançar níveis de desempenho elevados, a ponto de satisfazer algumas exigências de mercado.

No próximo capítulo daremos continuidade ao estudo de sistemas RAV, tendo a metodologia usada para o desenvolvimento do sistema de conversão grafema-fone utilizando técnicas de aprendizado de máquina e a criação do dicionário fonético de grande vocabulário para o PB, necessário para desenvolvimento do sistema LVCSR para o PB, descrito no capítulo 4.

3. CONVERSÃO GRAFEMA-FONE

3.1. Introdução

Um importante pré-requisito para serviços que envolvem reconhecimento e/ou síntese de voz é a correspondência entre a ortografia e a(s) pronúncia(s) das palavras. Por exemplo, como mencionado no capítulo anterior, para desenvolver um sistema LVCSR é necessário que se tenha um dicionário grande de pronúncias (ou fonético) que mapeie cada palavra do léxico com uma ou mais transcrições (pronúncia).

A conversão de uma seqüência de caracteres em seqüências de fones não é uma tarefa trivial e diversas técnicas de conversão vêm sendo adotadas ao longo da última década. Neste capítulo apresentaremos um breve histórico sobre conversão grafema-fone; conceitos importantes para a fonetização e a descrição do processo de criação do sistema de conversão grafema-fone criado, utilizando técnicas de aprendizado de máquina como forma de extrair regras de transcrições, a partir de um léxico.

3.2. Breve Histórico

O interesse por técnicas de conversão grafema-fone vem de séculos, fato comprovado pelas descrições não sistemáticas feitas em estudos gramaticais da língua. *Papers* sobre o assunto são raramente encontrados em jornais lingüísticos. Apesar disso, estudos importantes vêm sendo feito, nos últimos anos, sobre conversão grafema-fone.

Os sistemas grafema-fone podem ser organizados em métodos baseados em regras e *data-driven*. Em [14], a técnica de conversão baseada em regras usou a comparação de dois métodos de aprendizado de máquina, um baseado em redes neurais de múltiplas camadas e outro baseado na busca em uma tabela (*look-up table*). Mais recentemente, a combinação dos dois métodos (regras e *data-driven*) foi implementada

pela compilação de regras usando transdutores de estados finitos [15]. Regras manuais de conversão foram implementadas usando cálculo de estado finito, e regras automáticas foram criadas por indução TBL (*Transformation-Based Learning*), discutida também em [16].

Diversos estudos têm enfatizado o uso de algoritmos de aprendizagem associados à pronunciação por analogia [17] [18], problema de redes neurais ou conexionistas [19] [20] [21], alinhamento automático por método de indução [22], método computacional [23] [24], modelos ocultos de Markov [25], dentre outros casos.

Especificamente para o Português do Brasil, destaca-se o algoritmo de transcrição grafema-fone proposto por [28] e, sobretudo, o sistema Ortofon, baseado na Fonologia Articulatória e desenvolvido pelos investigadores do LAFAPE [29]. O Ortofon faz parte de um sistema TTS para o PB chamado Aiurietê [30]. Já para o Português de Portugal, um trabalho interessante vem sendo realizado pela equipe do Processamento de Fala do INESC e o grupo de Fonética e Fonologia do CLUL [26], os quais desenvolveram um módulo de conversão com base num sistema de regras multilinear [27], aproveitando o fato da ortografia ter um alto grau de foneticidade, i. e., uma grande similaridade entre a grafia e a fonia da língua.

A comunidade internacional busca beneficiar programas que promovem cooperações e até competições livres entre grupos de pesquisadores como, por exemplo, a recente criação do desafio internacional envolvendo conversão grafema-fone, *Pronalsyl Letter-to-Phoneme Conversion Challenge* [31].

3.3. Conceitos importantes

3.3.1. Fonética e Fonologia

A fonética é o ramo da gramática que estuda as particularidades dos sons da fala humana, tendo como unidade básica de estudo o fonema, menor unidade sonora de uma língua que entra na constituição de palavras. Enquanto que, a fonologia, é o ramo da

gramática que estuda o comportamento dos fonemas de uma língua, tomando-os como unidades sonoras capazes de criar diferença de significados. A fonologia simplificada do português é composta por cerca de trinta e três a trinta e oito fonemas, variando de acordo com a região onde são pronunciados [32].

Raramente uma palavra é pronunciada da mesma maneira em todo o país. Há muitas variações, influenciadas principalmente pela região em que o idioma é falado e também por fatores como a escolaridade, o meio social e a faixa etária do falante. Para registrar a notação fonética, como o Brasil ainda não admite um padrão oficial para o português falado, como ocorre, por exemplo, com a França.

3.3.2 Alfabeto Fonético

Para simbolizar via escrita a pronúncia real de um som, utiliza-se um alfabeto especial conhecido como alfabeto fonético. A finalidade da transcrição fonética e do alfabeto fonético é justamente a transcrição e a leitura de um som em qualquer idioma por uma pessoa treinada. Assim, esse alfabeto deve apresentar convenções inequívocas e de maneira explícita. A forma mais comum de representar os fonemas é através do alfabeto fonético internacional (*International Phonetic Alphabet* - IPA), vide Apêndice 1. Todavia, os símbolos adotados nas transcrições fonéticas deste trabalho fazem parte do alfabeto fonético SAMPA, Speech Assessment Methods Phonetic Alphabet, tendo alguns deles sido adaptados a casos particulares da pronúncia do português do Brasil.

O SAMPA consiste em um sistema de escrita fonético legível por computadores que usam o conjunto de caracteres ASCII de 7 bits ou ASCII extendido de 8 bits. Tem como base o Alfabeto fonético internacional (IPA).

O alfabeto fonético adotado neste trabalho foi uma modificacação do SAMPA, conforme tabela do Apêndice 4 (disponibilizada pela aluna de Pós-Graduação do Curso de Letras" da UFPa, Helane Fernandes).

3.3.3. Representação de Caracteres e Símbolos

Nos primórdios computacionais foi criada uma tabela de caracteres básica que continha 128 caracteres e, portanto era possível de ser armazenada em um binário de 7 bits chamado ASCII.

Com a disseminação dos computadores pelo mundo veio a necessidade de incorporar a esta tabela caracteres diferentes (variantes em cada país) composto não apenas por letras, números e sinais, mas por caracteres especiais. Vários países usam letras acentuadas, como é o caso do Brasil e, deste modo, a tabela foi expandida para binário de 8 bits (256 caracteres).

Cada computador (ou cada fabricante) adota um determinado padrão. O número de bits que será utilizado no padrão é uma decisão do fabricante e determinará quantos caracteres (e símbolos) poderão ser representados. O servidor Linux utilizado para realizar as simulações implementadas neste trabalho não reconhecia alguns caracteres especais, como as letras acentuadas do PB e, deste modo, foi necessário a criação de um programa que convertia os caracteres do formato DOS para o formato Unix e vice-versa.

3.4 Considerações Sobre a Língua Portuguesa

O português se situa entre as línguas de ortografia “homogênea” quanto a sua pronunciação. Mesmo assim, não podemos dizer que a transcrição de um texto em português seja uma tarefa simples. O sistema ortográfico e o sistema fonético não são equivalentes, pois nem sempre é verdadeira a afirmação de que cada grafema da seqüência ortográfica corresponda a um fonema. Há casos em que um mesmo fonema pode ser representado por grafemas diferentes (o fonema /s/, por exemplo, pode ser representado pelos grafemas “s”, “c” e “x”, como nas palavras “sela”, “cedo” e “próximo”, respectivamente). O inverso também acontece: fonemas distintos podem ser representados pelo mesmo grafema (considere o exemplo do grafema “g”, que pode

representar o fonema /Z/, como na palavra “gente”, ou o fonema /g/, como na palavra “gato”).

Há ainda casos em que um fonema simples pode ser representado por uma seqüência não unitária de grafemas, como em “carro” (/r/), “alho” (/´/) ou “ficha” (/S/). Um único grafema também pode representar uma seqüência de fonemas, como a letra “x” da palavra “tóxico” (/k/+/s/). Por fim há situações em que um dado grafema da seqüência ortográfica não é mapeado para nenhuma unidade fonológica como, por exemplo, a letra “h” na palavra “homem”.

No caso da língua portuguesa, em que existe uma regularidade grande entre a representação ortográfica e a transcrição fonética, pode-se determinar a pronúncia correta da maioria das palavras através da aplicação pura e simples de regras de transcrição. A aplicação destas regras parte do princípio de que uma seqüência de grafemas pertencente a uma dada palavra pode ser convertida em uma seqüência fonética a partir da análise do contexto dos grafemas que lhe são adjacentes tipo:

<contexto_esquerdo> contexto_de_análise <contexto_direito> → transcrição

O próximo item descreve o sistema grafema-fone desde a preparação dos dados, treino e teste dos dados até análise dos resultados.

3.5. Sistema de Aprendizado de Pronúncias

O sistema, feito na linguagem de programação Java, produz transcrições fonéticas automáticas das palavras de entrada e foi desenvolvido com o intuito de suprir a carência de um dicionário fonético grande o suficiente para ser utilizado LVCSR para o PB, o *software* aprende a pronúncia de textos em português e outros idiomas como, Inglês e Francês e seu princípio de funcionamento foi baseado em [19]. A aprendizagem e a classificação dos fonemas foram feitas utilizando-se o programa Weka. Mais especificamente, usou-se o algoritmo de árvore de decisão J.48 e *Naïve Bayes* [34].

Antes de treinar e testar os dados, nosso experimento requer que estes dados estejam previamente alinhados, isto é cada letra de uma palavra tem que estar alinhada com seu fonema correspondente. Para fazer este alinhamento, um *script* em Java foi criado tomando como base o método usado em [35]. Outros *scripts* Java foram criados para processar os módulos de conversão do nosso sistema e fazer a análise do erro por palavra resultante das classificações. Parte dos passos seguidos para processamento do módulo de conversão grafema-fone podem ser vistos no Apêndice 3.

3.5.1 Weka

O pacote Weka foi desenvolvido na Universidade de Waikato na Nova Zelândia. Encontra-se disponível on-line uma abrangente documentação do código fonte [33] e o grupo tem lançado periodicamente correções e *releases* do *software*, além de manter uma lista de discussões acerca da ferramenta. Grande parte de seus componentes de *software* são resultantes de teses e dissertações de grupos de pesquisa desta universidade. Alguns recursos e características do *software* estão descritos a seguir:

Recursos para Mineração de Dados

a) Aprendizado Supervisionado

Esta categoria de algoritmos possui esta denominação porque o aprendizado do modelo é supervisionado, ou seja, é fornecida uma classe à qual cada amostra no treinamento pertence. Estes algoritmos são preditivos, pois suas tarefas de mineração desempenham inferências nos dados com o intuito de fornecer previsões ou tendências, obtendo informações não disponíveis a partir dos dados disponíveis:

- Classificação: através destes algoritmos supervisionados (com ênfase na precisão da regra) é possível determinar o valor de um atributo através dos valores de um subconjunto dos demais atributos da base de dados. As formas mais comuns de representação de conhecimento dos algoritmos de classificação são regras e árvores. Os algoritmos Id3, C45, J48, *ADTree*, *UserClassifier*, *PredictionNode*, *Splitter*, *ClassifierTree*, *M5Prime*, por exemplo, geram como

resultado árvores de classificação, enquanto que outros como *Prism*, *Part*, *OneR* geram regras de classificação. Outra opção seria a representação através de tabela de decisão implementada, por exemplo, pelo algoritmo *DecisionTable*. Modelos matemáticos, de regressão e redes neurais também representam resultados de algoritmos como *SMO*, *LinearRegression*, Neural, dentre outros.

- Seleção de atributos: em bases de dados encontram-se atributos que têm um peso maior ou até determinante nas tarefas de mineração de dados. Com algoritmos de seleção de atributos é possível determinar os atributos de fato relevantes para a mineração dos dados, separando-os dos atributos irrelevantes. O Weka disponibiliza vários algoritmos para esta categoria de mineração, dentre eles *InformationGain*, *PrincipalComponents* e *ConsistencyEval*.

b) Aprendizado Não-Supervisionado

Nestes algoritmos o rótulo da classe de cada amostra do treinamento não é conhecido e o número ou conjunto de classes a ser treinado pode não ser conhecido a priori, daí o fato de ser um aprendizado não-supervisionado. Além disso, são também descritivos, pois descrevem de forma concisa os dados disponíveis, fornecendo características das propriedades gerais dos dados minerados:

- Associação: quando a classe de uma tarefa de mineração não é determinada como no caso da classificação, uma boa opção é o algoritmo de associação *Apriori* do Weka. O algoritmo elege os atributos determinantes (lado esquerdo da regra) e os atributos resultantes (lado direito) na tarefa revelando associações entre valores dos atributos, tendo o algoritmo sua ênfase no compromisso entre precisão e cobertura.
- *Clustering*: em algumas situações, torna-se necessário verificar como as instâncias de uma determinada base de dados se agrupam devido a características intrínsecas de seus atributos, sem que seja definida uma classe para a tarefa. A partir da definição de uma métrica de similaridade para cada atributo e uma função de combinação destas métricas em uma métrica global, os objetos são agrupados com base no princípio da maximização da similaridade intraclasse e da minimização da similaridade interclasse. Weka possui os

algoritmos *Cobweb*, *Simple Kmeans* e Em para tarefas que demandam a descoberta de padrões de agrupamento nos dados. Como exemplo, podemos utilizar algoritmos de *clustering* para identificar subgrupos homogêneos de clientes de uma determinada loja.

Recursos para Validação de Resultados

O sistema possui recursos e funcionalidades para avaliar e comparar resultados e modelos, dentre os quais: teste e validação, que fornecem parâmetros de validade e confiabilidade nos modelos gerados (*cross validation*, *suplied test set*, *use training set*, *percentage split*); indicadores estatísticos para auxiliar a análise dos resultados (matriz de confusão, índice de correção e incorreção de instâncias mineradas, estatística *kappa*, erro médio absoluto, erro relativo médio, precisão, *F-measure*, dentre outros).

Formato do arquivo

O Weka possui um formato de arquivo próprio, identificado pela extensão *arff*. Antes de aplicar os dados a qualquer algoritmo do pacote Weka, estes devem ser convertidos para o formato *arff*, o qual consiste basicamente de duas partes. A primeira parte chamada *header* contém uma lista de todos os atributos, onde se deve definir o tipo do atributo ou os valores que ele pode representar, ao utilizar os valores estes devem estar entre "{ }" separados por vírgulas. A segunda parte consiste das instâncias, i.e., os registros a serem minerados com o valor dos atributos para cada instância separado por vírgula, a ausência de um item em um registro deve ser atribuída pelo símbolo " ? ".

Modo de Operação

O Weka apresenta três modos de operação. O primeiro, *Simple CLI*, executa os algoritmos do Weka através de linha de comando. *Explorer* executa o modulo gráfico para execução dos algoritmos. E o terceiro, *Experimenter*, executa o módulo para manipular bases de dados.

3.5.2. Dicionário UFPAdic1.0

Uma contribuição deste trabalho é a criação do dicionário de pronúncias para PB versão 1.0 com 11.827 palavras transcritas manualmente, as quais foram obtidas de diversas fontes, sendo a maior parte delas proveniente de versões comercias de dicionários eletrônicos.

- UFPAdic1.0, 11.827 palavras, Português do Brasil.

Outros dicionários de pronúncias, usados para testes e validações do sistema, encontram-se disponíveis publicamente e podem ser encontrados on-line na página do *Pronalsyl* (http://www.pascal-network.org/challenges/pronalsyl/) [31]

- *NetTalk*, 20.008 palavras, Inglês Americano[19];
- *Brulex*, 27.473 palavras, Frânces [36];
- *Beep1.0*, 256.980 palavras, Inglês Britânico [37].

Comparando os tamanhos dos dicionários, nota-se que o dicionário UFPAdic1.0 é o menor, mas seu tamanho é comparável com os dicionários usados em estudos recentes (por exemplo, [16]). Importante citar que os resultados de experimentos conduzidos com uso de dicionários transcritos manualmente são providos de *ground truth*. Em contrapartida, a precisão de um conversor grafema-fone usando transcrições geradas automaticamente pode estar sendo influenciada da mesma forma que os algoritmos que originaram tais transcrições.

3.5.3. Módulos do conversor grafema-fone

a) Alinhamento da base

Alinhar duas representações simbólicas com a mesma ‘mensagem’ lingüística é um requisito comum para tecnologia de voz. Alinhamento do léxico é um passo importante e crucial para o esquema de treinamento do conversor. A partir deste alinhamento, constroem-se os dados de onde são extraídas as regras para transcrição. O

alinhamento pode ser feito manualmente, mas à medida que os dicionários tornam-se maiores, o alinhamento manual torna-se inviável, pois consome tempo, tende a erro, e limita o tamanho do vocabulário que pode ser usado para treino. Por outro lado, o alinhamento automático é um assunto difícil de implementar para determinadas línguas como o Inglês.

Neste trabalho, o alinhamento baseado em [35] é implementado por meio de Programação Dinâmica (PD) e devido a excelentes performances, este método vem sendo usado como base para muitos outros trabalhos.

Assumindo que temos um conhecimento prévio da probabilidade de mapeamento de uma letra com um fonema, a PD torna-se uma ferramenta simples e poderosa capaz de alinhar textos e fonemas. O conhecimento sobre o mapeamento letra-fonema é compilado em uma matriz de 'associação' A, de dimensão L x P, onde L é o número de letras (no caso do PB igual a 38) e P é o número de fonemas (no caso do PB igual a 35).

Vários métodos podem ser usados para inicialização da matriz A e a forma mais simples é realizada de forma 'ingênua', método heurístico, onde cada palavra do dicionário é varrida e cada vez que a letra l corresponder ao fonema p na mesma palavra o elemento correspondente a_{lp}^{0} da matriz A^0 é incrementado, sem considerar a posição. Depois do 1º passo, varrendo todo dicionário, cada elemento a_{lp}^{0} contém o número de vezes que a letra l e o fonema p aparecem na mesma palavra. Isso não quer dizer que a letra l e o fonema p estejam sempre alinhados, pois esta consideração só seria possível caso a comparação fosse feita com relação a mesma palavra.

A partir deste alinhamento imperfeito, podemos realizar a segunda varredura sobre o dicionário e produzir uma matriz de associação A^1 'melhorada' com elementos a_{lp}^{1} que contam o número de vezes que a letra l e o fonema p aparecem (alinhadas) na mesma posição, i. Nesta 1ª iteração 'nulos' são inseridos como conseqüência da PD, assim letras podem ser associadas a fonemas 'nulos' e fonemas podem ser associados com letras 'nulas'. Um novo conjunto de candidatas alinhadas são criadas e pontuadas. Novamete um 'bom' alinhamento é selecionado e A^1 é atualizada para A^2. Outras iterações podem ser usadas para melhorar os alinhamentos e estimar até estimação da

matriz de associação final, determinada pela convergência ou alcance do limite máximo de iterações.

O princípio da PD afirma que a solução global para o problema de encontrar um caminho ótimo pode ser encontrada pela seqüência de máximos locais; em outras palavras nenhum local não máximo pode contribuir para a solução de um máximo-global. (Este princípio é largamente usado em linguagem computacional e tecnologia de fala, e forma a base para o algoritmo *Viterbi* [39] [40]), usado de várias maneiras em reconhecimento de voz, síntese de voz, e processamento de texto. Segue um exemplo de como é realizado o alinhamento de palavras, considerando a palavra *phase*, referente ao dicionário de pronúncias *NetTalk*.

Grafema	p	h	a	s	e
	|	|	|	|	|
Fonema	-	f	eI	z	-

O processo de alinhamento grafia e fonia, para uma palavra específica, pode ser visto como um problema de busca do caminho ótimo em uma tabela, ou matriz B, indexada por letras da palavra e fonemas de sua pronunciação, como mostra a Figura 3.1. As entradas da matriz devem ser interpretadas como os graus de 'associação' entre cada letra e cada fonema.

	$	f	eI	z	$
#	0	0	0	0	0
p	0	9	0	0	0
h	0	2580	27	35	0
a	0	42	23098	937	0
s	0	79	3	45788	0
e	0	947	1732	2641	0
#	0	0	0	0	0

Figura 3.1 - Matriz de 'associação' B, indexada por letras da palavra soletrada "*phase*" e fonemas de sua pronunciação "*feIz*".

Os valores mostrados são relativos a 1ª iteração do algoritmo de PD, usado neste trabalho. Inicialmente são incluídos delimitadores de palavras (#) e fonemas ($) com a associação 0. Isto permite com que o algoritmo PD alinhe a primeira letra ou fonema de uma palavra com o símbolo 'nulo'. De outra forma a primeira letra seria sempre alinhada com o primeiro fonema.

O 'melhor' alinhamento é definido pelo caminho percorrido desde o topo à esquerda da matriz B até a base direita, tal que maximize o acúmulo de valores associados ao longo deste percurso. Para encontrar o melhor caminho são criadas 2 novas matrizes, matriz C e matriz D, Figura 3.2.

	$	f	eɪ	z	$
#	0, ϵ	0, →	0, →	0, →	0, →
p	0, ↓	9, ↘	9, →	9, →	9, →
h	0, ↓	2580, ↘	2580, →	2580, →	2580, →
a	0, ↓	2580, ↓	25678, ↘	25678, →	25678, →
s	0, ↓	2580, ↓	25678, ↓	71446, ↘	71446, →
e	0, ↓	2580, ↓	25678, ↓	71446, ↓	71446, ↘
#	0, ↓	2580, ↓	25678, ↓	71446, ↓	71446, ↘

Figura 3.2 - Superposição das matrizes C e D. Contém valores acumulados pelas associações através da PD, juntamente com os ponteiros indicando o movimento percorido para chegar à célula atual, tal que maximize o valor da associação.

A matriz C contém os valores acumulados pelas associações, e cada entrada representa o valor máximo acumulado pela associação até o exato ponto na tabela (i.e., até ponto atual de alinhamento). Já a matriz D contém os ponteiros, indicando a célula precursora, i.é., de onde o algoritmo PD moveu-se para célula atual. As equações de maximização usadas são:

$$Ci,j = máx\left\{\begin{matrix} C_{i-1,j-1} + B_{i,j} \\ C_{i-1,j} - \delta \\ C_{i,j-1} - \delta \end{matrix}\right\} \begin{matrix} 1 \le i \le |l_W| \\ 1 \le j \le |p_W| \end{matrix}$$

Onde $|l_W|$ e $|p_W|$ são os comprimentos das letras e fonemas da palavra, respectivamente (incluindo os delimitadores). E δ é um valor de penalidade, no caso setado em 0.

Se a maximização tiver escolhido o argumento $C_{i-l,j-1} + B_{i,j}$, isto corresponde a um movimento diagonal na matrizes B e C, e a entrada na matriz D é uma seta diagonal, indicando que o grafema alinha-se com a fonema. Caso a maximização tenha escolhido o argumento $C_{i-l,j} - \delta$, isto corresponde a um movimento vertical nas matrizes B e C, e a entrada na matriz D é uma seta pra baixo, indicando que o grafema alinha-se com o símbolo nulo. Em último caso, se a maximização optar por escolher o argumento $C_{i,j-1} - \delta$, isto corresponde um movimento à direita nas matrizes B e C e a entrada na matriz D será uma seta à direita, indicando que o fonema é alinhado com o símbolo nulo.

Vale lembrar que o algoritmo de PD, citado neste trabalho, foi inicialmente proposto por Needleman e Wunsch [38] e escolhido para o alinhamento entre grafia e fonia por ser simples e diversos testes já terem comprovado seu bom desempenho.

b) Criação dos Arquivos de Treino e Teste

Depois de alinhado o dicionário é dividido em dois arquivos distintos, um arquivo usado para treino e outro para teste. A Tabela 3.1 mostra exatamente o número de fonemas e palavras usados para treino e teste de cada dicionário.

		Fonemas	**Palavras**
NetTalk	Treino	88112	12000
	Teste	58831	8008
Brulex	Treino	141332	16500
	Teste	93921	10973
Beep 1.0	Treino	1413479	154200
	Teste	941369	102780
UFPAdic1.0	Treino	64799	8300
	Teste	27497	3527

Tabela 3.1 - Número de fonemas e palavras usadas para treino e teste dos dicionários.

c) Conversão dos arquivos de treino e teste para formato *arff* (Weka)

Antes de aplicarmos os dados a qualquer algoritmo do pacote Weka procedemos a conversão destes para o formato *arff*, utilizando variação de contexto, de onde são extraídas as regras de transcrição pelo algoritmo de aprendizagem. Neste trabalho, o contexto é variado simetricamente e recebe valores 1, 3 e 5. Um exemplo de variação do contexto, considerando a palavra 'casa', Figura 3.3:

grafemas	c	a	s	a
fonemas	k	a	z	a

Contexto = 1

Esquerda 1	Análise	Direita1	Fone
#	c	a	k
c	a	s	a
a	s	a	z
s	a	#	a

Contexto = 2

Esquerda 2	Esquerda 1	Análise	Direita1	Direita2	Fone
#	#	c	a	s	k
#	c	a	s	a	a
c	a	s	a	#	z
a	s	a	#	#	a

Figura 3.3 – Variação de contexto, arquivo *arff*.

Segue um exemplo das primeiras linhas do arquivo *arff* criado para o dicionário UFPAdic 1.0, usando contexto simétrico igual 1, onde a primeira parte do arquivo , chamada de *header*, contém uma lista de todos os atributos separados por '{}' e, neste trabalho, referenciando os grafemas e fonemas presentes no dicionário em trabalho. Já a segunda parte contém as instâncias, ou seja, os registros a serem minerados com o valor dos atributos para cada instância (grafema) separado por vírgula. O símbolo (#) é usado para preenchimento do contexto, no momento da expansão. O símbolo (-) indica a ocorrência de um grafema ou fonema nulo. O programa em Java, criado para a

conversão dos dados para o formato *arff*, permite que o valor do contexto seja flexível e escolhido pelo usuário.

Arquivo *UFPAdic1.arff*:

```
@relation GraphemeToPhoneme
@attribute left1 { ü,ú,ô,ó,í,ê,é,ç,ã,â,á,à,z,y,x,w,v,u,t,s,r,q,p,o,n,m,l,k,j,i,h,g,f,e,d,c,b,a,-,#}
@attribute central { ü,ú,ô,ó,í,ê,é,ç,ã,â,á,à,z,y,x,w,v,u,t,s,r,q,p,o,n,m,l,k,j,i,h,g,f,e,d,c,b,a,-,#}
@attribute right1 { ü,ú,ô,ó,í,ê,é,ç,ã,â,á,à,z,y,x,w,v,u,t,s,r,q,p,o,n,m,l,k,j,i,h,g,f,e,d,c,b,a,-,#}
@attribute phone { a^,i~,Z,o~,u~,U,S,R,O,a~,L,z,J,I,v,u,E,t,s,r,p,o,n,m,e~,l,k,i,h,g,f,e,d,b,a, -}

@data
#,a,d,a
a,d,j,d
d,j,e,Z
j,e,t,e
e,t,i,t
t,i,v,i
i,v,o,v
v,o,#,u
#,r,a,R
r,a,n,a~
a,n,c,-
n,c,o,k
c,o,r,o
o,r,#,h
```

grafemas	a	d	j	e	t	i	v	o
fonemas	a	d	Z	e	t	i	v	u

grafemas	r	a	n	c	o	r
fonemas	R	a~	-	k	o	h

d) Treinando e Testando os classificadores

Após o processo de preparação dos dados, estes são submetidos a algoritmos de classificação, em um processo de aprendizado supervisionado que tem por base informações a respeito dos grafemas e fonemas contidos no dicionário em questão. Os seguintes algoritmos, disponíveis no *software* Weka [33], foram utilizados no processo de classificação:

- *Naive Bayes*;
- Indutores de regras (J.48);

Alguns algoritmos possuem parâmetros a serem configurados antes do processo de treinamento. O desempenho dos classificadores gerados após o treinamento varia de acordo com os valores de parâmetros escolhidos. É necessário, portanto, durante o processo de aprendizado, buscar valores de parâmetros que maximizem este

desempenho. Esse processo, se realizado manualmente, apresenta inconvenientes, por necessitar da intervenção humana em todas as suas etapas.

A primeira etapa é escolher as combinações de parâmetros que se deseja avaliar. Depois de treinados classificadores utilizando cada combinação, é necessária uma avaliação do desempenho de cada classificador gerado para a escolha do melhor. Desta forma, o processo se torna exaustivo e demorado. Devem ser testadas e avaliadas muitas combinações para tornar a busca mais eficaz. No entanto, por conta da necessidade de intervenção freqüente do ser humano no processo de busca, existe um grande risco de não serem avaliadas as melhores combinações de parâmetros.

Em virtude disso, esse trabalho propõe uma forma de automatizar este processo utilizando a classe *cvparameterselection* disponível no Weka. Esta classe efetua validação cruzada sobre o limite de parâmetros estabelecidos buscando a combinação de parâmetros que leva a um melhor desempenho do classificador. Esta função pode ser executada atravé da seguinte linha de comando:

» java weka.classifiers.meta.CVParameterSelection -W weka.classifiers.trees.J48 -t train_dict.arff -T test_dict.arff -d classifier.model -P "M 2 6 3" -- -L -S -A

Este comando ativa a maquina virtual Java e a instrui a executar o algoritmo J48 utilizando a opção CVParameterSelection para seleção de parâmetros. A opção –t informa ao algoritmo que o próximo argumento é o nome do arquivo de treinamento. E por estarmos utilizando arquivos distintos para treino e teste (Supplied test set), a opção –T informa ao algoritmo que o próximo argumento é o nome do arquivo de teste. A opção –d armazenar o classificador gerado após treino e teste dos dados. A opção –P contém o limite de variação de valores a serem processados pela classe cvparameterselection. Já os parâmetros, neste exemplo –L – S –A recebem valores padrão do classificador, mas também podem receber valores estabelecidos pelo usuário.

Segue um exemplo de seleção de parâmetros via *CVParametersSelection* para

– P "M 2 6 3", onde:
M → é o menor número de instâncias por folha (árvore de decisão J.48)

a = 2 → valor mín.

b = 6 → valor máx.

n = 3 → passo

Variação = (valor máx. - valor mín) / n-1 => (6 - 2) / 2 = 2

De acordo com a faixa de valores escolhidas para a classe, a variação do número de folhas por instância assumidas para treino do classificador será de:

$M_1 = 2$

M_2 = M anterior + variação = 2 + 2 = 4

M_3 = M anterior + variação = 4 + 2 = 6

e) Regras de transcrição

As regras de transcrição são extraídas de forma automática usando técnicas de aprendizado de máquina. Em particular, técnicas de aprendizagem por indução são capazes de buscar características em comum num dado e generalizá-lo. Na fase de treino, varrem-se todas as instâncias do arquivo de treino (<Esquerda1> Análise <Direita1> → TranscriçãoOriginal) e a partir destas informações os algoritmos de classificação criam regras para transcrição que serão utilizadas na fase de teste, Figura 3.4.

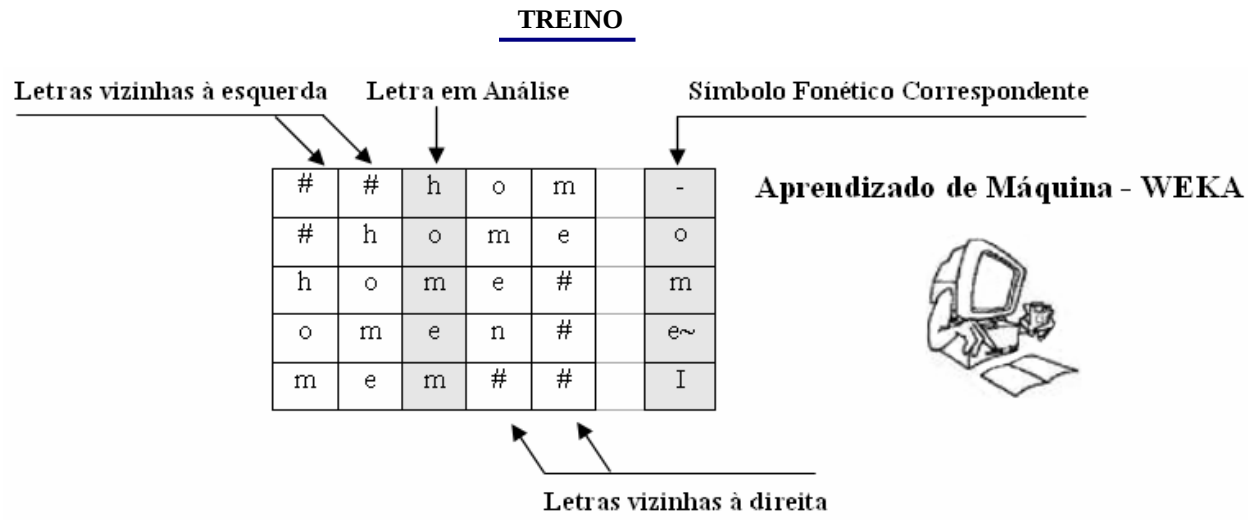


Figura 3.4 - Esquema de treino do conversor grafema-fone.

Já a Figura 3.5 mostra o processo da extração das regras de transcrição na fase de teste de um classificador, considerando o contexto de expansão simétrico e igual a 1. Uma janela móvel de grafemas do tipo (<Esquerda1> Análise <Direita1>) varrerá os grafemas de cada palavra do arquivo de teste e os dados serão entregues para o classificador já treinado e a partir das regras criadas na fase de treino irá gerar a saída do fonema correspondente ao grafema em análise (<Esquerda1> Análise <Direita1> → TranscriçãoSugerida).

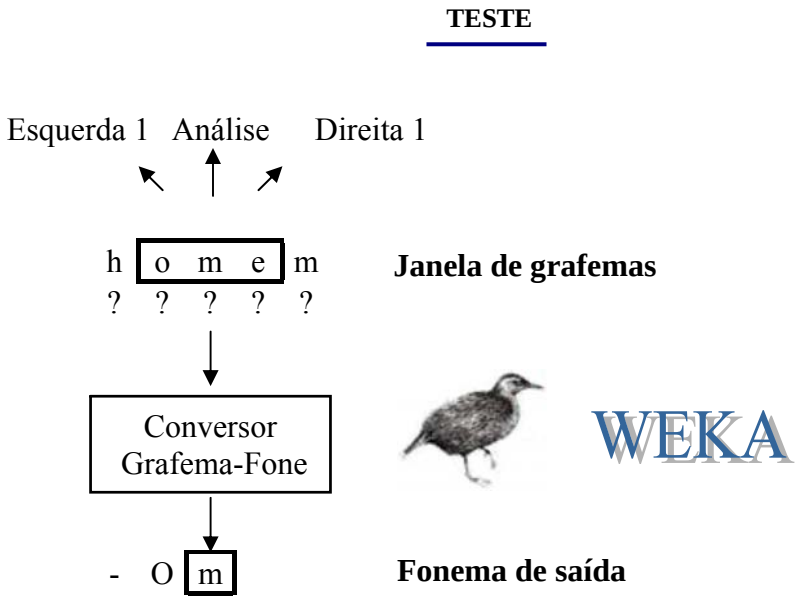


Figura 3.5 - Esquema de transcrição realizada pelo conversor grafema-fone.

3.6. Resultado das simulações

As tabelas 3.2 a 3.6 apresentam os resultados obtidos nos testes realizados com os seguintes dicionários: UFPAdic1.0 (Português Brasil), *NetTalk* (Inglês Americano), *Brulex* (Francês) e *Beep1.0* (Inglês Britânico). O classificador *Naïve Bayes* foi usado, apenas, para indicar o nível de performance que pode ser alcançado com tão simples algoritmo.

UFPAdic1.0	**Taxa de Erro por fonema (%)**		**Taxa de Erro por palavra (%)**	
	Árvore J.48	***Naive Bayes***	**Árvore J.48**	***Naive Bayes***
Contexto 1	2.77	5.53	18.66	33.82
Contexto 3	2.77	8.55	11.40	48.20
Contexto 5	1.60	12.52	10.92	62.63

Tabela 3.2 - Taxa de erros do dicionário UFPAdic1.0 variando o contexto de expansão.

NetTalk	**Taxa de Erro por fonema (%)**		**Taxa de Erro por palavra (%)**	
	Árvore J.48	***Naive Bayes***	**Árvore J.48**	***Naive Bayes***
Contexto 1	16.65	23.41	72.18	83.85
Contexto 3	10.60	22.4	52.45	81.38
Contexto 5	10.94	26.58	53.66	84.62

Tabela 3.3 - Taxa de erros obtidas para o dicionário *NetTalk*.

Brulex	**Taxa de Erro por fonema (%)**		**Taxa de Erro por palavra (%)**	
	Árvore J.48	***Naive Bayes***	**Árvore J.48**	***Naive Bayes***
Contexto 1	7.81	16.09	48.57	77.31
Contexto 3	2.12	14.75	13.19	71.30
Contexto 5	2.06	19.19	12.56	81.30

Tabela 3.4 - Precisão da transcrição obtida pelo *Brulex*.

Beep1.0	**Taxa de Erro por fonema (%)**		**Taxa de Erro por palavra (%)**	
	Árvore J.48	***Naive Bayes***	**Árvore J.48**	***Naive Bayes***
Contexto 1	14.65	23.82	72.92	89.81
Contexto 3	5.66	23.70	35.95	89.70
Contexto 5	xx.xx	26.71	xx.xx	91.99

Tabela 3.5 - Precisão da tanscrição obtida pelo *Beep1.0*.

Os resultados alcançados pelas simulações são compatíveis com os encontrados na literatura para o *Beep1.0*, *Brulex* e *NetTalk*. Os resultados obtidos para o Português do Brasil também foram de bom desempenho se comparado aos das outras línguas, haja vista, tratar-se de uma língua "homogênea" em termos de pronunciação.

Vale notar que os experimentos realizados neste trabalho são relativamente simples, e simulações mais elaboradas podem ser realizadas levando em consideração característica como, a silabificação, tonicidade, etc. Estas análises devem ser incorporadas ao sistema a ser apresentado no desafio proposto pelo *Pronalsyl* [31].

3.7. Construção do Dicionário Fonético

Um dos pontos fundamentais para desenvolvimento de sistemas LVCSR é a existência de um dicionário fonético de grande vocabulário, do qual são extraídas transcrições fonéticas de todas as palavras contidas no *corpus* usado para treino. Para a criação de tal dicionário, utilizou-se como base o dicionário UFPAdic1.0, de onde foram extraídas as regras de transcrições utilizando técnicas de aprendizado de máquina, item 3.4.

No caso específico deste trabalho, o melhor classificador resultante dos testes com o dicionário UFPAdic1.0 foi gerado com a aplicação do algoritmo J.48 (árvore de decisão) utilizando o arquivo *arff* de contexto igual a 5 (UFPAdic5.arff). A partir das regras contidas neste classificador foi possível realizar a transcrição automática de novas palavras (obtidas da Folha de São Paulo e outras fontes) usadas para criação do dicionário fonético de grande vocabulário (com mais de 60.000 palavras). Este dicionário foi usado no desenvolvimento do sistema LVCSR para o PB, descrito no capítulo anterior.

4. LVCSR PARA O PORTUGUÊS BRASILEIRO

4.1. Introdução

Dentro da área de reconhecimento de voz existem classes de problemas distintas a serem resolvidas, como visto no capítulo 2. Devido a essa diversidade de problemas, diferentes estratégias de solução podem ser aplicadas, adequadas a cada tipo de problema [10]. No caso de sistemas LVCSR, não há restrições quanto ao vocabulário e as palavras podem ser combinadas livremente, sem a necessidade de pronúncia pausada.

Neste capítulo será apresentada a metodologia usada para implementação do sistema LVCSR para o Português Brasileiro, usando o *software Hidden Markov Model Toolkit* (HTK) para treinamento e reconhecimento da base (*corpus* Spoltech).

4.2. Blocos Componentes de um sistema de RAV

Um modelo geral para reconhecimento de voz é mostrado na Figura 4.1. O sinal de voz é tratado através de diversos blocos que compõem um sistema RAV. No "estado da arte" em reconhecimento de voz, o processo complexo de decodificação do sinal realizado pela mente humana é feito por modelos estatísticos. Para que uma decisão seja tomada, é necessário um conhecimento a priori, dos modelos da linguagem (ML) e acústico (MA), que são obtidos a partir de *features* extraídas, tanto de vozes digitalizadas quanto de textos da língua em estudo. O modelo de linguagem representa o léxico e a probabilidade de seqüências de palavras do vocabulário. Nesta fase, vários exemplos de frases são selecionados para construir a base de dados necessária para a criação do modelo de linguagem.

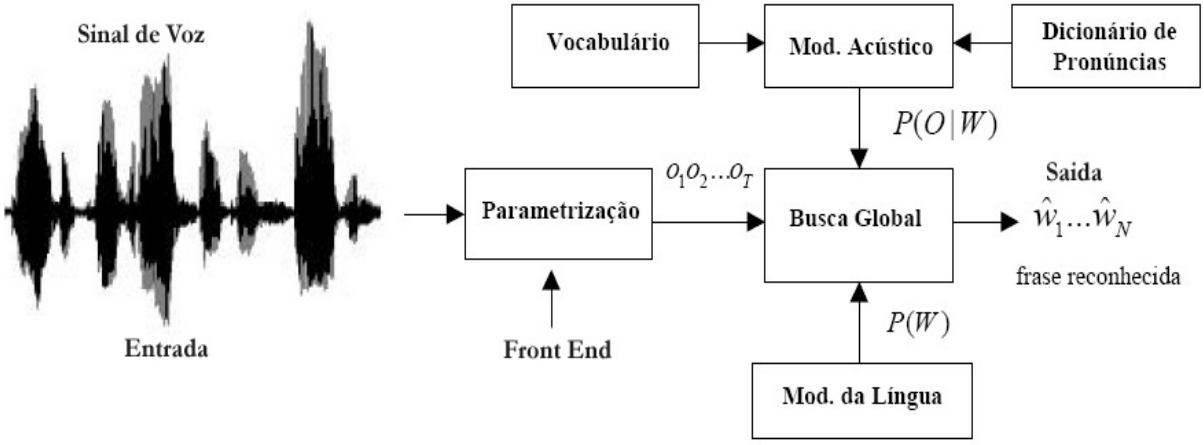


Figura 4.1 - Blocos componentes de um sistema de reconhecimento de voz (RAV).

Costuma-se representar a voz por uma seqüência de vetores de parâmetros extraídos do sinal de fala (*mel-cepstrais*, PLP, etc.) [2], conforme ilustra a Figura 4.2. A seqüência de vetores de parâmetros acústicos é representada por $O = \{O_1, ..., O_T\}$.

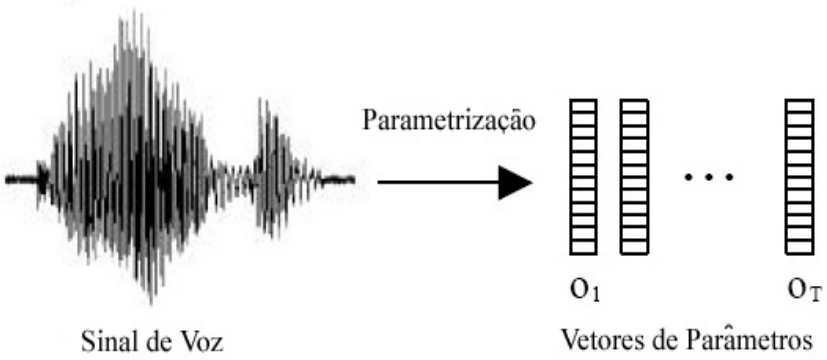


Figura 4.2 - Processo de parametrização.

Para encontrar a seqüência de palavras $\hat{W} = \hat{w}_1 ... \hat{w}_N$ aplica-se o critério da máxima probabilidade a posteriori:

$$\hat{W} = \arg_W \max P(W \mid O)$$

Aplicando a regra de *Bayes*, podemos decompor e escrevê-la em termos mais apropriados:

$$\widehat{W} = \arg_W \max\left\{\frac{P(O \mid W)P(W)}{P(O)}\right\}$$

O termo P(O) é constante para qualquer seqüência de palavras testada e por isso pode ser retirado do processo de decisão. A seqüência $\widehat{W}$ corresponderá à seqüência $\widehat{W}$ que maximiza o produto P(O|W).P(W), conforme definido por:

$$\widehat{W} = \arg_W \max\{P(O \mid W)P(W)\}$$

O termo P(O|W) é avaliado pelo Modelo Acústico e representa a probabilidade do modelo da sentença W gerar a seqüência observada de vetores O. O termo P(W) é avaliado pelo Modelo da Língua e consiste na probabilidade a priori de observar a seqüência de palavras $\widehat{W}$, independente do sinal observado. Em linhas gerais o processo de reconhecimento pode ser dividido nas seguintes fases:

- Aquisição do sinal de voz: feita através de um dispositivo conversor analógico/digital, obtendo-se o sinal a ser reconhecido;
- Extração de parâmetros: adquirido o sinal, o mesmo será representado, através de algum algoritmo de parametrização, por um conjunto de características que descrevem de maneira adequada as propriedades do sinal da voz;
- Reconhecimento do Padrão: após a extração das características do padrão, esta fase responsabiliza-se pela identificação dos mesmos, isto é, verifica a que padrão de referência (conhecidos) o padrão de entrada (o qual se deseja reconhecer) se assemelha.

4.3. Implementação do Sistema LVCSR para o Português Brasileiro

O primeiro estágio de qualquer projeto de desenvolvimento de reconhecedor é a preparação dos dados. São necessários dois conjuntos disjuntos de dados de voz digitalizada (um para treino e outro para o teste do sistema) e cada arquivo de voz

possui transcrição ao nível de palavras e/ou ao nível de fonemas. Além deste *corpus* é necessário um dicionário de pronúncias de grande vocabulário, descrito com mais detalhes no próximo capítulo. Neste trabalho foi utilizado o *corpus* de vozes Spoltech, desenvolvido pela UFRGS e OGI.

4.3.1. Descrição do Corpus Spoltech Versão 1.0

O Spoltech é um *corpus* para o Português Brasileiro. O projeto de criação deste *corpus* contou com o auxílio de pesquisadores, professores e estudantes do Instituto de Informática e de Letras da Universidade Federal do Rio Grande do Sul (UFRGS), Departamento de Informática da Universidade de Caxias do Sul, CSLR/CU (*University of Colorado, Boulder*) e CSLU/OGI (*Oregon Graduate Institute*). A base encontra-se atualmente distribuída pelo LDC (*Linguistic Data Consortium*) [11] e possui vozes gravadas por microfone, de diversas regiões do Brasil, com suas transcrições fonética ou ortográfica.

As expressões gravadas abrangem dois estilos de pronúncia: sentenças lidas, com o intuito de reproduzir voz bem articulada e pronunciada sem hesitações; e respostas a perguntas, com intuito de reproduzir a fala espontânea. As sentenças lidas em Português Brasileiro são similares as encontradas no *corpus* TIMIT, assim como as respostas às perguntas (por exemplo: qual o seu nome, seu endereço, número de telefone, número do CEP, e outras informações). O *corpus* é composto por vozes de 477 falantes e um total de 8080 expressões gravadas, das quais 2540 possuem transcrição ao nível de palavras (não alinhadas no tempo), e 5479 são transcritas em nível de fonemas (alinhadas no tempo).

Quanto à estrutura do diretório, o *corpus* apresenta, no nível mais alto, diretórios nomeados *speech*, *trans*, *lables*, *misc*, *docs*. Cada falante possui seu próprio diretório, com uma sua seqüência de identificação, consistindo de duas letras “BR”, seguido por um traço’-’, e 5 dígitos. As duas primeiras letras identificando o *corpus* em português brasileiro, e os cinco dígitos identificam o falante dentro do *corpus*. O diretório *lables* contém as transcrições fonéticas alinhadas no tempo (formato PHN), o diretório *trans*

contém a transcrições ortográficas não alinhadas no tempo (formato TXT) e o diretório *speech* os arquivos de voz (no formato WAV 44.1 kHz).

Por exemplo, a notação do arquivo *BR-00120.birtplac.wav* indica que a voz é do falante 120, o qual respondeu a pergunta "onde você nasceu?". E o arquivo pode ser localizado no seguinte diretório /*speech/BR-00120/*. A identificação da expressão gravada é feita por uma string que referencia a voz gravada como sendo uma determinada resposta à pergunta ou uma frase lida.

- Frases Balanceadas

Como primeiro passo no desenvolvimento de um reconhecedor de fala é indispensável e produtivo checar o controle da consistência lingüística do *corpus* em uso. Para que se tenha um bom funcionamento da base fonética, é necessário que, o *corpus* tenha uma ampla cobertura dos sons da língua, em contextos variados.

No Splotech a escolha de frases balanceadas foi a estratégia usada para a criação de um *corpus* capaz de cobrir todos os fonemas e alofones da língua. Um exemplo de frase escolhida foi:

"O presidente da república faz advertência ao ministro da justiça"

Para a qual a transcrição fonética (supondo um falante regular do Português Brasileiro) é:
/u pc p r e z i d e tS i d a x e pc p u b l i kc k a f a z a dZ i v e r tc t e s ja w m i n i s t r u d a Z u s tS i s a/

- Perguntas

O *corpus* Spoltech contém respostas a uma série de perguntas do tipo, qual o nome do falante, endereço, comida preferida, número de telefone, CEP, e assim por diante. Nota-se que as respostas sobre o nome completo, apresentam uma grande incidência de sobrenomes estrangeiros, estão transcritos acusticamente, mas não possuem regras de transcrição definidas pela fonologia do Português Brasileiro.

Vale a pena ressaltar que o *corpus* Spoltech possui algumas restrições, pois muitos arquivos WAV apresentam apenas transcrições fonéticas (PHN) e não ao nível de palavras (TXT) e vice-versa. Outro aspecto problemático é a existência de arquivos de voz apresentando falhas, como ruídos como risada, fala não clara e etc. Além de transcrições ortográficas (TXT) e fonéticas (PHN) com erros.

4.3.2. Dicionário Fonético

Tal dicionário provê, para cada palavra no vocabulário do reconhecedor, uma ou mais possíveis transcrições fonéticas. Através de um dicionário base (UFPAdic1.0 com 11.827 palavras) e utilização do aprendizado de máquina para extração de regras de transcrição, foi possível expandirmos o dicionário base e criarmos um dicionário grande o suficiente (por exemplo, com suporte para 60.000 palavras) necessário para o desenvolvimento do sistema LVCSR para o PB. O capítulo anterior apresentou os passos seguidos para criação de tal dicionário.

4.4. HTK - The Hidden Markov Model Toolkit

O HTK [12] é um *software* usado para modelar um sistema de reconhecimento de voz baseado em modelo oculto de Markov (HMM). HMM é um modelo matemático que representa a voz como um conjunto de vetores de observação provenientes de uma função probabilística e uma cadeia de estados conectados entre si, através de transições, com cada transição também sendo governada por uma medida de probabilidade. Deste modo, existem dois processos estocásticos associados a HMMs: um envolvendo as transições entre os estados e o outro envolvendo as observações de saída, como sendo manifestações do fenômeno sendo modelado.

Uma estrutura para o os modelos ocultos de Markov, usualmente (mas nem sempre) utilizada no processamento acústico (modelamento do sinal), é a estrutura *left-right* de 3 estados, Figura 4.3.

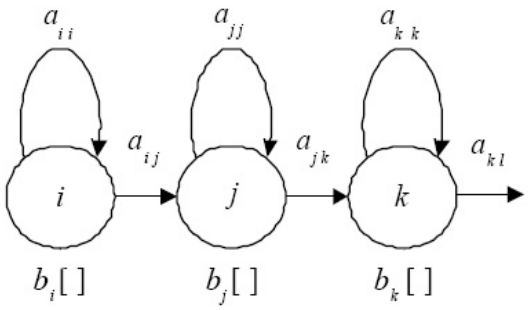


Figura 4.3 - Estrutura *left-right* para um HMM com 3 estados.

Os índices representados pela letra "a" indicam a distribuição de probabilidade de transição de estados e os índices representados pela letra "b" indicam a distribuição de probabilidades dos símbolos observdos no estado corrente. A estrutura *left-right* é bastante usada para aplicações de reconhecimento de voz por possuir somente transições da esquerda a direita que modelam as propriedades progressives dos sinais de voz.

Para essa estrutura, a matriz de probabilidade de transição de estados A terá alguns coeficientes nulos e, portanto, poderá ser simplificada conforme exemplificação da Equação a seguir. Seguindo a tendência geral, esta estrutura foi utilizada neste trabalho.

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

O HTK foi inicialmente projetado para construção de HMMs, por isso grande parte da infraestrutura de suporte no HTK é dedicada para esta tarefa. Além do projeto de HMMs, o *software* possui rotinas desde a análise do sinal até a fase de cálculo de desempenho. A Figura 4.4 mostra os dois principais estágios de processamento deste *software*. No primeiro estágio, as ferramentas de treinamento do HTK são usadas para estimar os parâmetros de um conjunto de HMMs usando treinamento de declarações e suas transcrições associadas. E no segundo estágio, declarações desconhecidas são transcritas usando as ferramentas de reconhecimento do HTK.

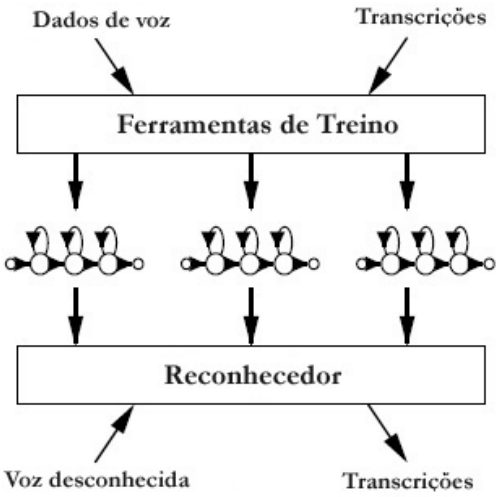


Figura 4.4 - Fundamentos do HTK.

4.4.1. Análise do Sinal

Para análise do sinal, fase na qual são extraídos os parâmetros relevantes a serem utilizados nas próximas fases do sistema, é utilizada a função *HCopy*, Figura 4.5.

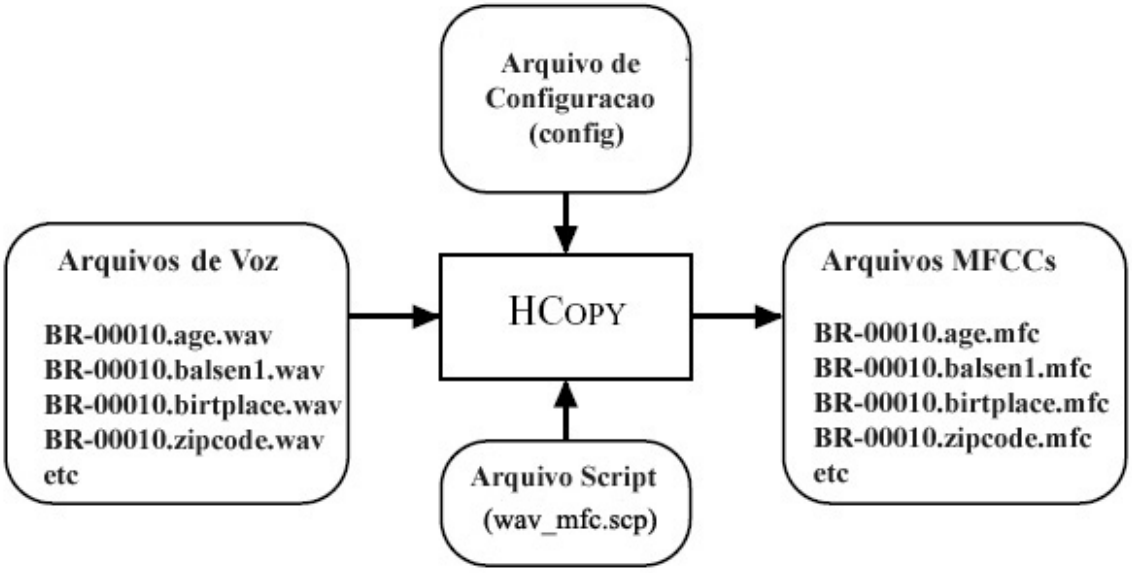


Figura 4.5 - Funcionamento do *HCopy*.

A função *HCopy* é evocada pela seguinte linha de comando:

» HCopy –C config –S wav_mfc.scp

O arquivo *config* contém as configurações utilizadas pelo procedimento *HCopy* (extração de parâmetros). A opção –S *wav_mfc.scp* é um *script* que possui a lista de arquivos de voz de treino e teste a serem parametrizados.

Arquivo *config*:

```
TARGETKIND =  MFCC_E_D_A
TARGETRATE = 100000.0
NUMCEPS = 12
CEPLIFTER =  22
NUMCHANS = 24
PREEMCOEF =  0.97
USEHAMMING = TRUE
WINDOWSIZE  =  200000.0
TARGETFORMAT =  HTK
SOURCEFORMAT = WAV
SOURCEKIND = WAVEFORM
```

Os parâmetros SOURCEKIND e SOURCEFORMAT definem o formato de arquivo de entrada. Da mesma maneira, os parâmetros TARGETKIND e TARGETFORMAT, definem o formato do arquivo de saída. Os valores WAVFORM, WAV definem que o arquivo de entrada é um sinal codificado do tipo WAV *windows*. Os valores, MFCC_E_D_A e HTK definem que o arquivo de saída é do tipo Mel-Cepstrais com adição do sinal de energia, 1ª e 2ª derivada, codificado no formato HTK.

O valor 100000.0 do parâmetro TARGETRATE define que o deslocamento adotado para obtenção da próxima janela é de 10ms. A variável WINDOWSIZE define o tamanho da janela utilizada de 20ms. Visto que o deslocamento entre janelas consecutivas é de 10ms, a superposição entre janelas será de 10ms. Por fim o parâmetro PREEMCOEF define o coeficiente do filtro de pré-ênfase que será de 0,97.

O parâmetro NUMCHANS define o número de canais do banco de filtros cepstrais, CEPFILTER define o número de filtros do banco de filtros cepstrais e NUMCEPS define o número de coeficientes de saída do banco de filtros cepstrais.

Arquivo *wav-mfc.scp*:

./spoltech/speech/BR-00001/BR-00001.age.wav	./spoltech/speech/BR-00001/BR-00001.age.mfc
./spoltech/speech/BR-00001/BR-00001.balsen1.wav	./spoltech/speech/BR-00001/ BR-00001.balsen1.mfc
./spoltech/speech/BR-00001/ BR-00001.birtplac.wav	./spoltech/speech/BR-00001/ BR-00001.birtplac.mfc
./spoltech/speech/BR-00001/ BR-00001.food.wav	./spoltech/speech/BR-00001/ BR-00001.food.mfc

4.4.2. Treinamento do Sistema

Para se fazer o treinamento do sistema, primeiramente procede-se com a inicialização dos modelos HMM. Isto pode ser feito de dois modos distintos dependendo do tipo de segmentação dos dados em trabalho, Figura 4.6.

Perante uma base de dados que já tenha sido feita uma segmentação fonética prévia, esses dados podem ser usados como *bootstrap* [1] para os modelos. Para isto, as ferramentas *HInit* e *HRest* promovem o estilo de treinamento de palavras isoladas (fronteira dos fonemas marcados no domínio do tempo) usando todos as amostras transcritas foneticamente. Desta forma, cada HMM é gerada individualmente. O *HInit* lê todos os dados de treinamento *bootstrap* e executa o processo *cut out* para todos os fonemas contidos nos dados. Então computa interativamente um conjunto inicial de valores de parâmetros usando o procedimento *segmental k-means* [13].

[1] Chama-se *bootstrap* a etapa de inicialização dos modelos. Como o treinamento dos modelos acústicos é iterativo, o *bootstrap* é uma etapa primordial, pois uma vez se tendo os modelos iniciais, algoritmos bem estabelecidos (*Baum-Welch,* por exemplo) são usados para gerar novos (e melhores) modelos a partir dos atuais. O *bootstrap* de HMMs representando fones fica mais fácil quando as transcrições fonéticas estão disponíveis, uma vez que para treinar um dado (fone), pode-se usar exatamente os segmentos de voz associados ao mesmo.

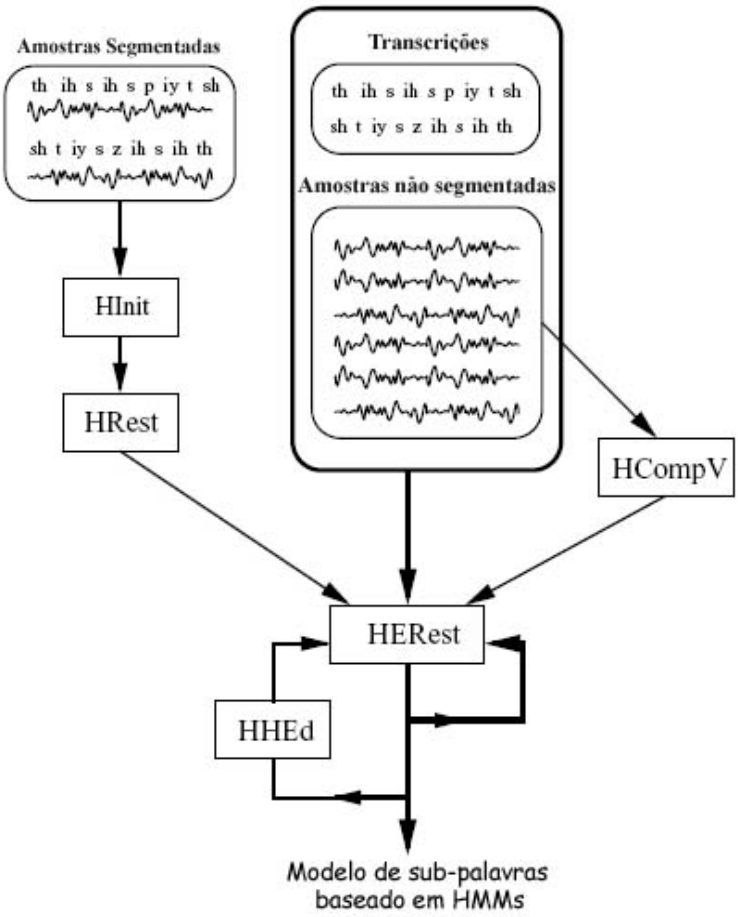


Figura 4.6 - Inicialização e treino dos modelos HMMs.

Na primeira iteração, os dados de treino são segmentados uniformemente, cada estado do modelo é combinado com o correspondente segmento do dado de treino e então as médias e variâncias são estimadas. Se um modelo de mistura de gaussianas tiver sido treinado, então é usada uma forma modificada de *k-means clustering*. Na segunda e demais interações, a segmentação uniforme é substituída pelo algoritmo de alinhamento de *Viterbi* [45]. Os valores iniciais dos parâmetros computados pelo *HInit* são, então, re-estimados pelo *HRest*. Em seguida, novamente, todos os dados transcritos do *bootstrap* são usados, porém ao invés de se usar o algoritmo *segmental k-means*, usa-se a re-estimação por *Baum-Welch* [45].

A outra forma de inicialização dos modelos HMM (inicialmente monofones), usada neste trabalho, ocorre quando não há dados para o *bootstrap*, e usa-se o chamado *flat start*. Neste caso, todos os modelos de fonemas são inicializados de forma idêntica,

possuindo estados com médias e variâncias iguais a média e variância global do conjunto de treino. A ferramenta *HCompV* pode ser usada para esta tarefa.

» HCompV -C config -f 0.01 -S treino.scp -M hmm0 -I phones.mlf proto

Arquivo *proto*:

```
~o <VecSize> 39 <MFCC_0_D_A_Z>
~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
                <Variance> 39
                        1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
        <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
                <Variance> 39
                        1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
        <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
                <Variance> 39
                        1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

        <TransP> 5
                0.0 1.0 0.0 0.0 0.0
                0.0 0.5 0.5 0.0 0.0
                0.0 0.0 0.5 0.5 0.0
                0.0 0.0 0.0 0.5 0.5
                0.0 0.0 0.0 0.0 1.0
<EndHMM>
```

O arquivo proto contém a definição do protótipo HMM, isto é modelo inicial da HMM. A opção –M hmm0 criará uma nova versão para o proto no diretório hmm0. A opção –f 0.01 criará uma macro de variância base (chamada de vFloors) igual a 0.01 vezes a variância global, substituindo as médias igual a zero e as variâncias igual um setadas no protótipo. A opção -S *treino.scp* contém a lista de arquivos de treino a serem processados pela inicialização de modelos via *HCompV*. A opção –I *phones.mlf* contém o arquivo com a transcrição fonética dos vetores de entrada (voz para treino).

Após o conjunto inicial de modelos estar criado, a ferramenta *HERest* é usada para desempenhar o treinamento *embedded*, usando como entrada o conjunto de treino. *HERest* executa simultaneamente uma única re-estimação de *Baum-Welch* sobre todo o conjunto de modelos de fonemas baseados em HMMs. Para as amostras de treino os correspondentes modelos de fonemas são concatenados e repassados ao algoritmo *forward/backward* que acumula as estatísticas de ocupação de estados, médias, variâncias, entre outros, para cada HMM.

Quando todos os dados de treino são processados, as estatísticas acumuladas são usadas para computar os parâmetros das HMMs re-estimadas. O *HERest* é o núcleo das ferramentas de treinamento do HTK, sendo projetado para processar uma grande quantidade de banco de dados, tendo facilidades de *pruning*, reduzindo complexidade computacional, e podendo ser executado em paralelo numa rede de computadores. A Figura 4.7 mostra como funciona este procedimento, invocado pela seguinte linha de comando:

» HERest -d hmm0 -C config -I phones.mlf -t (3.2.1) -S treino.scp -H hmm0/macros -H hmm0/hmmdefs -M hmm1 monophones0

A opção –d hmm0 indica onde encontrar as definições dos monofones gerados anteriormente com o *HCompV*. A opção –t admite setar 3 valores como limiar de poda (*pruning)*. A opção –H hmm0/macros -H hmm0/hmmdefs roda o arquivo de definições das HMMs contida nestes diretórios. A opção –M hmm1 armazena a saída neste novo

diretório. E o arquivo *monophones0* contém a lista de fones de acordo com a seqüência de modelos HMM dispostas nos arquivos descritos pela opção –H.

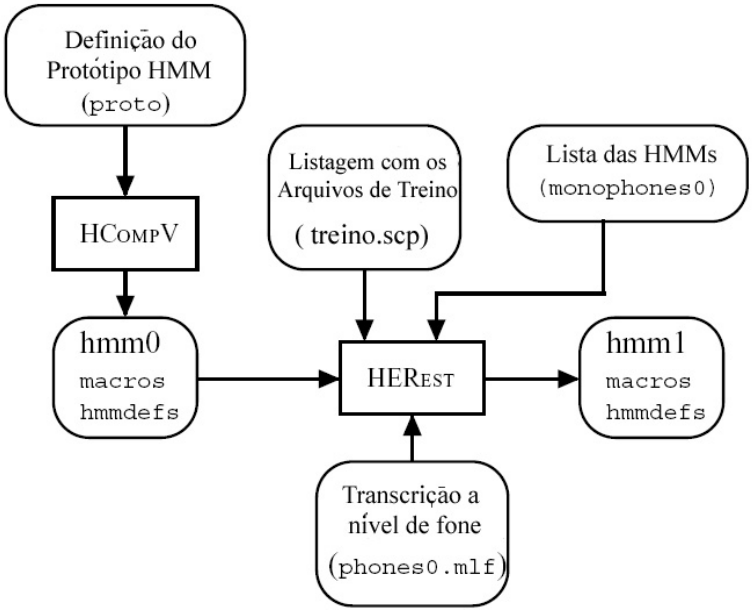


Figura 4.7 - Funcionamento do *HERest*.

4.4.3 Reconhecimento e Medidas de Desempenho

Para se fazer a medida de desempenho, primeiramente é feito o reconhecimento dos dados de voz desconhecidos do *corpus* Spoltech, a partr dos modelos HMM previamente treinados. Deste modo, é utilizado o arquivo que contém as definições do último modelo HMM treinado, no caso da figura 4.7 chamado de hmm1. O reconhecimento das frases é feito utilizando o algoritmo de *Viterbi*. Este algoritmo é implementado na função *HVite*. Esta função é executada como mostra a linha de comando a seguir:

» HVite -S teste.scp -H hmm1/macros -H hmm1/hmmdefs –i results.mlf –w wdnet dict monophones0

Onde *teste.scp* é o arquivo que contém o caminho das locuções a serem reconhecidas. A opção –H hmm0/macros -H hmm0/hmmdefs armazena as definições do

último modelo de HMMs estimado. O arquivo *results.mlf* será o arquivo de saída, contendo o resultado do reconhecimento. O arquivo *wdnet* contém o modelo de linguagem, que vai orientar o processo de busca. O arquivo *dict* é o arquivo que contém o vocabulário do sistema de forma transcrita (dicionário fonético).

De posse do resultado obtido pela função *HVite*, o arquivo *results.mlf*, executa-se a função *HResults* que vai medir o desempenho do sistema:

» HResults –I words_test.mlf monophones0 results.mlf

Esta função faz a comparação das palavras reconhecidas, contidas no arquivo *results.mlf* com as transcrições ortográficas originais *d*os arquivos de voz usados para teste, *words_test.mlf*. É feito um alinhamento entre a palavra reconhecida e a transcrição original correspondente e calculada, assim, a taxa de acerto das palavras reconhecidas.

Arquivo *HResults*:

```
====================== HTK Results Analysis =======================
  Date: Thu May 25 00:15:31 2006
  Ref : words_test.mlf
  Rec : results_chadia_24e16g.mlf
------------------------ ----------------------- Overall Results ---------------------------------------------
SENT: %Correct=7.58 [H=37, S=451, N=488]
WORD: %Corr=60.56, Acc=41.86 [H=2749, D=259, S=1531, I=849, N=4539]
====================================================================
```

A linha que começa com SENT: indica que das 451 expressões usadas para teste, 34 (7.58%) foram corretamente reconhecidas. A linha seguinte que começa com a palava WORD: indica as estatísticas ao nível de palavra onde num total de 4539, 2749 (60.56%) foram corretamente reconhecidas. Houve 259 erros por deleção (D), 1531 substituições (S) e 849 inserções (I). A precisão (Acc) de 41.86 % é menor que a percentagem correta (Cor), pois leva em considerações os erros por inserção que por sua vez é ignorado por (Cor).

A Figura 4.8, resume os dois últimos procedimentos *HVite* e *HResults*.

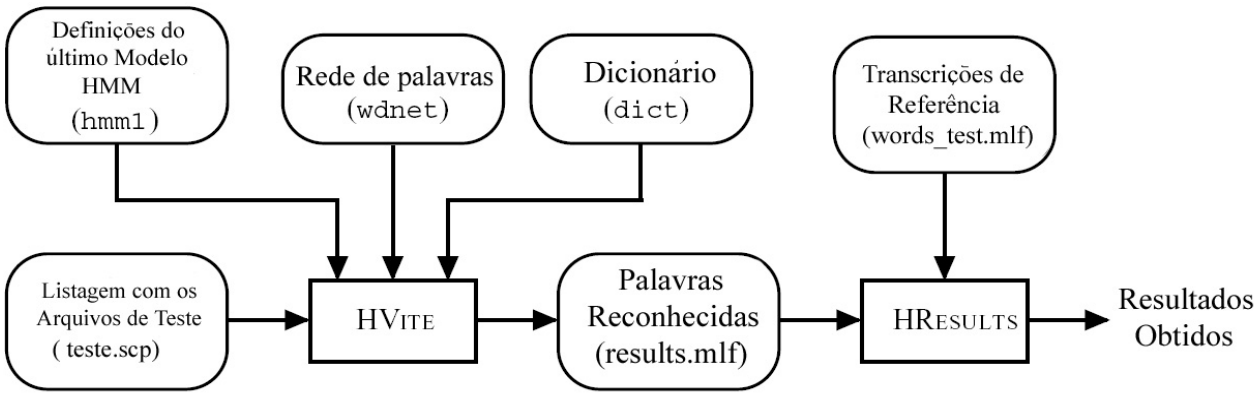


Figura 4.8 - Funcionamento do *HVite* e *HResults*.

Vale citar que a filosofia de criação de sistemas no HTK permite que as HMMs sejam refinadas gradualmente. Sendo uma típica progressão, inicializar o sistema com um simples conjunto de modelos baseados em fonemas, representados por uma única gaussiana e independente de contexto. A partir de então realizar uma sofisticação gradual de tais modelos, transformando-os em modelos dependentes do contexto, representados por uma mistura de gaussianas.

Por exemplo, para tornar o modelo de silêncio mais robusto, cria-se o modelo sp (*short pause*), através do compartilhamento do estado central do modelo so sil, Figura 4.9. Esta alteração permite que os estados individuais que representam o silêncio absorvam a variedade de ruídos impulsivos presentes nos dados de treino. O salto de volta do estado 4 ao 2, permite que a transição para a próxima palavra ocorra sem comprometer o modelo. O arquivo utilizado para proceder tal ateração é o *sil.hed*.

Arquivo *sil.hed*:

```
AT 2 4 0.2 {sil.transP}
AT 4 2 0.2 {sil.transP}
AT 1 3 0.3 {sp.transP}
TI silst {sil.state[3],sp.state[2]}
```

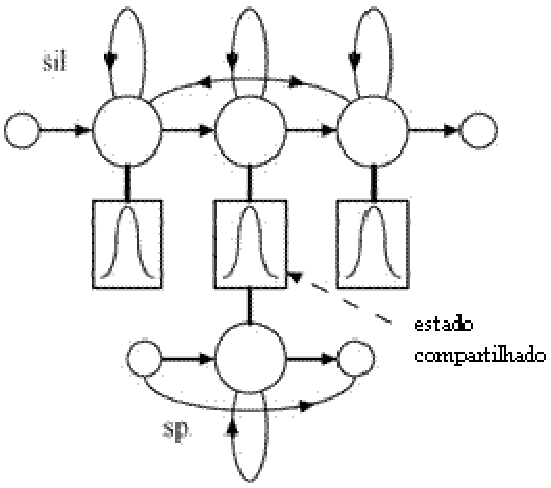


Figura 4.9 - Consertando o modelo de silêncio.

Para tal função a ferramenta *HHEd* é apresentada como um editor de definições de HMM, Figura 4.10. Esta ferramenta clona os modelos transformando-os em dependentes do contexto, permite aplicar uma variedade de parâmetros de junção (*tying*) e incrementa a distribuição especificada para mistura de gaussianas. Uma rotina largamente empregada, é modificar o conjunto de HMMs em estágios usando *HHEd*, e então, re-estimar os parâmetros dos conjuntos modificados usando *HERest* depois de cada estágio.

» HHEd –H hmm(X-1)/macros hmm(X-1)/hmmdefs –M sil.hed monophones1

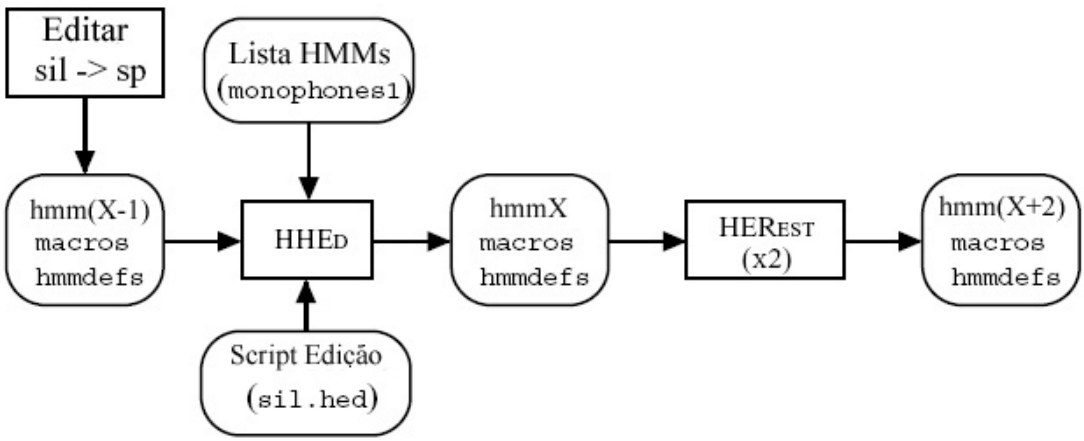


Figura 4.10 - Caracterização do *HHEd*.

Depois de corrigido, o modelo de silêncio, um editor de texto é usado para modificar as definições do modelo hmm(X-1). O modelo com as novas definições das HMMs (incluindo o modelo sp) é armazenado em hmm(X). A seguir, os modelo HMMs são re-estimados, usando a ferramenta HERest, e o último modelo HMM será usado para reconhecimento dos dados de voz desconhecidos, descrito no próximo item. Percebe-se que o arquivo *monophones0* é alterado para *monophones1*, este último contém o novo modelo sp.

Quanto mais complexo é o conjunto de modelos, mais dados são necessários para a realização de uma estimação robusta de parâmetros e um grande problema na construção de modelos HMMs dependentes do contexto é que, geralmente, os dados de treino são insuficientes. Devido à limitação do conjunto de treino, um equilíbrio entre complexidade e dados disponíveis deve ser feito. Para sistemas de densidade contínua, esse balanço é obtido pela amarração (*tying*) de parâmetros, permitindo que dados sejam unidos, compartilhando parâmetros e permitindo, assim, uma estimação mais robusta.

4.5 Resultado das Simulações

No processo de extração de parâmetros (*front end*) foram utilizados os consagrados MFCCs (*Mel-Frequency Cepstrum Coefficients*), com 12 parâmetros "estáticos", a energia e estimativas das duas primeiras derivadas, compondo um total de 39 parâmetros por quadro, como descrito anteriormente. A duração do quadro foi de 20 milisegundos (ms), com um deslocamento de 10 ms. Os modelos acústicos, para monofones, foram construídos baseados em HMMs, com diferente números de de gaussianas por estado. Para avaliação do desempenho dos diferentes modelos observa-se a taxa de erro por palavras ou WER (W*ord Error Rate*). Parte dos passos seguidos para preparação dos dados, criação dos modelos acústicos, de linguagem e testes do reconhecedor implementado podem ser vistos no Apêndice 5.

A Tabela 4.1 mostra o comportamento dos modelos estimados, baseados em monofones e a Figura 4.11 mostra o gráfico criado a partir destes resultados. Importante

citar que o modelo de linguagem bigrama usado neste trabalho foi criado a partir das expressões contidas no próprio *corpus* Spoltech.

Número de Estimações e Gaussianas/Mixtura	Taxa de erro por palavra WER (%)
6e1g	73.52
9e2g	73.14
12e3g	59.99
15e5g	52.59
18e8g	47.12
21e12g	42.01
24e16g	39.44

Tabela 4.1 – Desempenho do sistema LVCSR baseado em monofones, variando o número de estimações e gaussianas por mistura.

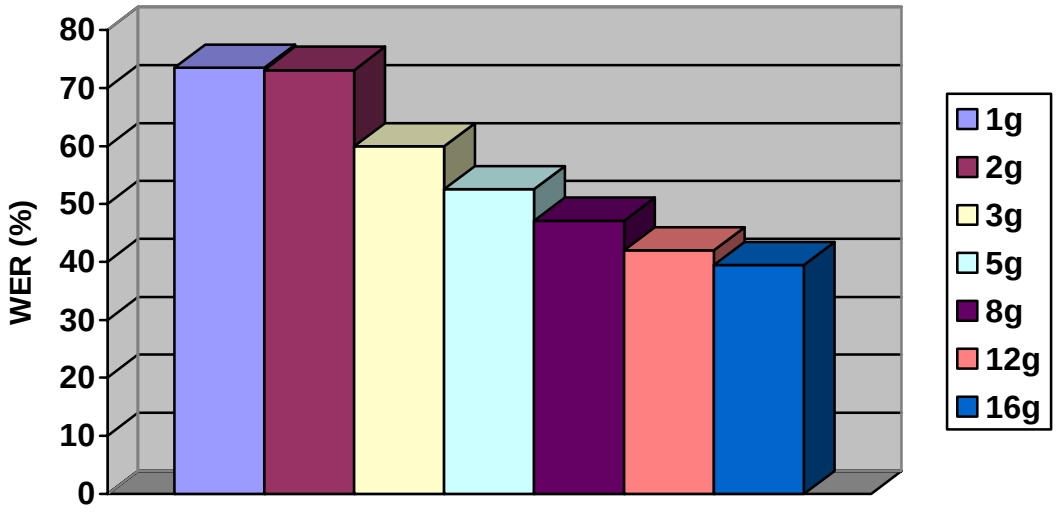


Figura 4.11 – Taxa de erro por palavra WER (%) variando o número de gaussianas por mistura para treino das HMMs.

A WER relativamente grande pode estar relacionada a diversos fatores como, a grande quantidade de palavras a serem reconhecidas, a disponibilidade de uma pequena quantidade de voz para o treinamento do modelo acústico, a existência de diversos arquivos de áudio com ruído (tipo vozes de outros locutores, sons de equipamentos, ar condicionado, e, até mesmo, provocado pelo próprio locutor, tais como tosses, espirros,

estalo dos lábios, suspiro, respiração forte, etc.), os quais deverão ser descartados sistema e, para isto, medidas quanto a verificação da consistência dos arquivos de áudio e transcrição ortográfica do *corpus* Spoltech já estão sendo realizadas.

Procura-se obter uma melhora significativa no desempenho do sistema após a verificação da consistência dos dados contidos no *corpus* Spoltech, evolução dos modelos HMM de monofones para trifones, criação de um modelo de linguagem mais abrangente baseado em expressões diversas não só as contidas no *corpus*, etc.

5. CONCLUSÃO

A construção de um sistema de Reconhecimento de Voz com suporte a grandes vocabulários é objeto de estudo e um constante desafio para muitos pesquisadores da área de processamento de voz. A quantidade de publicações e de pesquisadores dedicados a este apaixonante conhecimento técnico-científico tem crescido de forma impressionante nos últimos anos.

Atualmente, os maiores obstáculos para desenvolvimento de sistemas LVCSR para o PB, podem estar relacionados com a falta de um *corpus* de voz digitalizado e transcrito, grande o suficiente para o treinamento de modelos acústicos e de recursos específios ao PB, tal como um dicionário fonético. Frente a estas e outras adversidades, uma das metas deste trabalho é a disseminação dos recursos utilizados para a criação do LVCSR para o PB e permitir assim a reprodução dos resultados em diversas localidades. Em suma, facilitar as atividades dos que desejam atuar em reconhecimento de voz, mas que atualmente devem vencer diversas barreiras para a configuração de um sistema básico.

5.1. Discussão Geral

O presente trabalho apresentou o desenvolvimento de um sistema de reconhecimento de voz para o Português Brasileiro com suporte a grandes vocabulários e de um sistema de conversão grafema-fone, utilizado para criação do dicionário fonético. Para a concretização da primeira tarefa foi utilizado o *corpus* Spoltech e o *software* HTK para processar as rotinas desde a análise do sinal de voz até a fase de cálculo do desempenho. A ferramenta HTK (linguagem C) foi escolhida devido a sua credibilidde com relação à modelagem de sistemas de reconhecimento de voz, mas vale lembrar que existem outros bons *toolkits* de domínio público para desenvolvimento de sistemas LVCSR, tais como o Sphinx 4 [41] (Java), ISIP [42] (C++), Festival [43],

SONIC [44], etc. Para processamento do segundo módulo (conversor grafema-fone), foram utilizadas técnicas de aprendizado de máquina como forma de extrair regras de transcrição, a partir de um léxico.

Analisando os resultados obtidos pelo LVCSR criado, pode-se verificar que os experimentos realizados foram satisfatórios, constatando uma boa taxa de acerto nos sistemas testados, considerando a carência de um *corpus* de voz digitalizada adequado para a etapa de modelagem acústica, destacando-se alguns problemas detectados na base de dados, citados no subitem 4.3.1.

Quanto ao sistema de conversão grafema-fone, os resultados alcançados pelas simulações são compatíveis com os encontrados na literatura para todos os diconários testados. Mesmo dispondo de um *corpus* pequeno para treino, testes com o dicionário UFPAdic1.0 (PB), os resultados obtidos foram satisfatórios, haja visto, tratar-se de uma língua com alto grau de foneticidade.

5.2. Sugestões para Trabalhos Futuros

Como sugestões para trabalhos futuros, destacamos:

- Melhoria do *corpus* Spoltech, já em procedimento.

Para tal, criou-se recentemente, no Laboratório de Processamento de Sinais (LaPs) da UFPa, um programa que auxiliará na verificação da consistência dos arquivos de voz e transcrições referentes contidas no *corpus* Spoltech. Após esta verificação, será possível descartarmos os arquivos que estiverem apresentando problemas, por exemplo, qualquer tipo de ruído (risada, fala não clara e etc.). Simultaneamente à verificação do audio, está sendo feita a análise quanto a transcrição ortográfica referente a cada arquivo sendo escutado. Nesta fase estão sendo feitas, quando necessário, correções das transcrições existentes e sendo inseridas novas transcrições, haja vista, o *corpus* original apresentar apenas 2540 arquivos com trascrição ortográfica de um total de 8080 arquivos de voz.

Também são propostas melhorias para cada um dos blocos componente do sistema RAV (dicionário fonético, modelo acústico e modelo de linguagem):

- Dicionário fonético

Vale citar que os experimentos aqui realizados para conversão grafema-fone, utilizados posteriormente para a criação do diconário fonético, são relativamente simples e simulações mais elaboradas para esta conversão podem ser realizadas levando em consideração característica como, silabificação, tonicidade, etc. Estas análises devem ser incorporadas ao sistema a ser apresentado no desafio proposto pelo *Pronalsyl* [31]. Um grande dicionário fonético com transcrições concisas para o Português do Brasil é imprescindível para o sistema LVCSR proposto.

- Modelo acústico

Existem diversas técnicas que podem ser depuradas para melhorar o modelo acústico, tal como evolução dos modelos HMM de monofones para trifones, agrupamento usando árvores de decisão, etc.

- Modelo de Linguagem

Como mencionado, os resultados obtidos pelo LVCSR criado para o PB, o modelo de linguagem bigrama criado para o sistema LVCSR deste trabalho, leva em consideração apenas as expressões contidas no *corpus* Spoltech, o que não é indicado, pois restringe a capacidade do reconhecedor, caso diferentes expressões sejam inseridas no *corpus*.

A criação de um modelo de linguagem mais abrangente, contendo expressões diversas, já está sendo providenciada e para isso novos textos estão sendo obtidos via *crawling* da Internet, e serão incluídos no novo modelo de linguagem.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] http://www.laps.ufpa.br/falabrasil, (10/03/2006)

[2] **Rabiner, L., and Juang, B. H**., *Fundamentals of speech recognition*. Prentice Hall, 1993.

[3] **Sá, F. et al.**, *Reconhecimento Automático de Fala Contínua em Português Europeu Recorrendo a Streams Audio-Visuais*. Workshop de Sistemas de Informação Multimédia, Cooperativos e Distribuídos CoopMedia Porto, 2003.

[4] **Rudnicky, A. I., Hauptmann, A. G. and Lee, K. F**., Survey of Current Speech Technology". http://www.lti.cs.cmu.edu/Research/cmt-tech-reports.html.

[5] **Moore, R. K.**, *Speculating on the Future for Automatic Speech Recognition*, IEEE workshop on Automatic Speech Recognition and Understanding (ASRU), St. Thomas, US Virgin Islands, 2003.

[6] **Kurzweil, R.**, "*The Age of Spiritual Machines*", Phoenix, 1999.

[7] **Moore R. K.**, *Results from a survey of attendees at ASRU 1997 and 2003*, Proc. INTERSPEECH 2005 Lisbon, 5-9 September 2005.

[8] **Moore, R. K.,** *A comparison of the data requirements of automatic speech recognition systems and human listeners*, Proc. Eurospeech, Geneva, pp. 2582-2584, 1-4 September, 2003.

[9] **Keller, E.,** *Towards Greater Naturalness: Future Directions of Research in Speech Synthesis*, Improvements in Speech Synthesis, Keller, E., Bailly, G, Monaghan, A., Terken, J. and Huckvale, M. (eds.), Wiley & Sons, Chichester, UK, 2001.

[10] **Vaissière, J.,** *Speech recognition: a tutorial in Computer Speech Processing,* Fallside, F., Woods, W.A., Prentice-Hall International, University of Cambridge, 1983.

[11] http://www.ldc.upenn.edu. (14/04/2006).

[12] **Young, S. et. al.** "*The HTK book. Microsoft Corporation"*. Version 3.0, 2000. http://htp.eng.cam.ac.uk. (08/09/2005)

[13] **Rabiner, L. R. and Wilpon, J. G. and Juang, B. H.,** "*A Segmental k-Means Training Procedure for Connected Word Recognition*", AT&T Tech. Journal., Vol. 65, 1986.

[14] **Trancoso, I. and Viana, M. and Silva, F.**, *On the pronunciation of common lexica and proper names in European Portuguese*, in 2nd Onomastica Res. Colloq, 1994.

[15] **Bouma, G.**, *Comparison of two tree-structured approaches for grapheme-to-phoneme conversion*, http://citeseer.ist.psu.edu/article/bouma00finite.html, in 1st Meeting of the North-American Chapter of the Association for Computational Linguistics, Seattle, 2000.

[16] **Teixeira, A. and Oliveira, C. and Moutinho, L.**, *On the use of machine learning and syllable information in european portuguese graphemephone conversion*, in 7th Workshop on Computational Processing of Written and Spoken Portuguese (to be presented) - Itatiaia, Brazil, 2006.

[17] **Dedina, M. J. and Nusbaum, H. C.,** "*PRONOUNCE: A program for pronunciation by analogy*," Computer Speech and Language, pp. 5:55– 64, 1991.

[18] **Damper, R. I. and Marchand, Y. and Marsters, J. and Bazin, A.,** "*Can syllabification improve pronunciation by analogy of English?*" Natural Language Engineering, pp. 1–25, 2005.

[19] **Sejnowski, T. J. and Rosenberg, C. R.,** "*NETtalk: Parallel networks that learn to pronounce English text*". Complex Systems 1:145-168. 1987.

[20] **Bakiri, G., and Dietterich, T. G.,** *Converting English Text to Speech: A MachineLearning Approach*. Ph.D. thesis.Rep. No. 91-30-1. Department of Computer Science, Oregon State University. 1991.

[21] **Lucas, S. M. and Damper, R. I.,** "*Syntactic neural networks for bi-directional text-phonetics translation*". In G. Bailly and C. Benoit, editors, TalkingMachines, Theories, Models and Designs. North-Holland Publishers. 1992.

[22] **Hochberg, J. and Mniszewski, S. M. and Calleja, T. and Papcun, G. J.**, "*A default hierarchy for pronouncing English*". IEEE Transactions on Pattern Matching and Machine Intelligence 13(9): 957-964. 1991

[23] **Klatt, D. H. and Shipman, D. W.**, *Letter-to-phoneme rules: A semi-automatic discovery procedure*. Journal of the Acoustical Society of America 82: 737-793. 1982.

[24] **Klatt, D. H.,** *Review of text to speech conversion for English*. Journal of the Acoustical Society of America 82(3): 737-793. 1987.

[25] **Parfitt, S. and Sharman, R.,** *A bi-directional model of English pronunciation*. In Proceedings of Eurospeech, volume 2, pages 801-804. 1991.

[26] **Mamede, N. J. and Baptista, J. and Trancoso, I. and Nunes, M. G. V.,** "*Computational processing of the portuguese language*", 6th international workshop, propor 2003, faro, portugal, june 26-27, 2003. proceedings," in PROPOR. Springer, 2003, pp. 23–30.

[27] **Oliveira, L. C.,** "*Síntese de Fala a Partir de Texto*". Tese de Doutoramento. Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal, 1996.

[28] **Barbosa, F. et al.,** *Grapheme-Phone Transcription Algorithm for a Brazilian Portuguese TTS*. In: N. Mamede et al. (Eds): PROPOR 2003, LNAI 2721, Springer Verlag, 2003, 23-30.

[29] **Albano, E. and Moreira, A.,** *Archisegment-based Letter-to-Phone Conversion for Concatenative Speech Synthesis in Portuguese*. In Proceedings of ICSLP 96, vol.3,1996, 1708-1711.

[30] **Barbosa, P. et al.,** *Aiuruetê: a high-quality concatenative text-to-speech system for brazilian portuguese with demisyllabic analysis-based units and hierarchical model of rhythm production*, in Proceedings of the Eurospeech'99, pp. 2059–2062, Budapest, Hungary, 1999.

[31] http://www.pascal-network.org/challenges/pronalsyl. (20/01/2006).

[32] **Cunha, C. e Cintra, L.,** *Nova Gramática do Português Moderno*. Edições João Sá da Costa: Lisboa, 2000.

[33] http://www.cs.waikato.ac.nz/ml/weka. (10/02/2006).

[34] **Witten, I. H. and Frank, E.**, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*.

[35] **Damper, R. I., Marchand, Y., Marsters, J. and Bazin, A.,** *Aligning letters and phonemes for speech synthesis*, in 5th ISCA Speech Synthesis Workshop - pp. 209–214. Pittsburgh, 2004.

[36] **Content, A., Mousty, P. and Radeau, M.,** *Brulex: Une base de données lexicales informatisée pour le français éecrit et parlé*, L'Année Psychologique, pp. 551–566, 1990.

[37] ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries. (10/11/2005).

[38] **Needleman, S. B. and Wunsch, C. D.,** *An efficient method applicable to the search for similarities in the amino acid sequences of two proteins*, Journal of Molecular Biology,, vol. 48, no. 3, pp. 444–453, 1970.

[39] **Viterbi, A.J.,** *Error bounds for convolutioal codes and asymptotically optimum decoding algorithm*. IEEE Transactions on Information Theory, IT-13(2):260-269. 1967.

[40] **Neuhoff, D. L., "**The Viterbi algorithm as an aid in text recognition". In IEEE Transactions on Information Theory, IT-21:222-226. 1975.

[41] http://www.speech.cs.cmu.edu/sphinx/Sphinx.html. (10/04/2006).

[42] http://www.isip.msstate.edu/project/speech. (10/04/2006)

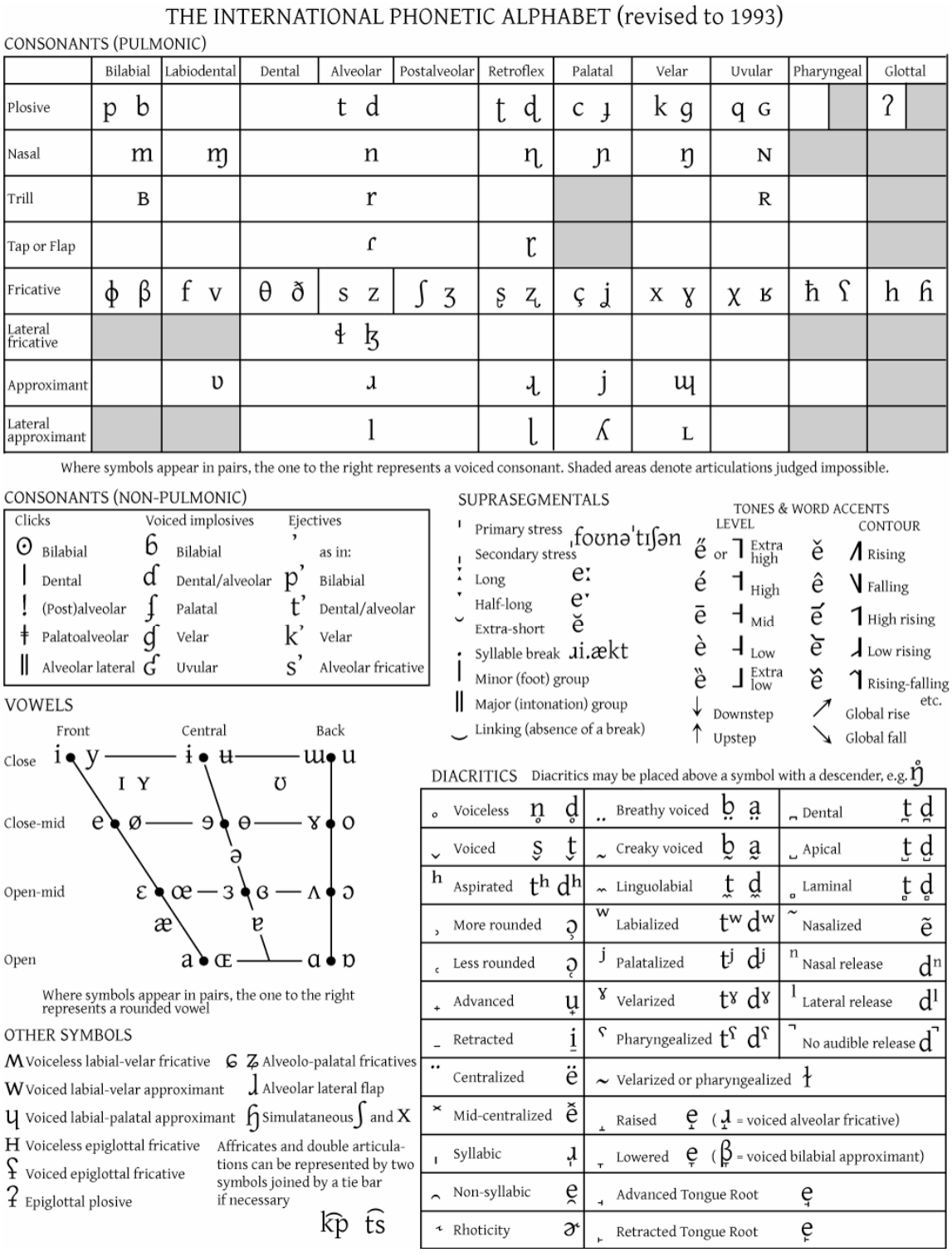
[43] http://www.cstr.ed.ac.uk/projects/festival. (10/04/2006)

[44] **Pellom, B.**, *"Sonic: The University of Colorado Continuous Speech Recognizer"*, Technical Report TR-CSLR-2001-01, CSLR, University of Colorado, March 2001.

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

[46] **Moore, R. K.,**Research Challenges in the Automation of Spoken Language Interaction. Proc. ISCA/COST ASIDE2005 workshop, Aalborg, 10th-11th November 2005.

Apêndice 1 - Alfabeto Fonético Internacional



THE INTERNATIONAL PHONETIC ALPHABET (revised to 1993)

CONSONANTS (PULMONIC)

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k g	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			r					ʀ		
Tap or Flap				ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

CONSONANTS (NON-PULMONIC)

Clicks		Voiced implosives		Ejectives	
ʘ	Bilabial	ɓ	Bilabial	ʼ	as in:
ǀ	Dental	ɗ	Dental/alveolar	pʼ	Bilabial
ǃ	(Post)alveolar	ʄ	Palatal	tʼ	Dental/alveolar
ǂ	Palatoalveolar	ɠ	Velar	kʼ	Velar
ǁ	Alveolar lateral	ʛ	Uvular	sʼ	Alveolar fricative

SUPRASEGMENTALS

ˈ	Primary stress	ˌfoʊnəˈtɪʃən
ˌ	Secondary stress	
ː	Long	eː
ˑ	Half-long	eˑ
◡	Extra-short	ĕ
.	Syllable break	ɹi.ækt
‖	Minor (foot) group	
‖	Major (intonation) group	
‿	Linking (absence of a break)	

TONES & WORD ACCENTS

LEVEL		CONTOUR	
˥ or ˩	Extra high / Extra low	˨˩	Rising
˦	High	˩˥	Falling
˧	Mid	˥˩	High rising
˨	Low	˩˨	Low rising
		˥˩˨	Rising-falling
↓	Downstep	↗	Global rise
↑	Upstep	↘	Global fall

etc.

VOWELS

Where symbols appear in pairs, the one to the right represents a rounded vowel

OTHER SYMBOLS

ʍ Voiceless labial-velar fricative	ɕ ʑ Alveolo-palatal fricatives
w Voiced labial-velar approximant	ɺ Alveolar lateral flap
ɥ Voiced labial-palatal approximant	ʃ͡ʂ Simulataneous ʃ and x
ʜ Voiceless epiglottal fricative	
ʢ Voiced epiglottal fricative	
ʡ Epiglottal plosive	

Affricates and double articulations can be represented by two symbols joined by a tie bar if necessary

k͡p t͡s

DIACRITICS Diacritics may be placed above a symbol with a descender, e.g. ŋ̊

◌̥	Voiceless	n̥ d̥	◌̤	Breathy voiced	b̤ a̤	◌̪	Dental	t̪ d̪
◌̬	Voiced	s̬ t̬	◌̰	Creaky voiced	b̰ a̰	◌̺	Apical	t̺ d̺
◌ʰ	Aspirated	tʰ dʰ	◌̼	Linguolabial	t̼ d̼	◌̻	Laminal	t̻ d̻
◌̹	More rounded	ɔ̹	◌ʷ	Labialized	tʷ dʷ	◌̃	Nasalized	ẽ
◌̜	Less rounded	ɔ̜	◌ʲ	Palatalized	tʲ dʲ	◌ⁿ	Nasal release	dⁿ
◌̟	Advanced	u̟	◌ˠ	Velarized	tˠ dˠ	◌ˡ	Lateral release	dˡ
◌̠	Retracted	i̠	◌ˤ	Pharyngealized	tˤ dˤ	◌̚	No audible release	d̚
◌̈	Centralized	ë	~	Velarized or pharyngealized	ɫ			
◌̽	Mid-centralized	e̽	◌̝	Raised	e̝	(ɹ̝ = voiced alveolar fricative)		
◌̩	Syllabic	ɹ̩	◌̞	Lowered	e̞	(β̞ = voiced bilabial approximant)		
◌̯	Non-syllabic	e̯	◌̘	Advanced Tongue Root	e̘			
◌˞	Rhoticity	ɚ	◌̙	Retracted Tongue Root	e̙			

Apêndice 2 - Tabela ASC II e Descrição

ASCII é a sigla para Código Padrão Américano para Troca de Informações (*American Standard Code for Information Interchange*). Os computadores só conseguem reconhecer números, então o Código ASCII é uma representação numérica de um caracter como 'a' ou '@' ou uma ação de qualquer tipo. O Código ASCII foi desenvolvido a muito tempo, e hoje em dia, os caracteres 'não-imprimíveis' raramente são usados. Abaixo, está a tabela de caracteres ASCII, e inclui a descrição dos primeiros 32 caracteres 'não-imprimíveis'.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Fonte : http://www.lookuptables.com

Código ASC II Extendido

À medida que menos pessoas precisam de computadores para entender outros e caracteres 'não-imprimíveis', o Código ASCII se tornou restritivo. E com tanta tecnologia, demora um pouco para entender esses caracteres extras, e também há uma pequena variação 'extendida' destes. Os mais conhecidos estão abaixo.

128	Ç	144	É	160	á	176	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	│	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	ÿ	168	¿	184	╕	201	╔	217	┘	233	Θ	249	∙
137	ë	153	Ö	169	⌐	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	█	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	⁻
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	φ	253	²
141	ì	158	₧	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

Fonte : http://www.lookuptables.com

Apêndice 3 – Passos Realizados para Conversão Grafema-Fone

A - Alinhamento da base (para experimentos usando dicionários no formato não alinhado)

Obter a matriz inicial A, a partir do dicionário não alinhado (nonaligned.dic), usando o método heurítico descrito em [35]

java scripts.graph2phon.EstimateInitialAssociationMatrix nonaligned.dicE nonaligned.dic.matrix.0

Script que implementa a Programação Dinâmica baseada em [35]. Atualiza a matriz A e através da PD resulta em diversos dicionários alinhadosaté a convergência.

java scripts.graph2phon.IterativelyAlignDictionary nonaligned.dic temp-aligned.dic $max_itera

B - Conversão do último dicionário alinhado (temp-aligned.dic) para o nosso formato. Passo que inclui o cabeçalho com informações sobre os grafemas e fonemas presentes no dicionário

java scripts.conversion.ConvertBeepToDictionary temp-aligned.dic > dic_name.dic

C – Criação dos arquivos de treino (em torno de 60% do total) e teste (em torno de 40% do total)

Train.dic

Test.dic

D - Criação dos Arquivos no formato WEKA (arff), usando expansão simétrica de contexto igual a 1, 3 e 5. Fica a critério do usuário o número de contextos a ser usado

java scripts.conversion.ConvertAlignedDictionaryToARFF Train.dic $contexto > Train.arff

java scripts.conversion.ConvertAlignedDictionaryToARFF Test.dic $contexto > Test.arff

E – Treino e teste dos dados usando métodos de aprendizado de máquina

#Naive Bayes (padrão)

weka.classifiers.bayes.NaiveBayes –t Train.arfff –T Test.arff -d naivebayes.model

#J48 (cvparameters)

java weka.classifiers.meta.CVParameterSelection -X 2 -S 1 -W weka.classifiers.trees.J48 –t train.arff –T test.arff -d j48.model -P "C 0.25 0.50 2" -P "M 2 6 3" -- -L -S -A

F - Obter estatísticas de Erro por palavra e fone do conjunto de teste

Transcrições originais do arquivo de teste: Test.dic

classificador: $classificador.model

Arquivo de treino no formato arff: train.arff

java scripts.graph2phon.GetErrorStatsForAlignedDic Test.dic $classificador.model train.arff false

Apêndice 4 – Mapeamento UFPAdic1.0 - SAMPA

Vogais	
Fonemas do Português Brasileiro	SAMPA
a	Oral = a Nasal = a~ Final de palavra = a Tônico e seguido de m, n, nh cana= a Semi-aberto = a
e	Oral = fechado e aberto E Nasal = e~ Final de Palavra = i
i	Oral = fechado i Nasal = i~ Semi-vogal = I
o	Oral = fechado o Aberto = O Nasal = o~ Final de Palavra = U
u	Oral = u Nasal = u~ Semi-vogal = U

Encontros Vocálicos	
Fonemas do Português Brasileiro	SAMPA
ai	aI
au	aU
ão	a~U
ei	eI
em	e~I~
eu	eU
ea (Oséas)	Ea

ia (tia)	Ia
oi	oI
oi (herói)	OI
ou	oU
ou (sol)	OU
ua (rua)	Ua

Consoantes	
Fonemas do Português Brasileiro	SAMPA
b	b
c	k (casa) s (cedo)
d	d (dedo) e dZ (seguido de i – dia)
f	F
g	g (gato) e Z (gelo)
j	Z
k	k
L	l e L (quando seguido de i como em litro)
m	m
n	n
p	P
q	K
r	R (rato) h (comer)e r (caro)
s	s (sapo) z (entre vogais: casa / mais a menina…) S (quando tiver o som de x como pronunciado o plura no Pará p. ex.: dois [dojs])
t	t (tala) e tS (diante de i como em tia)
v	v

x	S
z	z

Dígrafos	
Fonemas do Português Brasileiro	SAMPA
lh	L
nh	J
ch	S

Alguns Símbolos	
Fonemas do Português Brasileiro	SAMPA
Sinal de sílaba seguinte acentuada	,,
Sinal de prolongamento na pronúncia	:

Apêndice 5 – Passos da implementação do LVCSR para o PB, usando o *corpus* Spoltech

echo "Preparação do dicionário - Dicionário Spoltech"

Criado o dicionário de grande vocabulário (60K), a partir de regras de transcrição obtidas no processo de aprendizado de máquina.

java ufpa.curupira.scripts.graph2phon.GenerateNewPronunciations dicttotranscribe j48_5.model TrainUFPAdict5-corr.arff true > dict

echo "Codifiar os arquivos de audio para o formato MFCC feature vectors"

HCopy -A -T 1 -C config -S raw_mfc.scp >logs/hcopy.log

#"Criar arquivos mfc de treino e teste"

Dos 2540 arquivos com transcrição ortográfica

Foram usados 2051 arquivos para treino (train_mfc.scp) e 489 para teste (test_mfc.scp)

echo "Construindo MLF (MasterLabelFile) de treino e teste"

Transcrição ortográfica em nível de palavras dos arquivos de treino e teste

java ufpa.curupira.scripts.htk.CreateWordMLF trans_train.list

java ufpa.curupira.scripts.htk.CreateWordMLF trans_test.list

echo “Iniciar modelos HMM baseados em monophones0 (sem o sp)"

rm -f -r hmm0 hmm1 hmm2 hmm3 hmm4 hmm5 hmm6 hmm7

mkdir hmm0 hmm1 hmm2 hmm3 hmm4 hmm5 hmm6 hmm7

Converter o train_words.mlf (nível de palavras) em phone0.mlf (nível de fones)

HLEd -l '*' -d dictionary/dict -i phones0.mlf mkphones0.led words_train.mlf

monophones0 - lista de fones de acordo com a seqüência de modelos HMM dispostas nos arquivos de treino e teste (sem o short pause)

monophones1 - contém a mesma lista de fones em monophones0 com a adição do short pause (sp)

echo “criar o protótipo HMM”

java ufpa.curupira.scripts.htk.CreateHMMPrototype 5 1 39 MFCC_E_D_A proto > proto

echo “Flat-start spoltech”

criação do arquivo de definições do primeiro modelo HMM em spoltech/hmm1

HCompV -A -T 1 -C config -f 0.01 -m -S train_mfc.scp -M hmm0 proto >logs/hcompv_flat.log

echo “Criar arquivo master de definições e macros”

cp macros hmm0/

cat hmm0/vFloors >> hmm0/macros

perl scripts/CreateHMMDefs.pl hmm0/proto monophones0 > hmm0/hmmdefs

echo “Reestimar as HMMs iniciadas com o modo flat-strat (sem o sp)”

#1a invocacao de HERest não reestima transições nem usa pruning

HERest -T 1 -T 1 -d hmm0 -u mvw -I phones0.mlf -S train_mfc.scp -H hmm0/macros -H hmm0/hmmdefs -M hmm1 monophones0 > logs/hmm1.log

HERest -T 1 -d hmm1 -u tmvw -I phones0.mlf -t 250.0 300.0 1000.0 -S train_mfc.scp -H hmm1/macros -H hmm1/hmmdefs -M hmm2 monophones0 > logs/hmm2.log

HERest -T 1 -d hmm2 -u tmvw -I phones0.mlf -t 250.0 300.0 1000.0 -S train_mfc.scp -H hmm2/macros -H hmm2/hmmdefs -M hmm3 monophones0 > logs/hmm3.log

echo “Consertar o modelo de Silêncio e adicionar o short pause sp (HTKBook 3.2.2.)”

perl scripts/DuplicateSilence.pl hmm3/hmmdefs >hmm4/hmmdefs

cp hmm3/macros hmm4/macros

HHEd -A -T 1 -H hmm4/macros -H hmm4/hmmdefs -M hmm5 sil.hed monophones1 > logs/hhed_flat_sil.log

echo "Criando o Modelo de Linguagem e Dicionário com sp"

Modelo de Linguagem

HLStats -b bigram -o wlist_HLStats words.mlf

Construir a rede sobre o Modelo de Linguagem

HBuild -A -T 1 -n bigram wlist_correto wdnet

Adicionar sp no final das palavras transcritas no dicionário dict -> dictsp

cat dictionary/dict | sed 's/$/ sp/g' > dictsp

Converter o train_words.mlf (nível de palavras) em phones1.mlf (nível de fones incluindo o fone sp)

HLEd -l '*' -d dictsp -i phones1.mlf mkphones1.led words_train.mlf

echo “Reestimando HMMs agora com fone sp”

HERest -T 1 -d hmm4 -u mvw -I phones1.mlf -t 250.0 300.0 1000.0 -S train_mfc.scp -H hmm4/macros -H hmm4/hmmdefs -M hmm5 monophones1 > logs/hmm5.log

HERest -T 1 -d hmm5 -u mvw -I phones1.mlf -t 250.0 300.0 1000.0 -S train_mfc.scp -H hmm5/macros -H hmm5/hmmdefs -M hmm6 monophones1 > logs/hmm6.log

HERest -T 1 -d hmm6 -u mvw -I phones1.mlf -t 250.0 300.0 1000.0 -S train_mfc.scp -H hmm6/macros -H hmm6/hmmdefs -M hmm7 monophones1 > logs/hmm7.log

echo “Testando”

HVite -T 1 -H hmm7/macros -H hmm7/hmmdefs -S test_mfc.scp -i results_chadia_6e1g.mlf -w wdnet dictsp monophones1 > logs/hvite_chadia_6e1g.log

echo “Resultados”

HResults -I words_test.mlf listwords results_chadia_6e1g.mlf > hresults/hresults_chadia_6e1g.log

##

echo “Incrementando numero de Gaussianas de 2 a 16 Gau/mix e reestimando”

echo “ Mixup 1->2”

rm -r -f hmm8 hmm9 hmm10 hmm11

mkdir hmm8 hmm9 hmm10 hmm11

HHEd -H hmm7/macros -H hmm7/hmmdefs -M hmm8 mixup2.hed monophones1 >logs/hhed_mixup2.log

HERest -u tmvw -I phones1.mlf -t 250.0 300.0 1000.0 -S train_mfc.scp -H hmm8/macros -H hmm8/hmmdefs -M hmm9 monophones1 >logs/hmm9.log

HERest -u tmvw -I phones1.mlf -t 250.0 300.0 1000.0 -S train_mfc.scp -H hmm9/macros -H hmm9/hmmdefs -M hmm10 monophones1 >logs/hmm10.log

HERest -u tmvw -I phones1.mlf -t 250.0 300.0 1000.0 -S train_mfc.scp -H hmm10/macros -H hmm7/hmmdefs -M hmm11 monophones1 >logs/hmm11.log

echo “Testando-9e2g”

HVite -T 1 -H hmm11/macros -H hmm11/hmmdefs -S test_mfc.scp -i results_chadia__9e2g.mlf -w wdnet dictsp monophones1 > logs/hvite_chadia_9e2g.log

echo “Resultado-9e2g”

HResults -I words_test.mlf listwords results_chadia__9e2g.mlf > hresults/hresults_chadia_9e2g.log

##

echo “Mixup 2->3”

rm -r -f hmm12 hmm13 hmm14 hmm15

mkdir hmm12 hmm13 hmm14 hmm15

HHEd -H hmm11/macros -H hmm11/hmmdefs -M hmm12 mixup3.hed monophones1 >logs/hhed_mixup3.log

HERest -u tmvw -I phones1.mlf -t 250.0 300.0 1000.0 -S train_mfc.scp -H hmm12/macros -H hmm12/hmmdefs -M hmm13 monophones1 >logs/hmm13.log

HERest -u tmvw -I phones1.mlf -t 250.0 300.0 1000.0 -S train_mfc.scp -H hmm13/macros -H hmm13/hmmdefs -M hmm14 monophones1 >logs/hmm14.log

HERest -u tmvw -I phones1.mlf -t 250.0 300.0 1000.0 -S train_mfc.scp -H hmm14/macros -H hmm14/hmmdefs -M hmm15 monophones1 >logs/hmm15.log

echo “Testando-12e3g”

HVite -T 1 -H hmm15/macros -H hmm15/hmmdefs -S test_mfc.scp -i results_chadia_12e3g.mlf -w wdnet dictsp monophones1 > logs/hvite_chadia_12e3g.log

echo “Resultado-12e3g”

HResults -I words_test.mlf listwords results_chadia_12e3g.mlf > hresults/hresults_chadia_12e3g.log

##

O mesmo procedimento acima e feito para o incremento das seguintes gaussians:

“Mixup 3->5”

“Mixup 5->8”

“Mixup 8->12”

“Mixup 12->16”