UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA DA COMPUTAÇÃO

SISTEMAS DE RECONHECIMENTO DE VOZ PARA O PORTUGUÊS BRASILEIRO UTILIZANDO OS CORPORA SPOLTECH E OGI-22

Carlos Patrick Alves da Silva

BELÉM - PARÁ

2008

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA DA COMPUTAÇÃO

Carlos Patrick Alves da Silva

SISTEMAS DE RECONHECIMENTO DE VOZ PARA O PORTUGUÊS BRASILEIRO UTILIZANDO OS CORPORA SPOLTECH E OGI-22

Trabalho de Conclusão de Curso apresentado para obtenção do grau de Engenheiro em Engenharia da Computação, do Instituto de Tecnologia, da Faculdade de Engenharia da Computação.

BELÉM - PARÁ

2008

A adversidade leva alguns a serem vencidos e outros a baterem recordes. William Arthur Ward

SISTEMAS DE RECONHECIMENTO DE VOZ PARA O PORTUGUÊS BRASILEIRO UTILIZANDO OS CORPORA SPOLTECH E OGI-22

Este trabalho foi julgado adequado em _____ para a obtenção do Grau de Engenheiro da Computação, aprovado em sua forma pela banca examinadora que atribuiu o conceito _____.

_____

Prof. Aldebaro Barreto da Rocha Klautau Jr.

ORIENTADOR

_____

M.Sc. Nelson Cruz Sampaio Neto

COORIENTADOR

_____

Profa. Valquiria Gusmão Macedo

MEMBRO DA BANCA EXAMINADORA

_____

Eng. Tales Cesar de Oliveira Imbiriba

MEMBRO DA BANCA EXAMINADORA

_____

Prof. Gervásio Protásio Cavalcante

DIRETOR DA FACULDADE DE ENGENHARIA DA COMPUTAÇÃO

Dedico este TCC à meus pais que sempre me apoiaram e torceram pelo meu sucesso. À minha irmã sempre presente.

Agradecimentos

Agradeço a DEUS por ter me dado a vida e ter permitido que eu chegasse até aqui. Minha mãe Josemere, agradeço de coração pelo carinho e preocupação constante. Ao meu pai Gilberto, por ter acreditado e confiado em mim. Agradeço à minha irmã Camilla pela paciência e apoio e ao meu pequeno irmão Matheus pelos milhares de favores feitos.

Agradeço ao meu amigo e orientador Aldebaro Klautau que acreditou na minha capacidade e me deu a oportunidade de trabalhar no Laboratório de Processamento de Sinais (LaPS), além da paciência e encorajamento constante. Aos meus amigos Carol, Igor e José que me acompanharam durante todo o curso e espero que continuem a me acompanhar. Aos grandes amigos que me ajudaram no meu trabalho no LaPS, Tales e Nelson pela paciência em me explicar várias coisas, várias vezes. A minha amiga Yomara pela ajuda e aos demais amigos do LaPS pela convivência.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

A tecnologia de processamento de voz tem avançado de maneira considerável nos últimos anos. Sistemas de reconhecimento automático de voz (RAV) têm sido desenvolvidos para vários idiomas, tais como o inglês. Neste cenário, este trabalho tem como objetivo, junto com a iniciativa *FalaBrasil*, o desenvolvimento de um sistema de reconhecimento automático de voz (RAV) para o português brasileiro com suporte a grandes vocabulários. Basicamente foram desenvolvidos: modelos acústicos, modelos de linguagem e dicionário fonético. O sistema se utiliza de cadeias ocultas de Markov contínuas (HMMs) para modelamento acústico e modelos de linguagem baseados em *n*-gramas. Para construção do sistema utilizou-se dos *corpora* Spoltech e OGI-22. Para construção do modelo de linguagem utilizou-se do *corpus* de texto CETEN-Folha. O sistema desenvolvido utiliza softwares disponíveis livremente na Internet, tais como HTK, ATK e SRILM. Vários testes foram realizado com intuito de se analisar como os principais parâmetros de um sistema RAV, tais como perplexidade, tamanho do vocabulário, modelos acústicos e de linguagem impactam no desempenho do decodificador. Todos os recursos desenvolvidos estão disponibilizados na Internet, de forma a disseminar o uso dos mesmos junto à comunidade acadêmica e permitir que outros pesquisadores possam reproduzir e aprimorar os resultados.

Abstract

Speech tecnology has advanced considerably in the last years. Automatic speech recognition systems (ASR) has been developed for several languages, such as English. In this scenario, this work intend to develop, in cooperation with the *falabrasil* initiative, a large vocabulary speech recognition system for Brazilian portuguese. Basically it was developed: acoustic models, language models and phonetic dictionary. The system uses hidden Markov models (HMM) for acoustic modeling and language models based on *n*-grams. The system uses two public corpora: Spoltech and OGI-22. Language models were built with the CETENFolha corpus. Some public and free softwares to build ASR systems like HTK, ATK and SRILM were used. The influence of perplexity, vocabulary size, language and acoustic models over the decoder performance was analysed performing several tests. All resources are made available on the Internet in order to spread the resources in the academic community and allow other researchers replicate and improve the results.

Capítulo 1

Introdução

Desde o surgimento dos computadores sonha-se com a época em que humanos e máquinas terão um nível de interação visto apenas em filmes de ficção científica: robôs domésticos recebendo ordens, indústrias totalmente à base de robôs inteligentes, etc. Uma das prerrogativas para que isso aconteça é o total entedimento da fala humana por parte dos computadores. Capacidades como: falar, ouvir, ler, escrever, além do reconhecimento de pessoas pela voz, devem ser estabelecidas. Nesse cenário surge o reconhecimento automático de voz (RAV) com o objetivo de se desenvolver um sistema capaz de modelar a fala humana, permitindo que o computador de certo modo "entenda"o que está sendo dito. A tecnologia de processamento de voz já possui várias aplicações tais como celulares com discagem por comandos de voz, as unidades de resposta audível, utilizadas por empresas de *call center* (atendimento) de forma que sejam faladas as opções durante o atendimento eletrônico, tutores inteligentes, sistemas de ditado como o *ViaVoice*[1] da IBM e o *Dragon NaturallySpeaking* [2] da Nuance. Os portadores de necessidades especais também são beneficiados com tais sistemas, usuários que não podem usar as mãos e deficientes visuais fazem uso desses sistemas de forma a se expressarem ditando textos e realizando o controle sobre várias das funções do computador utilizando a voz.

Universidades e indústrias vêm tentando resolver problemas práticos da área, de forma a possibilitar o reconhecimento da fala natural. Existem vários grupos de pesquisa atuando no mundo, tanto nos principais institutos de pesquisas. A Microsoft é um exemplo de empresa que já possui um reconhecedor privado para o inglês e outras línguas. É observado o grande avanço alcançado por vários países como os de língua inglesa, que já possuem sistemas RAV com bom

[1] http://www-306.ibm.com/software/pervasive/embedded_viavoice/, Visto em Maio, 2008

[2] http://www.voicerecognition.com.au/dragon_preferred.htm, Visto em Maio, 2008

desempenho em domínio público[3]. No Brasil ainda não existem sistemas RAV com suporte a grandes vocabulários (mais de 30 mil palavras) de domínio público, entretanto sistemas RAV para o português brasileiro (PB) já foram alvo de vários trabalhos [1–4]. Sistemas de reconhecimento de voz com suporte a grandes vocabulários são conhecidos na literatura como LVCSR, do inglês *Large Vocabulary Continuos Speech Recognition*. Dentre as dificuldades encontradas na criação de sistemas LVCSR temos a disponibilidade de um corpus de voz digitalizada e transcrita grande o suficiente para treinamento do sistema, recursos como bases de textos de tamanho elevado e um dicionário fonético de amplo vocabulário.

Neste contexto, este trabalho se dispõe, junto com a iniciativa *FalaBrasil* [5] criada pelo laboratório de processamento de sinais (LaPS) da UFPA, a desenvolver e disponibilizar recursos para o PB, de forma a se criar um sistema de referência e permitir que outros grupos de pesquisa se utilizem dos recursos criados. Dentre os recursos a serem desenvolvidos tem-se o modelo acústico, modelo de linguagem, dicionário fonético e um sistema LVCSR para o PB com suporte a grandes vocabulários. Para o desenvolvimento dos recursos utilizou-se de vários *toolkits* de domínio público tais como o HTK (linguagem C) que consiste em uma ferramenta para manipulação de modelos ocultos de Markov (HMM), o SRILM utilizado para construção de modelos de linguagem e o *software* WEKA usado na construção do dicionário fonético. Para treino dos modelos acústicos fez-se uso do *corpus* Spoltech, que consiste de um conjunto de gravações realizadas em diferentes regiões do Brasil. O mesmo foi desenvolvido pela UFRGS, USC e OGI. Também utilizou-se o *corpus* OGI-22 com gravações telefônicas criado pelo CSLU e OHSU. Para criação do modelo de linguagem dispõe-se do *corpus* CETENFolha, que consiste em um agrupamento de textos retirados do jornal Folha de São Paulo compilado pelo NILC de São Carlos.

O presente trabalho está organizado da seguinte forma: o Capítulo 2 aborda os principais fundamentos teóricos sobre sistemas RAV, dificuldades e problemas encontrados e os principais blocos que compõe o sistema. No Capítulo 3 são listados os principais recursos utilizados na criação do sistema proposto. O Capítulo 4 explica como foi construído o sistema, mostrando quais ferramentas foram utilizadas e os resultados obtidos. No Capítulo 5 são expostas as conclusões do trabalho e algumas sugestões para trabalhos futuros.

[3]"http://www.voxforge.org", visto em maio de 2008

Capítulo 2

Fundamentos de Sistemas RAV

O reconhecimento automático de voz (ASR, *Automatic Speech Recognition*) consiste, de uma forma geral, no processo onde o sinal de voz (analógico) é convertido em sua representação textual. O avanço da tecnologia tem permitido utilização de várias técnicas antes não utilizadas devido à limitação computacional. A evolução de sistemas de reconhecimento de palavras isoladas para sistemas de reconhecimento de fala contínua é um grande progresso na área de voz. Novas aplicações vem sendo desenvolvidas a cada ano por vários grupos de pesquisa no mundo.

Neste capítulo é descrito um breve histórico da tecnologia de reconhecimento de voz, as principais características de um sistema RAV, problemas encontrados no desenvolvimento de tais sistemas e o que se espera no futuro da tecnologia RAV. Além disso, cada um dos blocos que compõe um sistema LVCSR é descrito procurando mostrar a influência dos mesmos na tarefa de reconhecimento de fala contínua.

2.1 Histórico da Tecnologia RAV

De acordo com [6], sistemas de reconhecimento automático de voz tem sido estudados desde os anos 50 nos Laboratórios Bell, onde foi desenvolvido o primeiro reconhecedor de dígitos isolados com suporte a apenas um locutor. Na mesma época, foi introduzido o conceito de redes neurais, mas devido a muitos problemas práticos a idéia não foi seguida. Nos anos 70, os russos iniciaram estudos sobre reconhecimento de padrões, no entanto muitas das idéias eram de difícil implementação devido a limitação tecnológica da época. A técnica predominante na época era o *Dynamic Time Warping* (DTW) [7], que consiste em um algoritmo que mede

a similaridade entre duas seqüências que variam no tempo, como a voz. Com o passar dos anos a pesquisa foi evoluindo e muitos problemas tecnológicos foram sendo superados, além da globalização e popularização dos computadores que levou a um aumento no número de pesquisas na área.

Inicialmente os sistemas de reconhecimento de voz tentavam aplicar um conjunto de regras gramaticais e sintáticas à fala [8]. Caso as palavras ditas caíssem dentro de um certo conjunto de regras, o programa poderia determinar quais eram aquelas palavras, para isso era preciso falar cada palavra separadamente (sistemas de voz discreta), com uma pequena pausa entre elas. Porém, devido as características como sotaques, dialetos e regionalismos e as inúmeras exceções da língua os sistemas baseados em regras não tiveram muito sucesso.

Nos anos 80, vários pesquisadores iniciaram pesquisas para reconhecimento de palavras conectadas utilizando métodos de modelos estatísticos, sendo o maior destaque para os modelos ocultos de Markov usados para modelar séries temporais, como por exemplo a voz. Além disso um estudo mais profundo mostrou a possibilidade de aplicação de redes neurais na classificação de sinais.

2.2 Características de Sistemas RAV

Muitos são os fatores que influenciam e dificultam o desempenho de reconhecedores, desde ruídos causados pelo ambiente ao sotaque do locutor. No projeto de reconhecedores vários parâmetros devem ser analisados, tais como o tamanho do vocabulário e o estilo de fala (contínua ou com intervalos) de forma a se obter um melhor desempenho. Este cápitulo tem por objetivo mostrar uma visão geral das principais caracteríticas dos sistemas LVCSR.

2.2.1 Dependência de Locutor

Sistemas de reconhecimento podem ser classificados como dependentes ou independentes de locutor. No primeiro caso o sistema é treinado para um locutor específico, sendo assim, o mesmo é apto a reconhecer com uma boa taxa de acerto apenas o locutor para o qual foi treinado. Sistemas independentes de locutor são capazes de reconhecer a fala de qualquer locutor, mesmo aquele que não participou do treino do sistema. Para se obter tal reconhecedor deve haver uma grande variedade de locutores no *corpus* de treino.

2.2.2 Tipo de Fala

Sistemas RAV podem ser construídos para reconhecer palavras isoladas ou palavras conectadas (fala contínua). Reconhecedores de palavras isoladas tem a tendência a apresentar um resultado muito superior aos de fala contínua. Isso acontece devido ao intervalo existente entre as palavras, que é utilizado pelo reconhecedor como refêrencia de início e fim de uma palavra. Um exemplo clássico de reconhecedores de palavras isoladas são os reconhecedores de dígitos, que alcançam taxa de menos de 2% de erro para dígitos de 0 à 10.

O reconhecimento de fala contínua é uma tarefa de difícil implementação, visto que ocorrem poucas pausas durante a fala espôntanea, não se tendo informação de onde começam e terminam determinadas palavras, muitas palavras são mascaradas, encurtadas e as vezes não pronunciadas. Os efeitos co-articulatórios estão fortemente presentes nesses casos, tornando ainda mais difícil a tarefa do reconhecedor em casos como "*ele vai morrer em dois dias*" que muitas vezes é dito como "*ele vai morrerem dois dias*".

2.2.3 Tamanho do Vocabulário

Um fator crucial na precisão de um sistema RAV é o número de palavras que o mesmo é apto a reconhecer. Quanto maior o vocabulário, maiores são as chances de equívocos por parte do decodificador que fará o reconhecimento. Em sistemas com grandes vocabulários existem muitas palavras ambíguas, ou seja, possuem realizações sonoras iguais ou semelhantes. Um maior vocabulário também gera um aumento no espaço de busca, que influencia no tempo de reconhecimento. Sistemas com vocabulários de tamanho reduzido são menos sucetíveis a erros e apresentam ótimos resultados.

2.2.4 Tamanho do Corpus para Treino

Como será mostrado adiante o sistema RAV criado é baseado em modelos estatísticos, com etapas de treino e teste. Sendo assim, o tamanho do *corpus* utilizado para treino e teste é um fator de extrema importância para se obter resultados confiáveis. Quanto mais dados houver para treino dos modelos acústicos e de línguagem, mais bem adaptados os mesmos estarão à língua em questão. Infelizmente os *corpora* existentes para o PB são considerados de pequeno porte (poucas horas). Sistemas de grande porte necessitam de centenas de horas para treinar seus modelos acústicos e textos de milhões de linhas para os modelos de linguagem. Apesar desse

déficit de dados para o PB, o sistema desenvolvido neste trabalho apresenta bons resultados e boa confiabilidade.

Existe uma grande variedade de opções e alternativas disponíveis na especificação de sistemas de reconhecimento de voz. Para o sistema proposto neste trabalho, tem-se como objetivo a construção de um reconhecedor de fala contínua com suporte a 60 mil palavras e independente de locutor.

2.3 Dificuldades na Construção de Sistemas RAV

Apesar do grande avanço da tecnologia de processamento de voz, o entendimento completo da fala humana por parte do computador ainda é uma tarefa difícil e complexa de se realizar. Existem muitos problemas que afetam na precisão do reconhecedor, alguns vem sendo reduzidos e eliminados com pesquisas na área e com o surgimento de novas tecnologias. Dentre as principais dificuldades destacam-se:

- Uma mesma palavra pode ser pronunciada de diferentes maneiras dependendo do tipo de locutor e da região, por exemplo a palavra “tia” dependendo do sotaque pode ser pronunciada como ”tia” ou “tchia”.

- Alguns locutores tendem a falar muito rápido, sendo assim, muitas sílabas ou vogais podem ser “engolidas” ou mal pronunciadas durante a fala.

- Em línguas como o português, que possui um vocabulário extenso, existem muitas palavras ambíguas, ou seja, que possuem a mesma pronúncia, essas palavras são conhecidas como homófonas. Vários são os exemplos deste tipo de palavra: “sessão/cessão”, “mais/mas”, “consertar/concertar”, etc. É praticamente impossível para o decodificador diferenciá-las apenas com informações acústicas, esse problema é resolvido com a ajuda do modelo de linguagem.

- Dificuldade na segmentação da fala: no reconhecimento da fala natural existe grande dificuldade por parte do decidificador em se determinar as fronteiras entre as palavras, principalmente durante conversas onde ocorrem poucas pausas.

- Variações nas características da fala como ritmo, timbre e intensidade causam grandes problemas durante o reconhecimento.

- Baixa relação sinal-ruído (SNR): Durante o reconhecimento, quanto mais "limpo" for o sinal de voz mais fácil é para o decodificador diferenciar as palavras. É difícil se conseguir arquivos de boa qualidade (com pouco ruído) para treino dos modelos acústicos. Características como o ruído ambiente (vozes de outros locutores, sons de equipamentos, etc) são difíceis de se contornar, principalmente em ambientes abertos[1] onde o ruído se torna inevitável.

Muitos problemas que surgem ao se usar um reconhecor de voz foram mostrados em público em uma demonstração recente do Windows Vista. Embora o sistema tenha tido um desempenho perfeito ao abrir programas e acessar documentos, a mesma precisão não apareceu na hora de transcrever textos. Os problemas provavelmente vieram do ruído de fundo e do eco presentes no grande auditório onde ocorreu a demonstração.

2.4 O Futuro da Tecnologia RAV

Em 1997 em um Workshop do IEEE na Inglaterra, vários pesquisadores da área fizeram previsões sobre o futuro na tecnologia de processamento de voz [9]. Dentre os eventos sugeridos temos:

1. 2010 - A maioria dos telefones passa a aceitar, além de digitos, a voz como entrada.
2. 2016 - Sistemas RAV são utilizados em ambientes domésticos (sistemas de gerenciamento e controle da casa).
3. 2057 - Telefones com sistemas tradutores permitem que pessoas de idiomas diferentes conversem normalmente.
4. 2060 - Interações com computador feita por gestos e comunicação natural em duplo sentido.

O avanço na área é notável, anualmente centenas de papers são publicados e soluções encontradas. Futuramente, é possível que o termo "reconhecimento de voz" se torne "compreensão de voz". Os modelos estatísticos que permitem que computadores reconheçam o que é dito por um usuário também podem vir a permitir que eles entendam o significado das palavras.

[1] Vários arquivos do *corpus* Spoltech foram gravados em ambientes ruidosos como feiras, universidades, etc.

Embora isso seja um gigantesco passo em termos de potência de computação e sofisticação dos programas, alguns pesquisadores defendem que o desenvolvimento do reconhecimento de voz oferece o caminho mais direto entre os computadores atuais e a inteligência artificial [10].

2.5 Blocos Básicos de um Sistema RAV

Os sistema de reconhecimento automático de voz atuais são em sua maioria baseados em reconhecimento estatístico de padrões [11], onde após o treinamento dos modelos, o sistema realiza a busca pela sequência de palavras mais provável, ou seja, que melhor representa o sinal de entrada. Para a realização da busca é necessário um conhecimento a priori que é representado pelos modelo acústicos e modelo de linguagem. Basicamente, faz-se uso da regra de Bayes:

$$\hat{W} = \arg\max_{W} \frac{P(O|W)P(W)}{P(O)} = \arg\max_{W} P(O|W)P(W) \tag{2.1}$$

onde se busca pela sequência de palavras W que maximiza a probabilidade condicional $\hat{W}$. Sendo O a matriz de parâmetros que reprenta o sinal acústico, tem-se $P(O|W)$ como sendo a probalilidade de se observar a matriz O dado a sequência de palavras W, valor esse fornecido pelo modelo acústico, e $P(W)$ como sendo a probabilidade da sequência de palavras W obtido através do modelo de linguagem. Como $P(O)$ não depende de W o mesmo pode ser descartado. O sistema construido foi baseado em cadeias escondidas de Markov (HMMs, *Hidden Markov Models*) que serão detalhadas adiante.

Um sistema RAV é composto por vários blocos como mostrado na Figura. 2.1 , dentre os quais temos o Front-End que é o bloco inicial responsável pela extração de parâmetros (*features*) do sinal de voz, o modelo acústico (MA) que busca modelar, a partir das *features*, o sinal acústico, o modelo de linguagem (ML) tenta, a partir de textos da língua, obter as possíveis sequências de palavras a serem reconhecidas e o decodificador que ,junto com os blocos citados, realiza o processo de transcrição da do sinal de voz. Esta seção tem por objetivo mostrar a estrutura e o funcionamento de cada um destes blocos.

2.5.1 Front-End

A primeira fase no processamento da voz é a conversão analógico-digital que traduz a onda analógica em dados digitais. Além disso durante a conversão o sistema filtra o som dig-

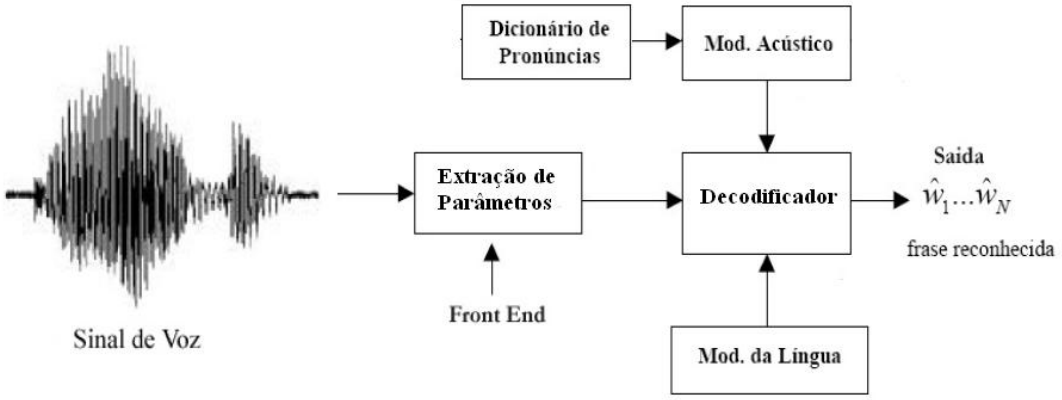

Figura 2.1: Principais blocos de um sistema de reconhecimento de voz.

italizado de forma a remover ruídos indesejados. Devido a natureza aleatória do sinal de voz, utilizar os mesmos na sua forma original não é uma prática recomendável. Sendo assim, é necessário encontrar uma outra forma de se representar o sinal acústico. Para isso faz-se uso da parametrização do sinal de voz, que nada mais é do que um processo de filtragem. Vários estudos foram realizados observando o aparelho auditivo e fonador do ser humano, como resultados tem-se diversos tipos paramétricos desenvolvidos: MFCC, PLP, FILTERBANK, LFCC. No trabalho proposto foi utilizado o MFCC, do inglês *Mel-frequency Cepstrum Coefficients*. O MFCC foi proposto em 1980 por Davis e Merlmestein [12] e é bastante utilizado em tarefas de reconhecimento de voz [13] [4] [3].

A extração de parâmetros normalmente é uma transformação não-inversível, ou seja, não é possível recuperar o sinal original devido a perda de informação. Essa perda de informação torna-se necessária devido a enorme complexidade computacional que seria necessária por parte dos blocos seguintes para processar as informações por completo. Portanto busca-se extrair as informações mais relevantes do sinal acústico, como informações que possam ser utilizadas para indentificar diferenças fonéticas, além disso a armazenagem compacta fornecida pelos parametros MFCC's torna-se um fator importante quando se trabalho com grande quantidade de arquivos.

Como processo inicial tem-se a segmentação do sinal de voz em segmentos curtos (janelas) de 20 a 25 milisegundos, com o deslocamento da janela de análise de 10 milisegundos, buscando uma sobreposição de 50% entre os *frames*. Esse processo é conhecido na área de processamento de sinais como janelamento [11]. O front-end opera em cada um desse *frames*,

convertendo-os em vetores de parâmetros MFCC's. O processo de janelamento é descrito na Figura 2.2.

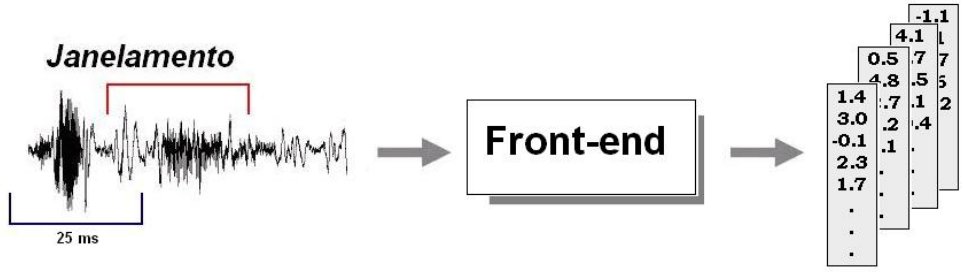

Figura 2.2: Processo de segmentação do sinal de voz e extração de parâmetros.

Após o janelamento, em cada *frame* é aplicada uma FFT (*Fast Fourier Transform*), obtendo assim seu espectro, que é passado por um conjunto de filtros triangulares na escala Mel, tal escala é a mesma do aparelho auditivo humando. Como passo seguinte tem-se uma compressão logarítmica, seguida de uma DCT (*Discrete Cosine Transform*) que diminui a correlação entre os elementos do vetor de parâmetros, tal processo é mostrado na Figura 2.3. Como saída tem-se os vários coeficientes cepstrais, porém no caso de modelamento de sistemas RAV é comum utilizar somentes os 12 primeiros coeficientes junto com a energia do sinal. A modelagem acústica assume que os vetores acústicos estão descorrelacionados dos seus vetores vizinhos, porém essa suposição não é garantida visto que os orgãos do aparato vocal humano garantem que há continuidade entre sucessivas estimativas espectrais. De acordo com Davis e Merlmestein, adicionando-se as derivadas de primeira e segunda ordem aos coeficientes e energia tem-se uma redução de tal problema. No final tem-se para cada frame uma saída com 39 parâmetros.

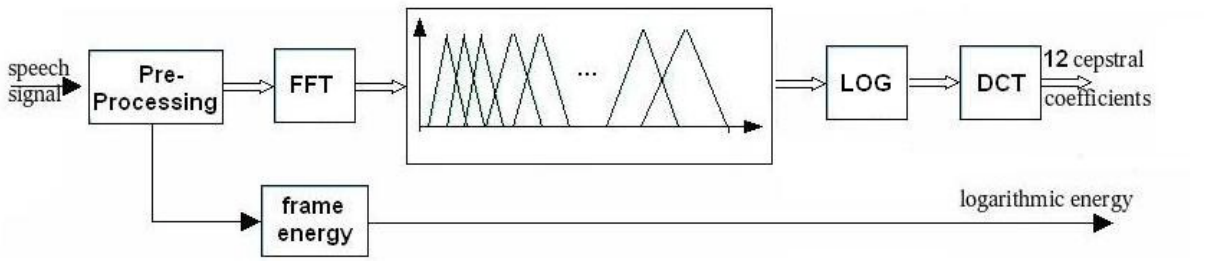

Figura 2.3: Processo de cálculo dos MFCCs.

2.5.2 Modelo Acústico

O modelo acústico busca através de *features* extraídas da voz criar um modelo matemático que represente o sinal original, de forma que dado segmentos desconhecidos os mesmos possam ser mapeados de forma correta para palavras no caso de reconhecimento de palavras isoladas, ou para fonemas no caso de reconhecimento de fala contínua. O trabalho proposto busca modelar sistemas contínuos, sendo assim são criados modelos para cada fonema[2] da língua. O conjunto de treino deve consistir de amostras contínuas, porém em geral, as fronteiras que dividem os fonemas não são conhecidas o que dificulta o processo de estimação dos modelos, porém a ferramenta HTK dispõe de ferramentas capazes de superar tal problema. No estado da arte modelos acústicos são representados por cadeias de Markov conectadas em sequência.

2.5.2.1 Cadeias Escondidas de Markov

Uma cadeia escondida de Markov [14] consiste em uma máquina de estados finita que modifica seu estado a cada unidade de tempo, mais especificamente, consiste de um modelo matemático formado por uma cadeia de estados conectados entre si, onde cada transição entre os estados possui uma probabilidade de ocorrência, além de cada estado está vinculado a um processo estocástico, que pode ser discreto ou contínuo, conhecido como processo de observação de saída. A observação de saída é a ocorrência do fenômeno sendo modelado, para o caso de sistemas LVCSR são os fonemas. A Figura 2.4 mostra o modelo básico de uma HMM com 3 estados emissores (estados 2,3 e 4) e 2 estados que não emitem observações. Estados não emissores são utilizados para concatenação de modelos. A evolução da cadeia de estados, para o caso de reconhecimento de voz, é da esquerda para direita (*letf-right*). Temos que a_{ij} é a probabilidade de transição do estado i para o estado j que ocorre a cada tempo t, onde nos casos em que i=j um estado sofre transição para ele mesmo, e $b_i(x_t)$ é a probabilidade da observação x no tempo t dado que o estado é i. Durante o processo não se tem infomação sobre a evolução da cadeia de estados(*hidden*).

Na construção de sistemas LVCSR utiliza-se modelos de Markov de 1^a ordem [14], onde cada fonema possui seu próprio modelo HMM, representado por λ. Na fase de treino os M modelos são estimados a partir de um conjunto de observações O. Na fase de testes é apresentada uma sequência de observações O', onde o algoritmo de reconhecimento busca pelo modelo $\hat{\lambda}$ dentre os modelos $\lambda_i(i = 1, 2, ..., M)$ o que possui a maior probabilidade de ter gerado a

[2] O fonema é a menor unidade sonora que descreve um som.

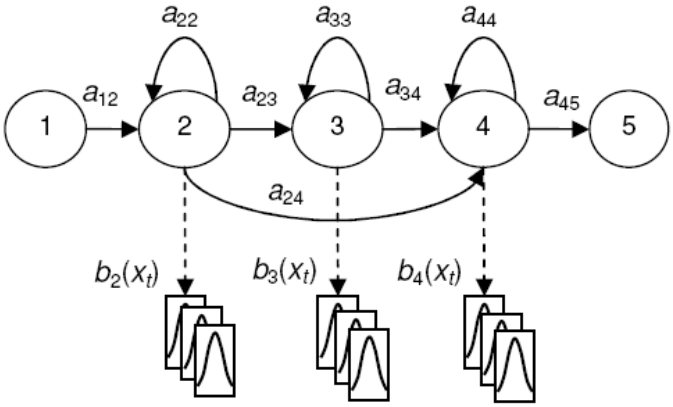

Figura 2.4: Modelo HMM.

sequência O', ou seja:

$$\hat{\lambda} = \arg\max_{\lambda} P(O'|\lambda) \tag{2.2}$$

Como mostrado na seção 2.5.1 o conjunto de observações consistem nos vetores MFCCs resultantes do bloco de extração de parâmetros. Abaixo são listados os principais parâmetros que constituem um modelo HMM.

- Número de estados N no modelo.
- Número de símbolos de observações distintas M por estado.
- Matriz de distribuição de probabilidades de transição entre os estados A, que armazena todos os possíveis valores de a_{ij}.
- Matriz de distribuição de probabilidades dos símbolos de observação no estado i, representada por B. A matriz B guarda todos os possíveis valores de $b_i(x_t)$.
- Matriz de distribuição do estado inicial π, que retorna a probabilidade do estado inicial ser o estado i.

A partir dos parâmetros mostrados um modelo completo de HMM pode ser representado por $\lambda = (A, B, \pi)$.

Na construção de modelos HMM encontramos basicamentes 3 problemas: $a)$ calcular de maneira eficiente $P(O|\lambda)$, que seria a probabilidade da sequência de observações O ter sido gerada pelo modelo λ, este problema consiste basicamente no cálculo da equação 2.2; $b)$ ajustar

os parâmetros A, B e π de forma a maximizar $P(O|\lambda)$ para cada modelo λ_i, o que equivale ao treinamento dos modelos HMM; $c)$ Encontrar (durante o reconhecimento) dentre as possíveis seqüências de estados que o sistema pode seguir, aquela com maior probabilidade de ter gerado a sequência O'.

Para resolução do problema (a) utiliza-se dos algoritmos *forward/backward* [14], que consistem em algoritmos recursivos que facilitam o cálculo de $P(O|\lambda)$. No caso do problema (b) que não possui solução analíca, faz-se uso de técnicas interativas de otimização, tendo como foco o algoritmo de reestimação de *Baum-Welch* [15]. E para o problema (c) a solução é encontrada com o uso do algoritmo de *Viterbi* [16], o qual faz uso de programação dinâmica na busca da melhor seqüência de estados, além disso o algoritmo é bastante importante na decodificação onde a determinação da seqüência de estados mais provável é a chave para o reconhecimento de uma seqüência de palavras desconhecidas.

2.5.2.2 Modelagem Contínua de Trifones

Até o momento temos considerado que o modelo acústico contém um modelo HMM por fonema, com isso tem-se a idéia de que um fonema pode ser seguido por qualquer outro, o que não é verdade já que os articuladores do trato vocal não se movem de uma posição para outra imediatamente na maioria das transições de fonemas. Neste sentido, na criação de sistemas que modelam a fala fluente busca-se um meio de modelar os efeitos contextuais causados pelas diferentes maneiras que alguns fonemas podem ser pronunciados. A solução encontrada é o uso de modelos HMM baseados em trifones. Modelos trifones consideram que um fonema em um contexto sofre influência dos fonemas vizinhos. Supondo a notação a-b+c, temos que b representa o fonema central ocorrendo após o fonema a e antes do fonema c.

Existem basicamente dois tipos de modelos trifones: os trifones intrapalavra (*word-internal triphones*) e os trifones entre-palavras (*cross-word triphones*). A diferença entre os mesmos é que no caso do *word-internal* as fronteiras entre as palavras não são consideradas, sendo assim, menos modelos são necessários, já no caso do *cross-word*, que considera as fronteiras entre palavras, a modelagem é mais precisa, porém o número de modelos trifones cresce muito, o que dificulta o trabalho do decodificador e gera uma necessidade de mais dados para treino. A Tabela 2.1 mostra as exemplos de transcrições utilizando ambos os tipos de trifones.

Tabela 2.1: Exemplos de transcrições utilizando trifones.

	arroz com bife
Monofones	sil a R o s k o∼ b i f i sil
Word-Internal	sil a+R a-R+o R-o+s o-s k+o∼ k-o∼ b+i b-i+f i-f+i f-i sil
Cross-Word	sil sil-a+R a-R+o R-o+s o-s+k s-k+o∼ k-o∼+b o∼-b+i b-i+f i-f+i f-i+sil sil

2.5.2.3 Vínculo de Estados de Modelos Trifones

Como observado na Tabela 2.1, o número de modelos cresce bastante. Por exemplo se utilizarmos 33 fonemas como modelos iniciais, a migração de monofones para trifones do tipo *cross-word* será para aproximadamente 41.650 modelos. Diante disso, nos deparamos com um grande problema na construção de sistemas LVCSR, insuficiência de dados para estimação dos modelos trifones, já que muitos desses trifones terão uma ou nenhuma ocorrência no *corpus*. Uma estratégia muito utilizada para redução desse problema é o compartilhamento (*tying*) de parâmetros entre os modelos. Muitos dentre os modelos trifones possuem características acústicas bastante semelhantes, sendo assim os mesmos podem compartilhar as distribuições de probabilidade em seus estados [17] [18] [19].

Existem basicamente duas técnicas para compartilhamento de estados, a primeira é conhecida como *data-driven*, na qual ocorre a clonagem dos monofones seguida pela conversão para trifones, por fim todos os estados centrais dos trifones devirados do mesmo monofone são vinculados conforme a Figura 2.5, isso é realizado levando em consideração que o contexto do trifone não afeta o estado central do modelo.

A segunda técnica consiste no uso de uma árvore de decisão fonética [20] [21], este método envolve a construção de uma árvore binária utilizando um procedimento de otimização seqüêncial *top-down* conforme [22], onde perguntas são anexadas a cada nó. As perguntas são relacionadas ao contexto fonético dos fonemas vizinhos do fonema examinado. As questões são do tipo "o fonema à esquerda é nasal ?", dependendo da resposta (sim/não) uma das possíveis direções é seguida. O objetivo da árvore e unir os estados que são acusticamente semelhantes, sendo assim, inicialmente todos os estados de um fonema são posicionados no nó raiz da árvore, de forma que ao percorrerem a árvore os estados vão sendo divididos, ao final do processo todos os estados no mesmo nó folha são agrupados. A Figura 2.6 mostra um exemplo desse processo.

As perguntas são escolhidas de forma a maximizar a verossimilhança entre os dados de

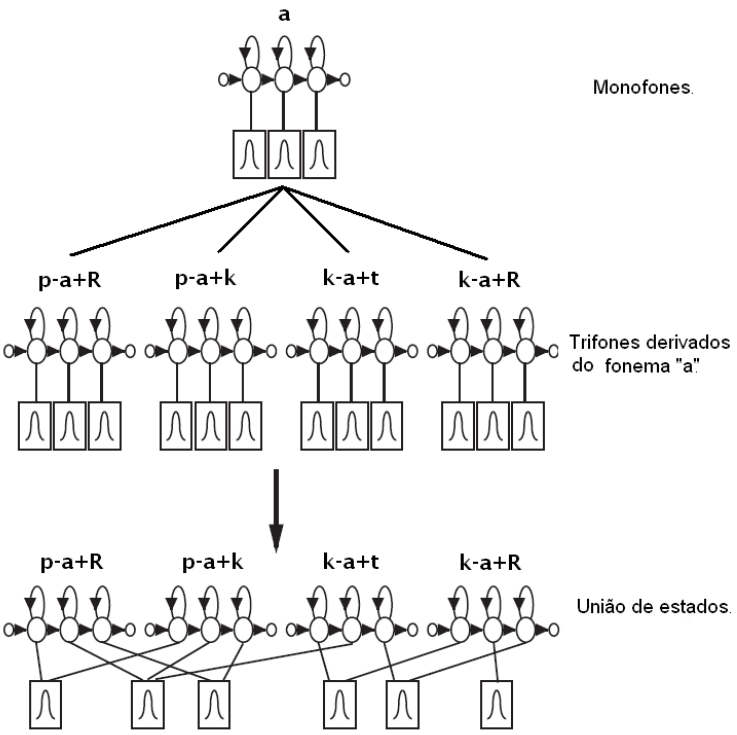

Figura 2.5: União de estados utilizando o método *data-driven*.

treino e o conjunto resultante da união dos estados, de forma que existam dados de treino suficientes para estimar de forma robusta os parâmetros das distribuições de probabilidade gaussiana, além disso trifones não observados nos dados de treino podem ser sintetizados realizando uma busca na árvore pelos respectivos nós terminais de seus estados. Os resultados obtidos utilizando árvore são em geral melhores do que com o método *data-driven* como visto em [21] [4] [2].

Uma outra estratégia utilizada para se obter uma melhor estimação das HMMs é a adoção de misturas de gaussianas no modelamento das observações de saída. Dentre os métodos estatísticos conhecidos as misturas tem-se mostrado mais robustas no modelamento de sinais variantes no tempo, como a voz. Vários trabalhos na área de voz obtiveram bom desempenho utilizando tal estratégia [23] [21] [2].

2.5.3 Modelo de Linguagem

Utilizar somente informações acústicas não é suficiente para o bom desempenho do sistema, já que assim, uma palavra poderia ser seguida por qualquer outra, o que dificulta muito o trabalho do decodificador, principalmente quando se lida com grandes vocabulários. Supondo

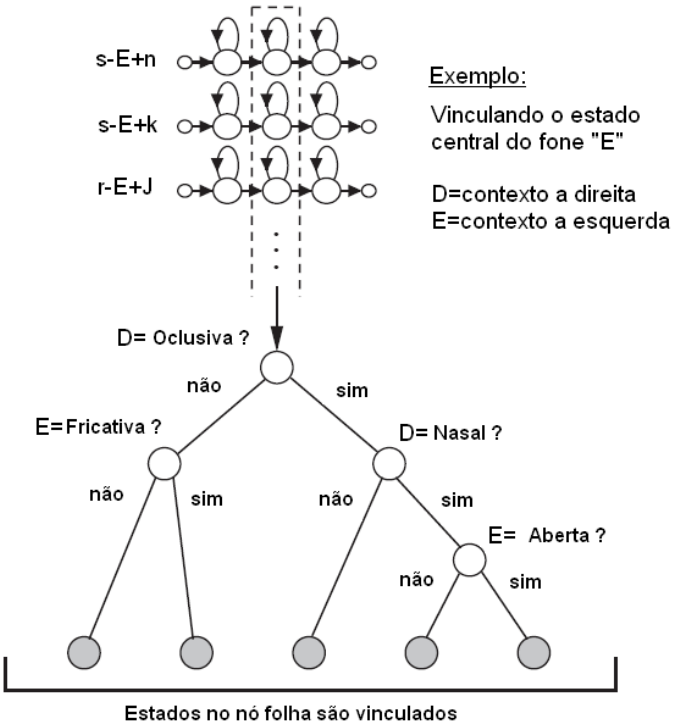

Figura 2.6: *Tied-States* utilizando árvore de decisão fonética.

um vocabulário de tamanho V, no reconhecimento de uma seqüência de N paravras existem V^N possibilidades. Nesse sentido, faz-se uso de modelos de linguagem (ML). Esses modelos buscam caracterizar a língua, tentando capturar suas regularidades e assim condicionar as combinações de palavras durante o reconhecimento, descartando frases agramaticais. O uso de modelos de linguagem tem levado a ganhos de desempenhos consideráveis [24], dentre as vantagens temos: a redução no espaço de busca, redução do tempo de reconhecimento e a resolução de ambigüidades acústicas [25].

O objetivo básico do modelo de linguagem é estimar de uma forma confiável a probabilidade de ocorrência de uma determinada seqüência de palavras $W = w_1, w_2, ...w_k$, onde k é o número de palavras na sentença. A probabilidade $P(W_1^k)$ é definida na equação 2.4

$$P(W_1^k) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_k|w_1, ..., w_k - 1) \quad (2.3)$$

$$P(W_1^k) = P(w_1)\prod_{i=2}^{k} P(w_i|w_1, ..., w_i - 1) \quad (2.4)$$

2.5.3.1 Modelos de Linguagem N-Grama

É observado o alto grau de complexidade no cálculo de $P(W_1^k)$, principalmente para grandes valores de k, porém uma solução para tal dificuldade é a utilização de n-gramas, onde assume-se que a distribuição de probabilidade para a palavra w_k depende somente das $n-1$ palavras anteriores a mesma, de forma que a equação 2.4 é reescrita da forma:

$$P(W_1^k) \approx P(w_1) \prod_{i=2}^{k} P(w_i | w_{i-n+1}^{i-1}) \tag{2.5}$$

Em línguas como o inglês e o português, onde a ordem das palavras é importante, os n-gramas são bastante eficientes, uma vez que, durante o treino, os mesmos capturam simultaneamente sintaxe e semântica da língua, não havendo necessidade de criação de regras, e nem de uma gramática formal. As distribuições de probabilidade são computadas diretamente de frases prontas. A estimação pode ser executada simplesmente contando o número de ocorrências dos n-gramas. Nesse sentido, para se obter uma boa estimação é necessário um conjunto de milhares de frases.

Em reconhecimento de voz normalmente faz-se uso de bigramas ($n = 2$), porém o uso de trigramas ($n = 3$) tem apresentado melhor desempenho, pois a maioria das palavras possui uma forte dependência das duas palavras precedentes. Sendo $C(w_i)$ o número de ocorrências das palavra w_i em um texto, o treino de bigramas é feito através da contagem das ocorrências dos bigramas seguida de uma normalização por $w_i - 1$, como visto na equação 2.6. Para validar os casos em que i=1 utiliza-se marcadores de inicio e fim de sentenças, tais como $< s >$ e $< /s >$ respectivamente.

$$P(w_i | w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})} \tag{2.6}$$

A mesma equação é facilmente estendida para trigramas com mostrado na equação 2.7

$$P(w_i | w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})} \tag{2.7}$$

É observado que para um vocabulário de tamanho V, existem V^3 possíveis trigramas, onde muitos desses trigramas nunca ou raramente aparecerão nos dados de treino, esse problema é conhecido como problema da *frequência zero*. Na fase de reconhecimento palavras com

probabilidade zero nunca serão reconhecidas pelo decodificador. Para resolução desse problema utiliza-se de técnicas de suavização (*smoothing*) [26] que buscam garantir que todas as palavras possuam probabilidades não nulas.

2.5.3.2 Perplexidade de um Modelo de Linguagem

Durante a criação de modelos de linguagem existem várias características que os diferenciam, dentre as quais destacam-se: o tamanho do vocabulário, ou seja o número de palavras presentes no modelo, e o tamanho e tipo de corpus utilizado para treino do mesmo. As frases utilizadas na estimação podem pertencer a domínios específicos, tais como: conversação, jornalismo, radiologia, etc. Como o modelo de linguagem busca facilitar o trabalho do decodificador, restringindo o número de possíveis palavras que podem seguir uma dada palavra, o mesmo influencia diretamente no seu desempenho, sendo assim, dois conceitos são introduzidos: *cross-entropy* e *perplexidade*.

Consideremos um vocabularios W e uma possível seqüência de palavras $w1, w2, ..., w_m$, a probabilidade de uma palavra w_i depende das palavras anteriores $w1, ..., w_{i-1}$. Para o cálculo da entropia cruzada (*cross-entropy*) tem-se a soma entre as probabilidades de todas as possíveis seqüências de palavras, porem considerando que tal soma é um processo ergódico, e adotando um alto valor para m, a entropia cruzada pode ser definida como:

$$H_P(T) = -\frac{1}{m} \log_2 P(w_1, w_2, ..., w_m) \tag{2.8}$$

A partir de 2.8 podemos definir a perplexidade (PP) de um modelo de linguagem como sendo:

$$PP = 2^{H_p(T)}. \tag{2.9}$$

Podemos imaginar o ML como sendo um grafo onde cada nó representa uma palavra, assim, a perplexidade seria o fator de ramificação médio em todos os pontos de decisão do grafo, ou seja, o número médio de palavras que podem seguir uma dada palavra. A perplexidade do conjunto de teste[3] avalia a capacidade de generalização do ML. É intuitivo perceber que quanto menor a perplexidade melhor tende a ser o desempenho do reconhecedor. Um exemplo de

[3] O conjunto de teste consiste de frases não vistas pelo modelo na fase de treino

perplexidade para sistemas LVCSR é o da língua inglesa, valores encontrados variam de 50 a 250, dependendo do domínio e do tamanho do vocabulário.

Como será visto a seguir, uma medida muito utilizada para avaliação de sistemas RAV é a taxa de erro por palavra *WER* (*Word Error Rate*). Outra forma de avaliar modelos de linguagem é fixando o modelo acústico e observando como diferentes valores de PP afetam a *WER*. De acordo com [27] existe uma forte correlação entre a PP e a *WER*, como indicado na expressão 2.10

$$\text{WER} \approx -12.37 + 6.48 \log_2(PP) = -12.37 + 6.48 H_p(T) \tag{2.10}$$

A tabela 2.2 mostra resultados obtidos para vários sistemas RAV com diferentes configurações de modelos de linguagem.

Tabela 2.2: Relação entre tamanho do vocabulário, perplexidade e WER de acordo com [27].

Corpus	**Tamanho do Vocabulário**	**Perplexidade**	**WER**
TI Digits	11	11	$\sim 0.0\%$
OGI Alphadigits	36	36	8%
Resource Management (RM)	1.000	60	4%
Air Travel Information Service (ATIS)	1.800	12	4%
Wall Street Journal	20000	200-250	15%
Broadcast News	80.000	200-250	20%
Conversational Speech	50.000	100-150	30%

2.5.4 Decodificador

Os itens anteriores mostraram os modelos matemáticos para construção de modelos acústicos e de linguagem. Tem-se como etapa final a transcrição de amostras de voz desconhecidas para sua forma textual, tarefa desempenhada pelo decodificador. O processo de decodificação é controlado por uma rede de palavras construída a partir do modelo de linguagem. A rede consiste de um conjunto de nós (palavras) conectados por arcos, onde cada

arco possui uma uma probabilidade de ocorrência (transição). Tendo em mãos tal rede, o dicionário fonético e um conjunto de modelos HMMs estimados, pode-se determinar, dada uma amostra de voz desconhecida, a probabilidade de qualquer um dos possíveis caminhos que a rede pode percorrer. A tarefa do decodificador é encontrar aqueles caminhos que são mais prováveis.

O processo de busca é descrito pela Equação 2.11, onde, tendo a sequência de observação $O_1^T = o_1, o_2...o_T$, busca-se pela sequência de palavras $W_1^N = w_1, w_2...w_N$ que maximize $\hat{W}_1^N$. O termo $P(O_1^T|W_1^N)$ pode ser expandido em função do modelo acústico, assim tem-se S_1^T como uma hipótese de seqüência de estados, onde busca-se pelo W_1^N que maximize $P(O_1^T, S_1^T|W_1^N)$ levando em conta todas as possíveis seqüências de estados. O somatório pode ser substituído por uma maximização, em um processo conhecido como aproximação de Viterbi [28], em que se utiliza apenas o caminho mais provável.

$$\begin{aligned} \hat{W}_1^N &= \arg\max_{W_1^N}\{P(W_1^N)P(O_1^T|W_1^N)\} \\ &= \arg\max_{W_1^N}\{P(W_1^N)\sum_{S_1^T} P(O_1^T, S_1^T|W_1^N)\} \\ &= \arg\max_{W_1^N}\{P(W_1^N)\max_{S_1^T}(P(O_1^T, S_1^T|W_1^N))\} \end{aligned} \tag{2.11}$$

Como já mencionado, essa busca é realizada através de programação dinâmica (DP - *Dynamic Programming* [29]). A Figura 2.7 mostra um exemplo de reconhecimento de palavras isoladas, onde tem-se uma HMM com 3 estados para a palavra "camisa" e outra para palavra "azul". No eixo X tem-se os *frames* vindos do bloco de extração de parâmetros. Cada um dos possíveis caminhos possui uma probabilidade de ocorrência, que é calculada pela soma do logaritmo das probabilidades de cada transição entre estados (a_{ij}) e das probabilidades dos estados emissores gerarem as respectivas observações ($b_i(x_t)$). A linha em vermelho na figura denota um possível caminho a ser percorrido. Adotando S_1^T como sendo uma possível seqüência de estados $s_1, s2, ..., s_T$, temos que

$$P_{S_T} = \sum_{S_1^T} \log(a_{s_{T-1}s_T}) + \log(b_{s_T}(S_T)) \tag{2.12}$$

representa o cálculo da probabilidade de uma possível sequência de estados S_1^T. O algoritmo de *Viterbi* busca pela seqüência S_T com maior probabilidade P_{S_T}. Tendo em mãos as

máximas probabilidades para cada HMM das palavras “azul” e “camisa”, aquela que apresenta maior *score* (maior $P(O, S_T|\lambda)$) é escolhida pelo decodificador.

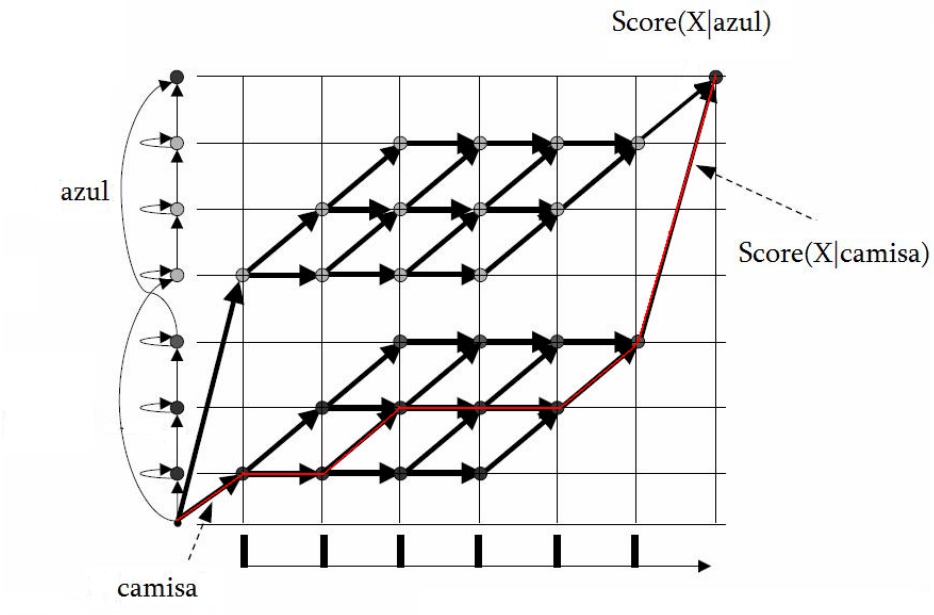

Figura 2.7: Exemplo de uma rede para reconhecimento de palavras isoladas.

Um raciocínio semelhante é feito para o reconhecimento de palavras conectadas. Inicialmente, as HMM’s que representam fonemas são concatenadas de maneira a formarem palavras, e tais palavras são concatenadas criando-se frases, como mostrado na Figura 2.8. De modo geral, no reconhecimento de palavras conectadas, tem-se uma rede formada por um grande conjunto de HMMs concatenadas.

As probabilidades de transições entre os estados e transições entre palavras são determinadas pelo modelo acústico e de linguagem, respectivamente. Para encontrar o caminho com maior probabilidade o decodificador faz uso do algoritmo *Token Passing* [30]. Um *token* pode ser entendido como um trecho de um caminho dentro da rede de palavras que se extende do tempo 0 até o tempo t. No tempo 0 um *token* pode iniciar em qualquer um dos possíveis nós de início.

A cada passo, os *tokens* caminham dentro da rede através das transições parando sempre que encontram um estado emissor. Quando existem múltiplos caminhos a serem seguidos a partir de um nó, é realizada uma cópia do *token* e ambos os caminhos são explorados em paralelo. Conforme os *tokens* percorrem a rede, os *logs* das probabilidades de transição entre estados e de emissão dos estados são incrementados. O número de *tokens* na rede Nt é limitado, sendo as-

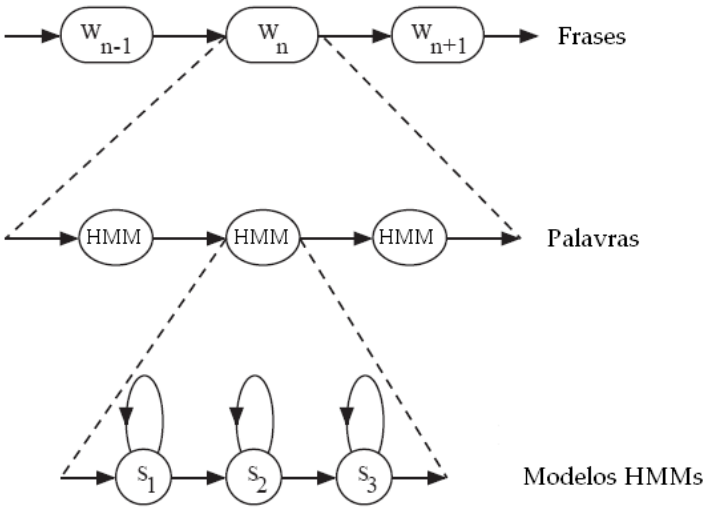

Figura 2.8: Modelos HMMs concatenados formando palavras e frases.

sim, no final de cada passo apenas os Nt melhores (com maiores probabilidades) são mantidos e os outros são descartados, tal processo é conhecido como *pruning*.

Cada *token* que passa pela rede mantém um histórico da sua rota. O nível de detalhe das informações guardadas depende do tipo de saída desejada. Normalmente apenas as seqüências de palavras são armazenadas, entretanto, é possível armazenar informações como a seqüência de estados percoridas em uma HMM e o intervalo de tempo em que a mesma ocorreu. O conjunto de informações armazenadas no histórico e o valor de Nt influenciam de maneira significativa nos recursos computacionais utilizados e no tempo de reconhecimento.

Uma vez que os dados de teste foram processados pelo decodificador, o próximo passo é a análise dos resultados. A medida de desempenho utilizada é a taxa de erro por palavra (WER) que pode ser definida como:

$$WER = \frac{S + D + I}{N} \tag{2.13}$$

onde N é o número total de palavras do conjunto de teste, S é o número de erros por substituição, D número de erros por deleção e I número de erros de inserção. O cálculo da WER é realizado facilmente através de ferramentas do HTK.

Capítulo 3

Recursos para Sistemas RAV

Para o desenvolvimento do sistema LVCSR, muitos recursos são necessários. Este cápítulo mostrar os principais recursos utilizados, desde vários *corpora*, softwares de domínio público e alfabeto fonético são descritos.

3.1 Alfabeto Fonético

Uma forma muito utilizada de se representar graficamente a pronúncia de um determinado som é através do *alfabeto fonético*. Sendo assim o alfabeto fonético permite a leitura de um som em qualquer idioma por uma pessoa treinada. Para que isso ocorra, o alfabeto deve apresentar convenções inequívocas e de maneira explícita.

Atualmente, existem vários alfabetos fonéticos utilizados em todo mundo, sendo o mais conhecido o alfabeto fonético internacional IPA (*Internacional Phonetic Alphabet*) [31]. A escolha do alfabeto fonético é de extrema importância na construção de sistemas LVCSR, já que os mesmos são utilizados na construção do dicionário fonético que é uma ferramenta essencial na estimação dos modelos acústicos. Na criação do sistema proposto foi utilizado o alfabeto fonético SAMPA[1], do inglês *Speech Assessment Methods Phonetic Alphabet*, que consiste em um sistema de escrita fonético baseada no IPA, o mesmo foi criado como alternativa para solucionar a impossibilidade dos codificadores de texto representarem os símbolos do IPA. O alfabeto SAMPA é descrito no Apêndice A.

Algumas modificações foram realizadas no alfabeto SAMPA para uma melhor adaptação ao PB, a tabela 3.1 mostra os símbolos fonéticos utilizados.

[1] "http://www.phon.ucl.ac.uk/home/sampa/index.html", visto em maio de 2008

Tabela 3.1: Símbolos Fonéticos utilizados na criação do sistema LVCSR para o PB.

v	a	i	s	i∼	k	u
z	E	r	t	e	o	f
l	p	g	a∼	n	e∼	S
m	o∼	d	O	u∼	h	J
b	R	Z	L			

3.2 O Corpus Spoltech

O Spoltech [32] é um *corpus* para o português brasileiro desenvolvido pela Universidade do Rio Grande do Sul, Universidade de Caxias do Sul e OGI (*Oregon Graduate Institute of Science and Technology*)[2] e subsidiado pelo CNPq, no Brasil e pelo NSF nos Estados Unidos. O corpus é distribuido pelo LDC *Linguistic Data Consortium*[3] e OGI. O mesmo consiste de gravações (através de microfones) de 477 locutores de várias regiões do Brasil com suas respectivas transcrições fonéticas e ortográficas. As gravações consistem tanto de leituras de frases curtas quanto de respostas a perguntas (fala espontânea). No total, o *corpus* é composto de 8.080 arquivos WAV, 2.540 arquivos com transcrições em nível de palavra (TXTs sem alinhamento temporal) e 5.479 arquivos com transcrições em nível de fonema (PHNs com alinhamento temporal).

O ambiente de gravação não foi controlado, sendo assim, algumas gravações foram feitas em estúdios e outras em ambientes ruidosos (feiras, escolas, etc). Os dados foram gravados a uma taxa de 44.1 KHz (mono, 16-bit).

Uma verificação inicial nos arquivos do *corpus* mostrou que o mesmo apresenta muitos problemas. Alguns arquivos WAV não possuem seus respectivos arquivos TXT e PHN, além disso muitos arquivos TXT e PHN apresentam vários erros. Outro problema é que muitos arquivos WAV apresentam péssima qualidade de gravação, ou seja, contém um elevado grau de ruído causado pelo ambiente de gravação, tais como risadas por parte dos locutores e bips no inicio de algumas gravações. Portanto antes de se utilizar o *corpus* uma trabalhosa revisão foi realizada com intuito de minimizar tais problemas.

Primeiramente todos os arquivos WAV foram ouvidos e uma seleção daqueles conside-

[2] "http://cslu.cse.ogi.edu/corpora" Visto em Março, 2008.

[3] "http://www.ldc.upenn.edu." Visited in March, 2008.

rados em bom estado foi realizada. No final foram selecionados 7238 arquivos WAV (89% do *corpus*). Na etapa seguinte os arquivos TXT inexistentes foram criados e aqueles que apresentavam erros foram corrigidos, todos esses arquivos foram disponibilizados para *download*. Como será explicado no Capítulo 4 os arquivos PHN não foram utilizados na criação do sistema RAV.

O corpus foi dividido em dois conjuntos disjuntos, um de treino e outro de teste. No treino dos modelos acústicos foram utilizados de 5541 arquivos WAV, que correspondem a 180 minutos. Para teste utilizou-se 1697 arquivos que totalizam 40 minutos.

3.3 O Corpus OGI-22

O *corpus* OGI 22 Language [33] para o português brasileiro é um esforço do *Center for Spoken Language Understanding* (CSLU), departamento integrante do OGI, juntamente com a Universidade de Caxias do Sul. Esse corpus de fala espontânea, independente de gênero e de locutor é bastante útil em pesquisas de processamento de voz, apesar do seu tamanho relativamente reduzido. O mesmo é composto de 2500 arquivos de áudio, nem todos com transcrição ortográfica e nenhum com transcrição fonética.

Os arquivos de áudio consistem de informações coletadas através de respostas diretas para perguntas, como idade e sexo, respostas espontâneas para questionamentos a respeito do clima local, rota para chegar ao local de trabalho, última refeição, entre outros; e dois com tema livre, um deles necessariamente na língua inglesa e o outro na língua nativa do locutor. Todas as gravações foram feitas por telefone (8 KHz). A fidelidade das transcrições ortográficas não é 100% garantida. Análises informais feitas durante o período de elaboração, indicarem um alto nível de inconsistência.

Do mesmo modo como feito para o Spoltech uma seleção dos arquivos de áudio foi realizada e as transcrições ortográficas do corpus OGI-22 foram verificados. Quando necessário, as transcrições ortográficas existentes foram corrigidas, as inexistentes criadas e as palavras ausentes foram inseridas no dicionário com a sua respectiva transcrição fonética.

Nos experimentos o conjunto de treino foi composto de 2.017 arquivos, que correspondem a 184,5 minutos e o conjunto de teste com 209 arquivos que totalizam 14 minutos. Os arquivos em inglês e aqueles que possuiam ruído excessivo não foram utilizados.

3.4 O CETENFolha

O CETENFolha [34] (Corpus de Extractos de Textos Electrônicos NILC/Folha de S. Paulo) é um corpus de cerca de 24 milhões de palavras em português brasileiro, criado pelo projeto de processamento computacional do português, com base nos textos do jornal Folha de S. Paulo, tais textos fazem parte do *corpus* NILC/São Carlos, compilado pelo Núcleo Interinstitucional de Lingüística Computacional (NILC). Para utilização do *corpus* foi necessário uma grande formatação, de forma que pudesse ser usada pela ferramenta HTK e SRILM. A formatação consistiu de:

- Remoção de pontuação e *tags* presentes nos textos ([ext], [t], [a], entre outras).
- Conversão de letras maiúsculas para minúsculas.
- Conversão de números para forma extensa.
- Expansão de siglas (PM, MEC, PT , etc).
- Correção de palavras.

Um trecho do corpus antes da formatação é mostrado abaixo:

```
<ext id=165691 cad="Mundo" sec="pol" sem="94a">
<s> <t> Recuo dilui dividendo político do presidente </t> </s>
<s> <situacao> De Washington </situacao> </s>
<p>
<s> No primeiro momento, a ≪Operação Sustentar a Democracia≫
traz dividendos políticos para Bill Clinton . </s>
<s> Mas o futuro é incerto . </s>
</p>
<p>
<s> O Senado tem uma <<caixa preta>> de R$ 2 milhoes </s>
</p>
<p>
<s> Depois de ter anunciado com solenidade que ≪o tempo dos ditadores
haitianos acabou≫, Clinton voltou atrás e concordou em negociar . </s>
</p>
</ext>
```

Após a formatação o mesmo trecho:

```
<s> Recuo dilui dividendo político do presidente </s>
<s> situacao De Washington </s>
<s> No primeiro momento a operação sustentar a democracia
traz dividendos políticos para bill clinton . </s>
<s> Mas o futuro é incerto . </s>
<s> o senado tem uma caixa preta de dois milhoes de reais . </s>
<s> Depois de ter anunciado com solenidade que o tempo dos ditadores
haitianos acabou clinton voltou atrás e concordou em negociar. </s>
```

Como resultado o *corpus* formatado possui 1,5 milhões de frases e aproximadamente 23 milhões de palavras.

3.5 Softwares Utilizados

Na construção do sistema RAV foi indispensável a utilização de vários *toolkits* disponíveis livremente na internet, tais como: HTK, ATK, SRLIM, Weka e os sistemas Unix.

3.5.1 HTK - *The Hidden Markov Model Toolkit*

O HTK[4] é um toolkit criado para construção e manipulação de cadeias escondidas de Markov. O HTK foi desenvolvido para pesquisa em reconhecimento de voz, porém como pode modelar qualquer série temporal, o mesmo tem sido usado em inúmeras aplicações, como por exemplo na pesquisa de síntese de voz, reconhecimento de caracteres e sequenciamento de DNA. O *toolkit* foi desenvolvido no laboratório de inteligência de máquina do departamento de engenharia da Universidade de Cambridge (CUED).

O HTK consiste em uma biblioteca de módulos e ferramentas disponível na linguagem C. As ferramentas proporcionam facilidades para análise da fala, treino de modelos ocultos de Markov, teste e análise de resultados. O software suporta implementação de HMMs contínuas e discretas, ambas com múltiplas distribuições gaussianas, o que permite a construção de sistemas com alto grau de complexidade. Para o caso de sistemas RAV, a Figura 3.1 mostra os principais estágios. No primeiro estágio as amostras de voz com suas respectivas transcrições

[4]http://htk.eng.cam.ac.uk/

são utilizadas pelas ferramentas do HTK para estimar os parâmetros das HMMs. No estágio seguinte declarações desconhecidas são transcritas através das ferramentas de reconhecimento (decodificação) do HTK.

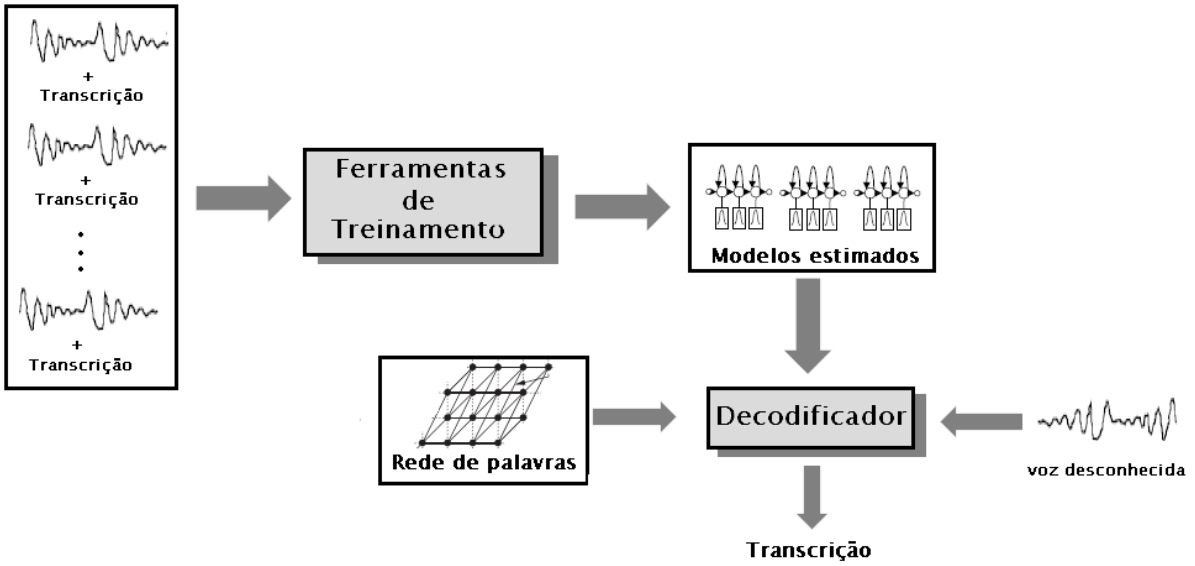

Figura 3.1: Principais estágios na criação de sistemas RAV com o HTK.

3.5.2 ATK - API para o HTK em Tempo Real

O ATK[5] é uma API desenvolvida para facilitar a realização de experimentos através do HTK. O mesmo consiste de uma coleção de bibliotecas em C++ que fazem uso das bibliotecas do HTK para testes em tempo real. Dentre as vantagens do ATK temos como a principal, o decodificador (*AVite*) com suporte a modelos de linguagem trigramas, o que não era possível com o decofificador do HTK (*HVite*), além disso, o (*AVite*) apresenta uma maior velocidade de reconhecimento. Com uma estrutura multiplataforma o mesmo pode ser usado em sistemas Unix e Windows, porém apenas a versão para o Windows possui interface gráfica.

O ATK possui um ambiente de programação bastante flexível para construção de sistemas de diálogo, já que também conta algumas ferramentas para síntese de voz (apenas para o inglês) no pacote *CMU Flite*. O ATK também possui suporte a múltiplos processos em paralelo (*multi-threaded*), ou seja, permite múltiplas entradas de usuário (comandos de voz, cliques do mouse, gestos, etc) sejam combinadas em um único pacote de dados. O seu decodificador pode retornar

[5]"http://mi.eng.cam.ac.uk/research/dialogue/atk_home ", visto em Maio, 2008

os resultados do reconhecimento palavra por palavra, de maneira a diminuir a latência e permitir respostas rápidas do sistema.

3.5.3 SRILM - *SRI Language Modeling Toolkit*

O SRILM[6] é um *toolkit* para construção e manipulação de modelos estatísticos de linguagens, criado para aplicações em reconhecimento de voz e processamento de linguagem natural. Criado pelo *SRI Speech Technology and Research Laboratory* o SRILM ainda está em processo de desenvolvimento, mas já conta com um grande número de ferramentas. O seu desenvolvimento foi baseado em algumas ferramentas de construção de MLs do HTK (*CMU-Cambridge toolkit*), porém, diferente do HTK, o mesmo implementa mais opções para construção de MLs, além de várias técnicas de suavização, o que o torna muito útil na criação do sistema prosposto neste trabalho. O software possui código aberto, além de ser multiplataforma e já possuir uma boa documentação. Uma maior descrição do software pode ser vista em [35].

O SRILM consiste basicamente dos seguintes componentes:

- Um conjunto de bibliotecas em C++ que implementam de maneira eficiente algoritmos para modelos de linguagem, suporte a estruturas de dados e a uma variada gama de funções.

- Uma variedade de programas executáveis que utilizam as bibliotecas para executar tarefas padrões como treino e teste de MLs.

- Uma coleção de *scripts* que facilitam a maioria das tarefas citadas, além de uma API (*Application Programming Interface*) composta por vários *toolbox* de comandos pra criação e testes de MLs.

- Além de implementar MLs baseados em palavras (*word-based models*), o SRILM também pode implementar vários outros tipos de MLs, tais como: modelos baseados em classes (*Class-based models*), modelos baseados em eventos ocultos (*Disfluency and hidden event language models*), modelos com interpolamento dinâmico (*Dynamically interpolated LMs*), dentre outros.

[6] http://www.speech.sri.com/projects/srilm/, visto em Maio, 2008

3.5.4 WEKA - *Waikato Environment for Knowledge Analysis*

O WEKA[7] é uma ferramenta de aprendizado de máquina de domínio público desenvolvida na Universidade de Waikato na Nova Zelândia [36]. O sistema construído na linguagem Java, consiste em um conjunto de implementações de algoritmos para tarefas de mineração de dados. O Weka possui ferramentas para pré-processamento de dados, classificação, métodos de aprendizagem não-supervisionada, entre outros.

A linguagem Java o torna acessível a várias plataformas computacionais, além do código aberto que facilita a adaptação dos algoritmos de aprendizagem de máquina para casos específicos. Como será descrito no próximo capítulo, o WEKA foi utilizado para construção do dicionário fonético para o PB.

3.5.5 Sistema Operacional Linux

No desenvolvimento do sistema descrito neste trabalho, o sistema operacional Linux foi uma ferramenta essencial. Os sistemas Unix apresentam uma vasta gama de ferramentas muitos úteis na manipulação de grandes quantidades de arquivos (os *corpora* utilizados, por exemplo), manipulação de arquivos de texto (CETENFolha), criação de scripts para treino e teste do sistema, entre outras.

[7] http://www.cs.waikato.ac.nz/ml/weka/

Capítulo 4

Desenvolvimento do Sistema RAV para PB

Neste capítulo é descrito todo o processo de desenvolvimento do sistema LVCSR para o PB. Cada um dos blocos de construção do sistema, junto com as respectivas ferramentas do software HTK utilizadas são descritos. Na última seção temos os resultados obtidos com as várias simulações, onde buscou-se entender como se comportam os principais componentes de um sistema LVCSR.

4.1 Construção do Dicionário Fonético

Na construção de sistemas LVCSR é indispensável a posse de um dicionário fonético de grande vocabulário. Basicamente o dicionário consiste de um conjunto de palavras com suas respectivas transcrições fonéticas. Infelizmente, tal recurso não é disponibilizado para o português brasileiro, sendo assim, foi necessário a construção de um conversor grafema-fonema (*grafene to foneme - g2p*) para realizar tal tarefa. O dicionário utilizado nesse trabalho foi obra da ex-aluna de mestrado do LaPS Chadia Hosn [37], que também compunha o corpo do já citado projeto *falabrasil*. Para criação do conversor grafema-fonema para o PB fez-se uso do programa Weka.

Apesar do Português ser considerado um lingua de ortografia "homogênea" quanto a pronúncia, a tarefa de mapeamento grafema-fonema não é tão simples, já que a correspondência entre grafema e fonema (e vice-versa) em uma palavra não de é um para um, como por exemplo no fonema /s/ nas palavras *céu*, *surdo* e *máximo*.

Inicialmente uma primeira versão do dicionário (UFPAdic 1.0) com 11.827 palavras foi construída a partir de versões comerciais de dicionários eletrônicos. Mesmo com o tamanho re-

lativamente pequeno, o mesmo possui um tamanho semelhante a dicionários utilizados em outros estudos [38] e foi suficiente para treino do conversor grafema-fonema. Além do UFPAdic, dicionários de outras línguas foram obtidos com a finalidade de validar os resultados do conversor. Os dicionários *NetTalk* [39] (Inglês Americano) com 20.000 palavras, *Brulex* [40] (Francês) com 27.473 palavras e Beep 1.0 [41] (Inglês Britânico) com 256.980 palavras encontram-se disponíveis on-line na página do *Pronalsyl* [1].

Como etapa inicial tem-se o alinhamento da base, a partir desse alinhamento obtem-se os dados de onde serão extraídas as regras para transcrição. A tarefa de alinhamento grafema-fonema pode ser feita manualmente, mas a medida que o tamanho do dicionário aumenta, a mesma se torna inviável. Para alinhamento da base utilizou-se de programação dinâmica que realizou o mapeamento grafema-fonema através de várias iterações como feito em [42]. Um exemplo do processo de alinhamento pode ser visto na Figura 4.1

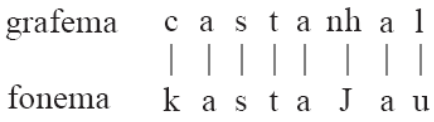

Figura 4.1: Alinhamento grafema-fonema para a palavra *castanhal*.

Após o alinhamento, o dicionário foi dividido em dois conjuntos, um de treino e outro de teste. Tanto para treino como para teste foram utilizados algoritmos de classificação do *software* Weka, em um aprendizado supervisionado que tem por base informações sobre os grafemas e fonemas no dicionário usado. Mais especificamente foi utilizado o algoritmo de Árvore de Decisão J.48 [36]. As técnicas de aprendizagem por indução, como o J.48, são capazes de identificar características em comum em dados e generalizá-las, o que corresponde ao treino do classificador. A partir dessas informações o algoritmo cria regras, de forma que, dada uma palavra desconhecida, o classificador é capaz de transcrevê-la. O *software* Weka possui um formato de arquivo próprio, sendo assim, foi necessária uma conversão do dicionário para o formato ARFF como feito em [3].

No processo de construção do conversor, considera-se que uma letra possui diferentes realizações acústicas e símbolos fonéticos dependendo do contexto em que a mesma se encontra, ou seja, o som produzido pela pronúncia da letra sofre influência das suas letras vizinhas.

[1] http://www.pascal-network.org/challenges/pronalsyl/ , visto em janeiro 2006

A Figura 4.2 mostra o exemplo para a palavra *leão* com análise de contexto igual a 2, onde uma letra sofre influência das 2 letras vizinhas da esquerda e da direita.

Esquerda 2	Esquerda 1	Análise	Direita1	Direita2	Fone
#	#	l	e	ã	l
#	l	e	ã	o	e
l	e	ã	o	#	a~
e	ã	o	#	#	u

Figura 4.2: Análise do grafema *leão* para o contexto igual a 2.

Na construção do classificador, utilizou-se de um contexto igual 5, tendo a letra examinada como a central, seguida de suas respectivas letras vizinhas a esquerda e a direita. Alguns símbolos foram utilizados para identificar situações especiais, tais como o ”#”para completar contextos e o “-”para fonemas nulos. Com o classicador treinado, seguiu-se a etapa de testes, os resultados podem ser vistos na Tabela 4.1.

Tabela 4.1: Resultados dos testes com o classificador J.48 utilizando contexto igual a 5.

Dicionário	**Taxa de erro por fonema (%)**	**Taxa de erro por palavra (%)**
NetTalk	10.94	53.66
Brulex	2.06	12.56
Beep 1.0	5.04	37.5
UFPAdic1.0	1.6	10.92

Os resultados obtidos foram satisfatórios para o PB e compatíveis com os encontrados para literatura para os dicionários de lingua estrangeira. Após a validação dos resultados, um novo treinamento do classificador foi realizado utilizando-se de todo o dicionário UFPAdic1.0. Assim, novas palavras foram obtidas de várias fontes, como o jornal Folha de São Paulo, e então repassadas ao classificador. Como resultado obteve-se um dicionário fonético para o PB com mais de 60 mil palavras.

4.2 Treino do Modelo Acústico

Esta seção mostra todos os passos e ferramentas usadas para criação dos modelos acústicos utilizados no sistema desenvolvido, desde a preparação dos dados até técnicas de compartilhamento de dados que foram abordadas. O processo de treino foi realizada com ferramentas do software HTK junto com vários *scripts* Java, necessários para criação de arquivos específicos, como será mostrado adiante.

4.2.1 Preparação dos Dados

Como descrito no capítulo 2, o primeiro bloco da criação de um sistema LVCSR consiste na extração de parâmetros do sinal de voz. Para realizar tal tarefa, fez-se uso da ferramenta **HCopy** do HTK. A mesma aceita como entrada vários tipos de arquivos, além do formato WAV, e os converte para o tipo paramétrico desejado. A Figura 4.3 mostra todos os possíveis formatos de entrada e saída suportados.

	Formatos de saída										
Formatos de entrada	WAVEFORM	LPC	LPREFC	LPCEPSTRA	IREFC	MFCC	FBANK	MELSPEC	USER	DISCRETE	PLP
WAVEFORM	√	√	√	√	√	√	√	√		√	
LPC		√	√	√	√					√	
LPREFC		√	√	√	√					√	
LPCEPSTRA		√	√	√	√					√	
IREFC		√	√	√	√					√	
MFCC						√				√	
FBANK						√	√			√	
MELSPEC						√	√	√		√	
USER									√	√	
DISCRETE										√	
PLP										√	√

Figura 4.3: Tipos de codificação suportados pela ferramenta **HCopy**.

Como já citado, utilizou-se dos paramêtros MFCCs como representação acústica. No processo de extração, para cada arquivo de entrada um outro correspondente contendo os vetores de parâmetros é criado (vide Figura 4.4). Para utilização da ferramenta é necessário um *script* de configuração que descreve todas as características desejadas na conversão do sinal. O *script* de configuração usado na criação do modelo acústico é mostrado abaixo:

```
Arquivo de configuração: hcopy-wav.conf

TARGETKIND = MFCC_E_D_A   # tipo paramétrico
TARGETRATE = 100000.0     # taxa de amostragem em centésimos de nano segundos
SAVECOMPRESSED = T        # Salvar o arquivo de saída com compressão
SAVEWITHCRC = T           # Adiciona um "checksum" a saída do arquivo
WINDOWSIZE = 250000.0     # Tamanha da janela de análise
USEHAMMING = T            # Utilizar o janelamento de Hamming
PREEMCOEF = 0.97          # Indica o coeficiente de pré-ênfase
NUMCHANS = 26             # Número de canais no banco de filtros
CEPLIFTER = 22            # Coeficiente de liftering cepstral
NUMCEPS = 12              # Número de parâmetros ceptrais
ENORMALISE = T            # Normalização do log da energia do sinal
SOURCEFORMAT = WAV        # Formato do arquivo de entrada
SOURCEKIND  = WAVEFORM    # Tipo de codificação de entrada
ZMEANSOURCE = T           # Fonte com formato de onda com média zero
SOURCERATE  = 227         # valor definido como: 10000/fs
```

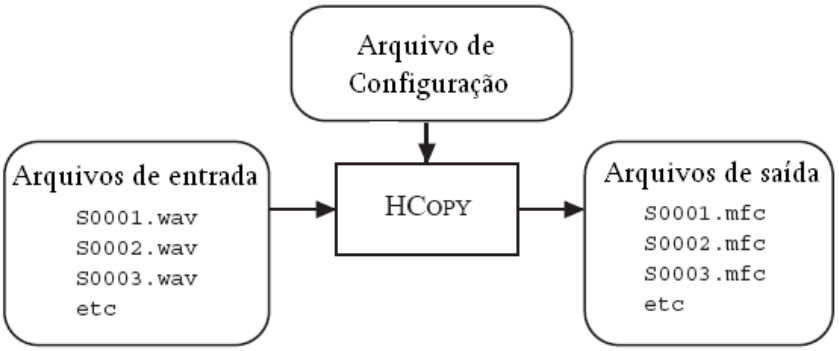

Figura 4.4: Funcionamento da ferramenta **HCopy**.

Basicamente, utilizou-se a mesma configuração para ambos os *corpora*, diferindo apenas no valor da variável SOURCERATE, que depende da frequência de amostragem do sinal, onde temos 44,1 KHz para o Spoltech e 8 KHz para o OGI-22. Como saída temos 13 parâmetros (12 coeficientes cepstrais mais a energia), dos quais são extraídos as primeira e segunda derivada resultando em ((12+1)*3) 39 parâmetros representando cada *frame* do sinal.

4.2.2 Criação de Modelos Monofones

A estratégia utilizada na construção de modelos acústicos com a ferramenta HTK é que as HMMs devem ser refinadas gradualmente. Nesse sentido, iniciamos com um simples conjunto de modelos, baseados em fonemas, com uma única gaussiana por mistura e independente do contexto, então gradualmente refina-se tais modelos, os expandindo para modelos dependentes do contexto (trifones) e usando distribuições com múltiplas misturas de componentes gaussianas.

Primeiro, um conjunto inicial de modelos (protótipos) é criado, cada modelo HMM contem 3 estados emissores e 2 não emissores utilizando a estrututa *left-right*. O sistema construído utiliza 32 fonemas mais um fonema que representa o silêncio. Após a criação dos protótipos, a ferramenta **HCompV** é utilizada para calcular médias e variâncias globais, que são replicadas para todos os modelos HMMs criados, sendo assim, todos os modelos iniciais possuem estados (médias e variâncias) e matrizes de transição iguais.

Os modelos HMM são salvos no formato MMF (*Master Macro File*), que consiste de 2 arquivos: o arquivo *hmmdefs* que armazena todos os fonemas com seus respectivos estados e matrizes de variância, e o arquivo *macros* que guarda o tipo paramétrico utilizado e a variável *vFloors* gerada pelo **HCompV**, essa variável armazena o valor base de variância, que é igual a 0.01 vezes o valor da variância global.

Para treino dos modelos faz-se uso da ferramenta **HERest**, a mesma realiza o treinamento *embedded* do sistema através do algoritmo de *Baum-Welch*. Para utilização dessa ferramenta é necessário a criação de um arquivo MLF (*Master Label File*) que contém as transcrições fonéticas dos arquivos de treino. Desse modo, inicialmente cria-se o arquivo MLF com transcrição a nível de palavra (*words.mlf*), tal arquivo é criado através de um *script* Java (*TxtToMLF.jar*) que tem como entrada a lista de todos os arquivos com transcrições ortográficas. Então a ferramenta **HLEd**, junto com o dicionário fonético, realiza a conversão das palavras para sua respectiva representação fonética (*phones.mlf*), como mostrado na tabela 4.2.

O **HERest** segmenta os dados (MFCC's) uniformemente de acordo com o MLF (*phones.mlf*), e executa simultâneamente uma única re-estimação de Baum-Welch sobre todo o conjunto de modelos HMMs.

Para as amostras de treino os correspondentes modelos HMMs são concatenados e então 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

Tabela 4.2: Exemplos dos arquivos MLF com transcrições a nível de palavra e fonema.

words.mlf	**phones.mlf**
“*/BR-00310.work.lab”	“*/BR-00310.work.lab”
estudante	sil
.	e
	s
	t
	u
	d
	a∼
	t
	i
	sil
	.

processados, as estatísticas acumuladas são usadas para computar os parâmetros das HMMs re-estimadas.

A probabilidade de transição entre dois estados não emissores é zero, de forma que é obrigatório a ocorrência de pelo menos uma observação (estado emissor), caso ocorra algum tipo de ruído durante o reconhecimento o mesmo é contabilizado, diminuindo o desempenho do decodificador. Um outro problema é que, como o sistema se propõe a modelar a fala continua, dificilmente ocorrerá silêncio entre as palavras, o que dificulta a identificação de onde começa e termina uma palavra, para redução desses problemas faz-se uso do modelo *short-pause* (também conhecido como *tee-model*) [21], que consiste em um modelo HMM composto somente de 1 estado emissor e 2 não emissores, o modelo possibilita a não emissão de nenhuma observação, representado pela transição entre os estados não emissores, isso torna o modelo mais robusto, permitindo que o mesmo absorva ruídos impulsivos. Para criação do modelo *short-pause*, representado pela sigla **sp**, faz-se uma cópia do estado central do modelo do silêncio (**sil**), seguida por um vínculo do estado central do modelo do silêncio com o estado emissor do modelo *short-pause*, de forma que, durante o treino, os mesmos compartilhem os mesmos parâmetros, como mostrado na Figura 4.5. O Vínculo de estados é realizado através da ferramenta **HHEd**. Após

o vínculo de estados, faz-se uma modificação no arquivo de treino *phones.mlf*, onde se insere o fone **sp** na fronteira entres as palavra.

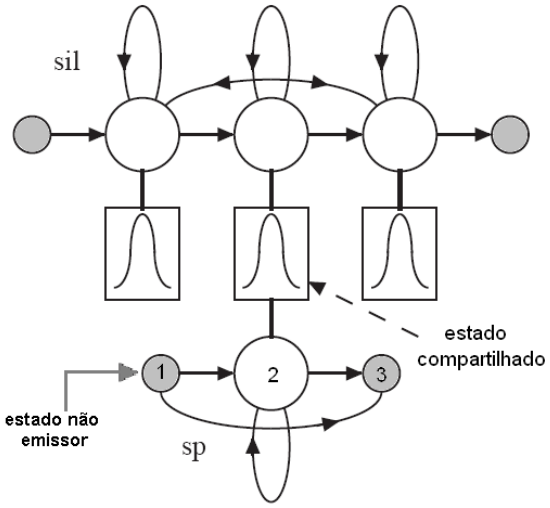

Figura 4.5: Modelo HMM do *short-pause* que compartilha parâmetros com o modelo do silêncio.

Além disso, é necessária a inserção da HMM **sp** no final de cada palavra do dicionário fonético, como mostrado abaixo. Assim, rápidas pausas podem ser identificadas, facilitando a identificação do fim de uma palavra durante o reconhecimento.

```
...
absolutamente    a b s o l u t a m e~ t i sp
absolutas        a b s o l u t a s sp
absolutismo      a b s o l u t i z m u sp
absoluto         a b s o l u t u sp
absolutos        a b s o l u t u s sp
...
```

Uma outra opção de treino de monofones é a utilização da ferramenta **HRest**, esta realiza o treinamento no estilo de palavras isoladas. Tal ferramenta se utiliza dos arquivos PHNs que contém o tempo de inicio e fim de cada fonema do arquivo WAV correspondente. Tendo e mãos o tempo exato de ocorrência de cada fonema o **HRest** estima os parâmetros dos modelos monofones de forma mais precisa. Porém dos 7.238 arquivos WAV do *corpus* Spoltech, apenas 5.479 arquivos possuem seu respectivo PHN, além disso, o alfabeto fonético utilizado para criação das transcrições fonéticas foi o *WordBet*. Uma checagem nos PHNs foi feita, e foram encontrados aproximadamente 103 símbolos fonéticos diferentes. Um experimento foi realizado com intuito de minimizar tal problema. Um mapeamento do *WordBet* para o SAMPA foi feito, seguido

por um treinamento usando a ferramenta **HRest**, porém os resultados foram muito semelhantes aos encontrados utizando o treinamento *embedded* da ferramenta **HERest**. Levando em conta a limitação da quantidade de dados que poderiam ser utilizados, e os possíveis erros de transcrição fonética, optou-se por não se utilizar das transcrições fonéticas.

4.2.3 Criação de Modelos Trifones

Como etapa seguinte, tem-se a adaptação das HMMs para modelos dependentes de contexto, para isso é utilizada a ferramenta **HLEd** do HTK. Para utilização do **HLEd** é necessário um *script* (*mktri.led*) contendo as modificações a serem realizadas para a expansão de modelos monofones para modelos trifones, além disso, a ferramenta modifica o arquivo MLF utilizado para treino, de forma a representar modelos dependentes de contexto (*wintri.mlf*). Como mostrado na subseção 2.5.2.2, modelos trifones podem ser do tipo *cross-words* ou *word-internal*. Para as simulação, ambos os tipos foram treinados e testados como será mostrado na seção de resultados.

Após a criação do MLF trifone e da lista de trifones, faz-se uso da ferramenta **HHEd**, a qual clona todos os trifones e realiza o vínculo das matrizes de transição. Parte do *script* (*mktri.hed*) é mostrado abaixo.

```
CL trifone
TI T_v {(*-v+*,v+*,*-v).transP}
TI T_a {(*-a+*,a+*,*-a).transP}
TI T_i {(*-i+*,i+*,*-i).transP}
TI T_s {(*-s+*,s+*,*-s).transP}
```

O comando CL tem como argumento o arquivo com a lista de todos os trifones (e bifones), o comando realiza a clonagem de cada trifone, onde todos os trifones herdam os 3 estados do seu monofone central, seguido pelo comando TI que vincula todas as matrizes de transição para cada conjunto de trifones que possuem o mesmo estado central, como mostrado na Figura 4.6. Após essas modificações, novamente os modelos passam pela re-estimação de Baum-Welch com a ferramenta **HERest**.

4.2.4 Vínculo de Estados

Como próximo passo na construção do modelo acústico tem-se o vínculo de estados dos trifones, esse vínculo permite o compartilhamento de dados, de forma a se obter uma

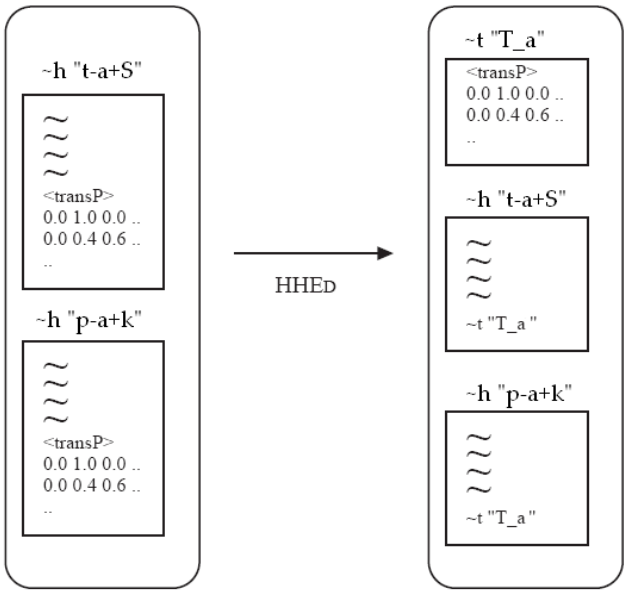

Figura 4.6: Vínculo das matrizes de transição para trifones com o mesmo fone central

estimação mais robusta dos parâmetros das HMMs. Com o intuito de otimizar os modelos trifones, o método escolhido foi o vínculo através de árvore de decisão fonética, como descrito na subseção 2.5.2.3. A árvore de decisão fonética usada neste trabalho foi construída através de uma vasta pesquisa sobre classificação de fonemas, procurando agrupar fonemas com características semelhantes, um trecho da árvore é visto abaixo. O *script* contendo a árvore completa é descrito no Apêndice B.

```
QS  "R_V-Oral"      { *+a,*+e,*+i,*+o,*+u,*+O,*+E }
QS  "R_V-Front"     { *+i,*+E,*+e }
QS  "R_V-Central"   { *+a }
QS  "R_V-Back"      { *+u,*+o,*+O }
QS  "R_V-Aberta"    { *+a,*+E,*+O }
```

Cada comando QS define uma questão, e cada questão é definida para os contextos a esquerda e a direita do trifone. Por exemplo, a primeira questão da árvore *R_V-Oral* é verdadeira se o contexto a direita corresponde á alguma dentre as vogais orais: *a, e, i, o, u, O* ou *E*. Uma descrição completa dos comandos do *script* pode ser vista em [21].

Para aplicação da aŕvore é utilizada a ferramenta **HHEd**. Tendo como entrada os arquivos de definição de HMMs, (*macros* e *hmmdefs*), e o *script* contendo a árvore (*tree.hed*), a ferramenta tem como saída o arquivo *tiedlist* contendo todos os trifones, junto com os respectivos

rótulos mostrando com quais outros trifones os mesmos compartilham estados. A redução no número de trifones é notável, por exemplo, para o modelo acústico construído, tem-se aproximadamente 34.850 trifones lógicos (para *word-internal*), que passam a ser 2409 físicos após a aplicação da árvore, onde apenas as HMMs físicas é que são modificadas durante o treino.

4.2.5 Mistura de Gaussianas

Como etapa final tem-se a implementação de múltiplas componentes gaussianas nas HMMs. O mecanismo utilizado para realizar tal tarefa é o comando MU da ferramenta **HHEd**, o qual incrementa o número de componentes gaussianas, tal processo é conhecido como *mixture splitting*. Normalmente o número de gaussianas é incrementado gradualmente, onde cada incremento é seguido por várias re-estimações através do **HERest**, esse processo se repete até atingir o número desejado. A ferramenta **HHEd** tem como argumento o arquivo de definições de HMM, seguido pelo *script* que contem o comando MU, como exemplificado abaixo:

```
MU 2 {*.state[2-4].mix}
```

O comando incrementa, para todas as HMMs, o número de componentes gaussianas dos estados 2, 3 e 4, que são os estados que emitem observação. Depois de vários testes e pesquisas, foi observado que a melhor seqüência de incremento foi de 1 gaussianas para 2, seguido de incrementos duplos (1 gauss/mix, 2 gauss/mix, 4, 6, 8, etc). Como será mostrado na seção de resultados, a partir de 14 componentes gaussianas o sistema sofre *overfitting*, onde os resultados tendem a estabilizar. A Figura 4.7 mostra o fluxograma do processo de criação do modelo acústico. O *script* completo utilizado para criação do modelo acústico se encontra no Apêndice C.

4.3 Treino do Modelo de Linguagem

Para treino do modelo de linguagem, foram utilizadas tanto ferramentas do HTK quanto do SRILM. Apesar de o HTK possuir um conjunto de ferramentas destinadas a essa tarefa (*HLMToolkit*), o SRILM possui uma variedade muito maior de configurações de treino que a torna mais eficiente para essa tarefa, principalmente na construção de modelos de linguagem do tipo *n*-grama. Primeiramente foi realizada a junção de todas as frases dos *corpora*: CETEN-Folha, Spoltech e OGI-22, de forma a se obter o maior número de frases possível. Após a junção

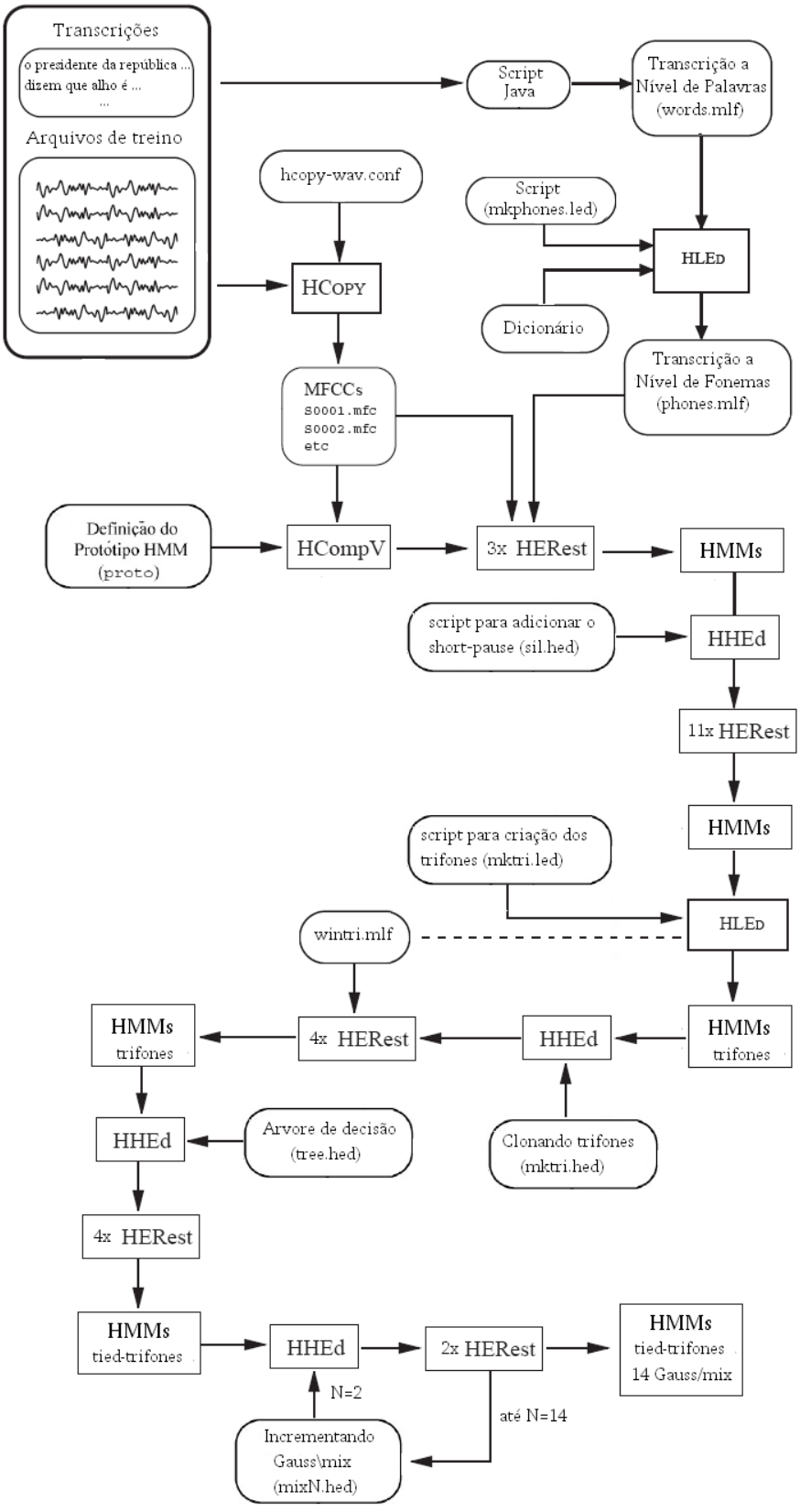

Figura 4.7: Fluxograma do processo de criação do Modelo Acústico

e formatação (remoção de pontuações, siglas, etc), o arquivo total de treino do ML contabilizou 1.360.000 frases, e para cálculo da perplexidade (teste), utilizou-se de 9.965 frases. Para uma melhor avaliação do sistema, vocabulários de tamanhos variados foram criados, variando de mil a 60 mil palavras, a escolha das palavras foi baseada no número de ocorrências das mesmas nos *corpora*. Além disso, modelos de linguagem contendo apenas as frases dos *corpora* Spoltech e OGI-22 também foram treinados e testados no decodificador. O formato utilizado para treino e teste foi o XML, que possui marcadores de inicio e final de sentença ($< s >$,$< /s >$), tais marcadores são contabilizados no treinamento do modelo tal como palavras, um pequeno trecho do arquivo XML é visto a seguir:

```
...
<s> a postura antiética de alguns políticos é inaceitável </s>
<s> cem duzentos trezentos </s>
<s> os trapezistas dão piruetas arriscados no ar </s>
<s> é necessário paciência de vez em quando </s>
<s> o orvalho da manhã as vezes é confundido com a chuva </s>
<s> quarenta cinquenta sessenta </s>
<s> espanha </s>
<s> os jogadores vitoriosos festejam o resultado </s>
<s> os cantores devem cuidar bem da goela </s>
<s> quarta feira </s>
<s> bife com batata frita </s>
...
```

Para treino utilizou-se a ferramenta **ngram-count** do SRILM, a qual pode ser usada para gerar ou manipular *n*-gramas. O *software* primeiramente conta o número de ocorrência dos bigramas e/ou trigramas a partir da leitura de um arquivo de texto, um dos parâmetros repassados à ferramenta é o vocabulário, visto que palavras encontradas nas frases de treino que não estejam no vocabulário (*OOV – Out of Vocabulary*) não terão suas freqüências contabilizadas, por seguinte tem-se na saída um modelo de linguagem *n*-grama no formato ARPA contendo o resultado das contagens. Dentre as várias opções para estimação de MLs, temos como as principais: a escolha do vocabulário e o método de suavização que será utilizado. Vários métodos de suavização foram testados, e o que apresentou melhor resultado (menor perplexidade) foi o modelo de desconto absoluto de Kneser-Ney [43]. Para cálculo da perplexidade fez-se uso da ferramenta **ngram** (SRILM), que a partir do modelo *n*-grama e de um conjunto de frases de teste (não vistas na etapa de treino) realiza o cálculo da entropia e perplexidade do ML.

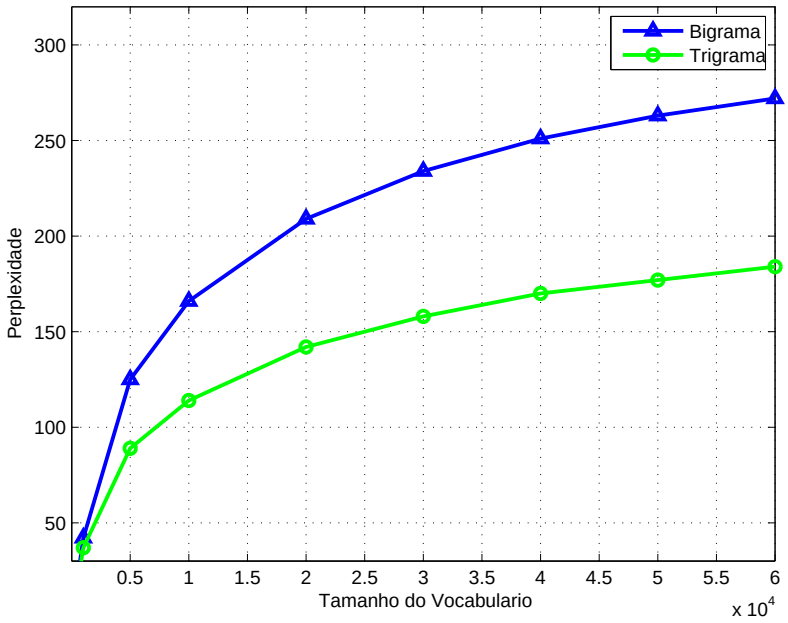

Figura 4.8: Perplexidade dos MLs Bigrama e Trigrama variando o tamanho do vocabulário.

Para teste do sistema através do decodificador *AVite*, é necessário uma rede de palavras que descreva os possíveis caminhos a serem seguidos durante o reconhecimento. Essa rede é construída a partir do modelo *n*-grama, através da ferramenta *HBuild* do HTK, entretanto a mesma possui a limitação de criar apenas redes bigramas, o que limitou o número de testes com o decoficador. O *script* utilizado para construção dos modelos de linguagem encontra-se no Apêndice D.

Vários tipos de MLs foram construídos e como medida de desempenho tem-se a perplexidade. A Figura 4.8 mostra as perplexidades encontradas para os modelos bigrama e trigrama treinados variando o número de palavras no vocabulário. É observado que quantos mais palavras se possui em um modelo de linguagem, maior sua perplexidade, já que assim, mais palavras que podem seguir uma dada palavra. Como forma de se obter um melhor entendimento de como o conjunto de treino impacta na perplexidade, testes foram realizados, fixando o vocabulário em 5 mil palavras, e variando o número de frases utilizadas para treino de 10 mil até 1 milhão de frases. Os resultados são mostrados na Figura 4.9.

É observada a diminuição da perplexidade à medida que se aumenta o número de frases no treino. Isso é explicado pela melhor estimação das probabilidades com um maior número de frases, já que assim, tem-se uma maior ocorrência de bigramas e trigramas no *corpus* de texto.

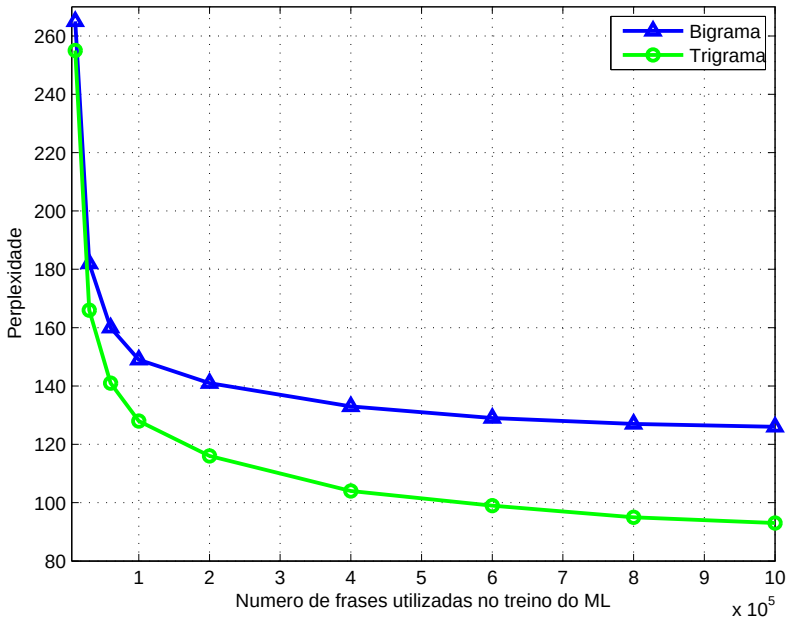

Figura 4.9: Perplexidade dos MLs Bigrama e Trigrama variando o número de frases no treino.

Os resultados são compatíveis aos encontrados em outros trabalhos [2] [4].

4.4 Resultados das Simulações

Esta seção vem mostrar os principais resultados obtidos, onde diferentes configurações de modelos acústicos , modelos de linguagem e tamanhos do vocabulário foram testadas, buscando assim entender como os mesmos se relacionam e o modo como influenciam na tarefa de decodificação. Para realização da tarefa de decodificação têm-se as ferramentas **HVite** do HTK e **AVite** do ATK. Nos testes, utilizou-se do **AVite** devido a vantagens já mencionadas na subseção 3.5.2. Vale ressaltar que o trabalho com o *corpus* Spoltech vem sendo realizado a mais tempo que o OGI-22, sendo assim, a maior parte dos resultados foram obtidos para o modelo acústico do Spoltech.

Nos primeiros testes, utilizou-se de modelos de linguagem bigrama criados somente com o conjunto de frases e vocabulários dos respectivos *corpora* Spoltech (1.104 palavras, 7.238 frases) e OGI-22 (3.285 palavras, 2.017 frases). Tais modelos apresentaram uma perplexidade igual a 7 e 22, respectivamente. Nas simulações, modelos acústicos trifones do tipo *word-internal* e *cross-word* foram estimados. Como avalição inicial tem-se a variação do número de

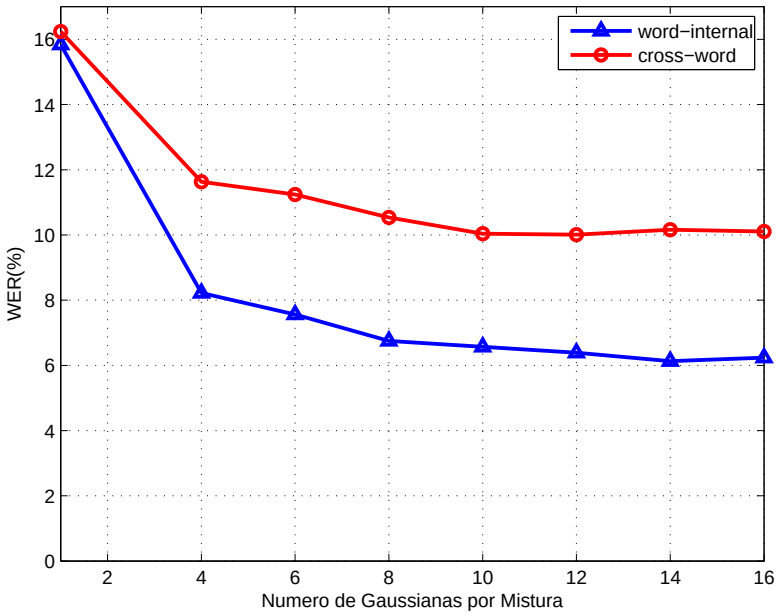

Figura 4.10: Variação da WER com o aumento do número de gaussianas para o MA do Spoltech.

gaussianas, partindo de uma única gaussiana a até 16 gaussianas por mistura. A Figura 4.10 exibe os resultados para o MA gerado com o Spoltech. É observado um melhor desempenho do MA com trifones *word-internal* que obtém um WER de 6,13% com 14 gaussianas. Isso é justificado pela insuficiência de dados na estimação dos modelos trifones do tipo *cross-word*, pois como mostrado na subseção 2.5.2.2 o número de trifones necessários para se adotar tal estratégia é bem superior ao número de trifones *word-internal*. Simulações semelhantes foram realizadas para o OGI-22, porém apenas com trifones *word-internal* como mostrado na Figura 4.11 onde se tem 23,4% de WER para 10 gaussianas.

Na segunda etapa dos testes, fixou-se o modelo acústico e simulações com modelos de linguagem bigrama com vocabulários de tamanhos variados foram realizadas. A proposta deste experimento é avaliar o comportamento do sistema LVCSR conforme se insere novas palavras no vocabulário. No treino dos modelos de linguagem utilizou-se 500 mil frases e vocabulários variando de 1.000 a 60 mil palavras. Ambos os modelos acústicos são do tipo *word-internal*, os resultados são apresentados nas Figuras 4.12 e 4.13, sendo que para o Spoltech utilizou-se 14 gaussianas e para o OGI-22 apenas 10. No Spoltech é observado um aumento inicial da WER quando se expande o vocabulário de 1.000 para 5 mil, já que novas palavras, além das presentes no próprio *corpus*, podem ser reconhecidas, tais palavras foram obtidas do *corpus*

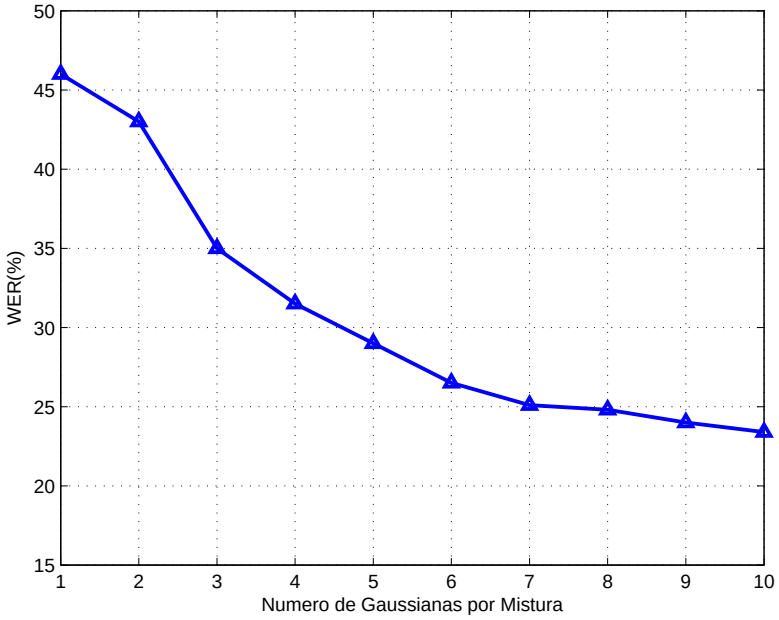

Figura 4.11: Variação da WER com o aumento do número de gaussianas para o MA do OGI-22.

CETENFolha. Após 10 mil palavras a WER permanece quase constante. No caso do OGI-22 tem-se uma pequena queda da WER seguida por um aumento a partir de 5 mil palavras. Após 20 mil palavras a WER tende a se estabilizar. O tempo médio de reconhecimento para frases de 5 a 12 palavras cresceu de 1,3 segundos/frase para 25,22 segundos/frase quando o vocabulário foi expandido de 1.000 para 60 mil nos testes com o Spoltech. Esse comportamento está ligado ao custo computacional adicionado à busca, uma vez que com mais palavras aumenta-se o número de possíveis caminhos na rede, gerando eventuais erros de reconhecimento como mostrado na Tabela 4.3.

Na terceira etapa dos testes, modelos de linguagem trigramas foram estimados. Utilizou-se de dois modelos trigramas, o primeiro construído somente com as frases do Spoltech e com um vocabulário de 1.104 palavras, e o segundo construído com 1 milhão de frases e com um vocabulário de 60 mil palavras. Os testes foram realizados apenas para o Spoltech, onde o modelo acústico utilizado foi do tipo *word-internal* com 14 gaussianas por mistura. Para o primeiro ML obteve-se 5,27% de WER, e com o segundo ML tem-se 30,33% de WER.

Pelos gráficos é observado um melhor desempenho no modelo acústico do Spoltech. Isso é justificado pela melhor qualidade nas gravações do Spoltech (44,1 KHz) em relação ao OGI-22 (8 KHz). Isso proporciona um melhor modelamento dos fonemas e consequentemente a uma

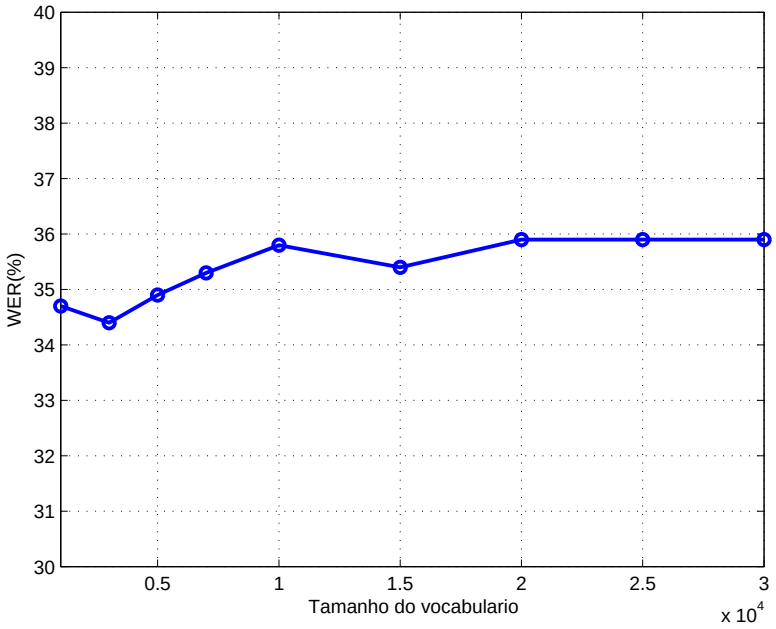

Figura 4.12: Variação da WER com o aumento do vocabulário para o OGI-22.

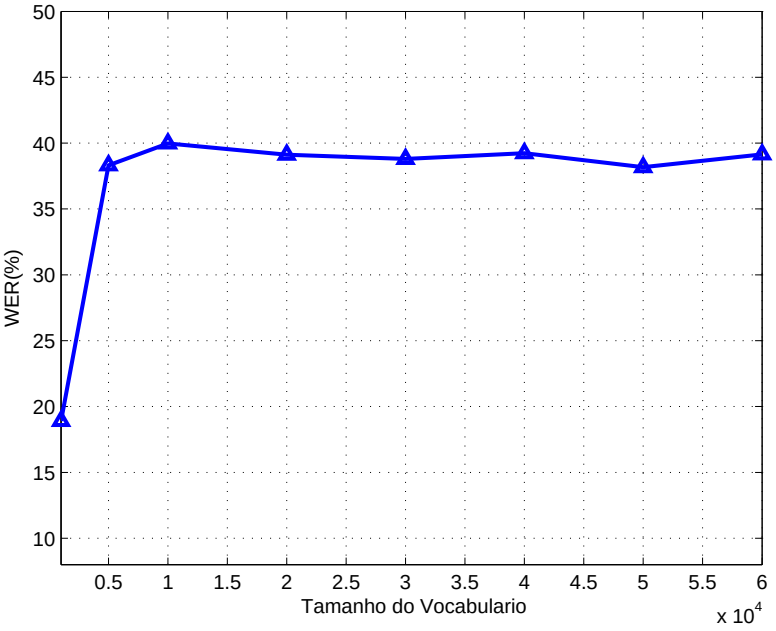

Figura 4.13: Variação da WER com o aumento do vocabulário para o Spoltech.

Tabela 4.3: Exemplo de erro durante a decodificação para um ML bigrama com vocabulário de 60 mil palavras.

Frase original	o orvalho da manhã as vezes é confundido com a chuva
Frase reconhecida	o orvalho da moeda é confundido com a chance

menor WER. Um experimento utilizando ambos os *corpora* para criação do modelo acústico foi realizado. Para isso foi necessário uma re-amostragem dos arquivos do Spoltech para 8 KHz. Os resultados encontrados foram bem abaixo do esperado, alcançando 34% de WER para 10 gaussianas e um vocabulário de 3.837 palavras, isso pode ser justificado pela grande perda de qualidade dos arquivos do Spoltech devido a re-amostragem.

Capítulo 5

Considerações Finais

Neste trabalho foram estudados vários aspectos sobre reconhecimento automático de fala contínua para grande vocabulários. Vários conceitos importantes foram abordados e soluções para problemas comuns da área foram apresentados. Foram criados e testados modelos acústicos e de linguagem com diferentes configurações, além da criação do conversor grafema-fonema utilizado para criação do dicionário fonético.

A implementação do sistema LVCSR apresentado neste trabalho foi baseada em cadeias ocultas de Markov e modelos de linguagem *n*-grama. Apesar das taxas de erro um pouco distantes das obtidas para o inglês (vide Tabela 2.2), os resultados obtidos foram satisfatórios se comparados a outros trabalhos da área para o PB [1–4]. As simulações com grandes e pequenos vocabulários mostram que o tamanho do vocabulário influencia até certo ponto na WER, quando se insere novas palavras, além das presentes no *corpus*, a WER aumenta, porém rapidamente se estabiliza. É observado também um aumento considerável na complexidade computacional e no tempo de reconhecimento conforme se aumenta o vocabulário. A utilização de mistura de gaussianas se mostrou eficaz no modelamento acústico, observadas as taxas de 6,13% para um vocabulário de 1.104 palavras e 39,14% para 60 mil, ambas utilizando 14 gaussianas e modelos de linguagem bigrama para o Spoltech.

Os resultados com o OGI-22 foram bem inferiores aos obtidos com o Spoltech, fato justificado pela baixa qualidade das gravações telefônicas, e seu tamanho reduzido. É observada a superioridade dos modelos de linguagem do tipo trigrama em relações aos bigramas, vistos pelos baixos valores de perplexidade obtidos nos testes com o CETENFolha e pelo melhor desempenho obtido nos testes com trigramas utilizando o Spoltech, onde tem-se 5,27% e 30,33% para vocabulários de 1.104 e 60 mil palavras respectivamente. Tal resultado é justificado pelo

fato de trigramas absorverem melhor a correlação entre as palavras, disciplinando de maneira mais eficiente as possíveis combinações entre as mesmas.

O tamanho reduzido dos *corpora* dificulta a construção dos modelos acústicos e conjuntos de textos de domínios diferentes, além do jornalismo (CETENFolha), são necessários para uma boa estimação de modelos *n*-grama. As ferramentas dos *toolkits* HTK, ATK e SRILM foram essenciais para a criação do sistema e na realização dos testes, porém existem vários outros *toolkits* de domínio público que podem ser utilizados para desenvolvimento de sistemas LVCSR, tais como o Sphinx 4 [44], ISIP [45] (C++) e Festival [46].

A construção de um sistema LVCSR para o PB é um grande desafio. Por isso, o número de pesquisadores na área de processamento de voz cresce a cada ano, novos *corpora* para o PB vêm sendo desenvolvidos, além de várias ferramentas. Os recursos desenvolvidos neste trabalho, tais como o dicionário fonético, as transcrições ortográficas corrigidas e *scripts* para estimação de modelos acústicos e de linguagem estão disponibilizados na página do projeto *falabrasil* [5]. Espera-se que outros grupos de pesquisa possam se utilizar de tais recursos para desenvolvimento de novos sistemas e aprimoramento dos resultados.

5.1 Publicações Geradas

Conferências:

- Patrick Silva, Nelson Neto, Aldebaro Klautau, Andre Adami, *Spoltech and OGI-22 Baseline Systems for Speech Recognition in Brazilian Portuguese*, International Conference on Computational Processing of Portuguese Language - PROPOR, 2008, 08-10 de setembro, Aveiro, Portugal.

- Patrick Silva, Nelson Neto, Aldebaro Klautau, Andre Adami, Isabel Trancoso, *Speech Recognition for Brazilian Portuguese using the Spoltech and OGI-22 Corpora*, XXVI Simpósio Brasileiro de Telecomunicações - SBrt, 2008, 02 a 05 de setembro de 2008, Rio de janeiro, Brasil.

Seminários:

- Carlos Patrick Alves da Silva, *Reconhecimento de Voz para o Português Brasileiro*, XVIII Seminário de Iniciação Científica da UFPA, 01-05 de setembro de 2007, Belém, Brasil.

5.2 Trabalhos Futuros

- Melhoria e expansão do dicionário fonético, o dicionário ainda apresenta alguns equívocos em suas transcrições e pode ser expandido de forma a cobrir um maior número de palavras para o PB.

- Aprimoramento dos modelos de linguagem. Através de técnicas de *Crawling* é possível retirar textos da internet de forma a se obter uma maior diversidade de textos.

- Obtenção de novos *corpora*. Muitos grupos de pesquisas vem trabalhando em conjunto para o desenvolvimento de novos *corpora* para o PB.

- Utilização de outros *toolkits* e outras técnicas de decodificação.

Referências Bibliográficas

[1] R. Fagundes and I. Sanches, "Uma nova abordagem fonético-fonológica em sistemas de reconhecimento de fala espontânea," *Revista da Sociedade Brasileira de Telecomunicações*, vol. 95, 2003.

[2] R. Teruszkin and F. Vianna, "Implementation of a large vocabulary continuous speech recognition system for brazilian portuguese," *Journal of Communication and Information Systems, vol. 21, no. 3, pages 204-218*, 2006.

[3] C. Hosn, "Conversão grafema-fone para um sistema de reconhecimento de voz com suporte a grandes vocabulários para o português brasileiro," Master's thesis, Universidade Federal do Pará, Centro Tecnológico, 2006.

[4] Ênio dos Santos Silva, "Desenvolvimento de um reconhecedor automático de voz com suporte a grandes vocabularios para o português brasileiro," Tech. Rep., 2005.

[5] "http://www.laps.ufpa.br/falabrasil," Visited in April, 2008.

[6] A. M. da Cunha and L. Velho, "Métodos probabilísticos para reconhecimento de voz," Laboratório VISGRAF - Instituto de Matemática Pura e Aplicada, Tech. Rep., 2003.

[7] L. Rabiner and B. Juang, *Fundamentals of speech recognition*. Englewood Cliffs, N.J.: PTR Prentice Hall, 1993.

[8] Jelinke and Frederick, "Métodos estatísticos para reconhecimento de voz," *The MIT Press*, 1998.

[9] M. R. K., "Results from a survey of attendees at asru 1997 and 2003," *INTERSPEECH 2005 Lisbon*, 2005.

[10] Kurzweil and Raymond, “”quando o hal vai entender o que estamos falando?”, em ”o legado de hal: o computador de ’2001 - uma odisséia no espaço’ como sonho e realidade”,” *The MIT Press*, 1998.

[11] L. Rabiner and R. Schafer, *Digital Processing of Speech Signals*. Prentice-Hall, 1978.

[12] S. Davis and P. Merlmestein, “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences,” *IEEE Trans. on ASSP*, pp. 357–366, Aug. 1980.

[13] J. Picone, “Signal modeling techniques in speech recognition,” *Proceedings of the IEEE*, vol. 81, no. 9, pp. 1215–47, Sep. 1993.

[14] L. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–86, Feb. 1989.

[15] L. R. Welch, “Hidden markov models and the baum-welch algorithm,” *IEEE Information Theory Society Newsletter.*, vol. 53, pp. 10–12, 2003.

[16] G. D. Forney, “The viterbi algorithm,” *Proceedings of the IEEE*, vol. 3, pp. 268–278, 1973.

[17] P. Woodland and S. Young, “The htk tied-state continuous speech recognizer,” *In: Proc. Eurospeech’93, Berlim*, 1993.

[18] S. Young and P. Woodland, “State clustering in hmm-based continuous speech recognition,” *Computer Speech and Language*, vol. 8, pp. 369–384, 1994.

[19] M. Hwang and X. Huang, “Shared distribution hidden markov models for speech recognition,” *IEEE Trans Speech and Audio Processing*, vol. 1, pp. 414–420, 1993.

[20] L. Bahl, P. Souza, P. Gopalakrishnan, D. Nahamoo, and M. Picheny, “Context dependent modeling of phones in continuous speech using decision trees,” in *DARPA Speech and Natural Language Processing Workshop*, 1994, pp. 264–270.

[21] S. e. Young, *The HTK Book*. Microsoft Corporation, Version 3.0, 2000.

[22] O. JJ, “The use of decision trees with context sensitive phoneme modelling,” Master’s thesis, Cambridge University Engineering Department, 1992.

[23] P. Woodland, J. Odell, V. Valtchev, and S. Young, "Large vocabulary continuous speech recognition using htk," *IEEE Int'l Conf. Acoustics, Speech, and Signal Processing, vol. 2, pp. 125–128, Adelaide*, 1994.

[24] R. Lippmann, "Speech recognition by machines and humans," *Speech Communication*, vol. 22, pp. 1–15, 1997.

[25] L. Pessoa, F. Violaro, and P. Barbosa, "Modelos da língua baseados em classes de palavras para sistema de reconhecimento de fala contínua," *Revista da Sociedade Brasileira de Telecomunicações*, vol. 14, no. 2, pp. 75–84, Dez 1999.

[26] E. Silva, M. Pantoja, J. C., and A. Klautau, "Modelos de linguagem n-grama para reconhecimento de voz com grande vocabulário," in *TIL2004 - III Workshop em Tecnologia da Informação e da Linguagem Humana*, 2004.

[27] J. PICONE, "Ece 8463: Fundamentals of speech recognition." [Online]. Available: http://www.ece.msstate.edu/research/isip/publications/courses/ece_8463/

[28] F. Jelinek, "Continuous speech recognition by statistical methods," in *Proceedings. of. the IEEE, vol. 64 no. 4*, 1976, pp. 532–555.

[29] T. Vintsyuk, "Speech discrimination by dynamic programming," *Kibernetika (Cybernetics)*, vol. 4, pp. 81–88, 1968.

[30] S. Young, N. Russell, and J. Thornton, "Token passing: a conceptual model for connected speech recognition systems," 1989. [Online]. Available: svr-ftp.eng.cam.ac.uk

[31] I. P. Association, *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge: Cambridge University Press. ISBN 0-521-65236-7 (hb) ISBN 0-521-63751-1 (pb), 1999.

[32] "Advancing human language technology in Brazil and the United states through collaborative research on portuguese spoken language systems," Federal University of Rio Grande do Sul, University of Caxias do Sul, Colorado University, and Oregon Graduate Institute, 2001.

[33] T. Lander, R. Cole, B. Oshika, and M. Noel, "The ogi 22 language telephone speech corpus," *In: Proc. Eurospeech'95, Madrid*, 1995.

[34] "http://acdc.linguateca.pt/cetenfolha/," Visited in January, 2008.

[35] A. Stolcke, "Srilm - an extensible language modeling toolkit," *Proc. Intl. Conf. Spoken Language Processing, Denver, Colorado*, 2002.

[36] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

[37] C. Hosn, L. A. N. Baptista, T. Imbiriba, and A. Klautau, "New resources for brazilian portuguese: Results for grapheme-to-phoneme and phone classification," *In VI International Telecommunications Symposium, Fortaleza*, 2006.

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

[39] T. J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce english text," *Complex Systems*, vol. vol. 1, pp. 145–168, 1987.

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

[41] "ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries," Visited in March, 2008.

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

[43] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1995.

[44] "http://www.speech.cs.cmu.edu/sphinx/," Visited in March, 2008.

[45] "http://www.isip.msstate.edu," Visited in March, 2008.

[46] "http://www.cstr.ed.ac.uk/projects/festival," Visited in March, 2008.

Apêndice A

Tabela SAMPA para o PB

Consoantes		
Oclusivas		
Símbolo	**Palavra**	**Transcrição**
p	pai	paj
b	barco	"barku
t	tenho	"teJu
d	doce	"dos@
k	com	ko~
g	grande	"gr6nd@
Fricativas		
f	falo	"falu
v	verde	"verd@
s	céu	sEw
z	casa	"kaz6
S	chapéu	S6"pEw
Z	jóia	"ZOj6
Nasais		
m	mar	mar
n	nada	"nad6
J	vinho	"viJu
Líquidas		
l	lanche	"l6nS@
L	trabalho	tr6"baLu
r	caro	"karu
R	rua	"Ru6

Vogais e ditongos				
Símbolo	**Palavra**	**Transcrição**	**Palavra**	**Transcrição**
i	vinte	"vint@	lápis	"lapiS
e	fazer	f6"zer		
E	belo	"bElu		
a	falo	"falu		
6	cama	"k6m6	madeira	m6"d6jr6
O	ontem	"Ont6~j~		
o	lobo	"lobu		
u	jus	ZuS	futuro	fu"turu
@	felizes	f@"liz@S		
i~	fim	fi~		
e~	emprego	e~"pregu (ou em-)		
6~	irmã	ir"m6~		
o~	bom	bo~		
u~	um	u~		
aw	mau	maw etc.: iw, ew, Ew, (ow)		
aj	mais	majS etc.: ej, Ej, Oj, oj,		
6~j~	têm	t6~j~ etc.: e~j~, o~j~, u~j~		

Outros símbolos	
"	Acento tónico primário (posto antes da sílaba tónica)
%	Acento tónico secundário (posto antes da sílaba tónica)
.	Separador silábico

Apêndice B

Árvore de decisão fonética

```
RO 100.0 "stats"

TR 0

QS  "R_V-Oral"      { *+a,*+e,*+i,*+o,*+u,*+O,*+E }
QS  "R_V-Front"     { *+i,*+E,*+e }
QS  "R_V-Central"   { *+a }
QS  "R_V-Back"      { *+u,*+o,*+O }
QS  "R_V-Aberta"    { *+a,*+E,*+O }
QS  "R_V-Fechada"   { *+i,*+e,*+o,*+u }
QS  "R_V-Nasal"     { *+a~,*+e~,*+i~,*+o~,*+u~ }
QS  "R_Oral"                    { *+p,*+b,*+f,*+v,*+t,*+d,*+s,*+z,*+l,*+r,*+S,*+Z,*+L,*+k,*+g,*+R,*+h }
QS  "R_Nasal"                   { *+n,*+m,*+J }
QS  "R_Oclusiva"                { *+p,*+b,*+t,*+d,*+k,*+g,*+n,*+m,*+J }
QS  "R_Sonora"                  { *+b,*+d,*+g,*+v,*+z,*+Z,*+r,*+R,*+n,*+m,*+J,*+h }
QS  "R_Surda"                   { *+p,*+t,*+k,*+f,*+s,*+S }
QS  "R_Oral_Oclusiva"           { *+p,*+b,*+t,*+d,*+k,*+g }
QS  "R_Oral_Oclusiva_Sonora"    { *+b,*+d,*+g }
QS  "R_Oral_Oclusiva_Surda"     { *+p,*+t,*+k }
QS  "R_Oral_Construtiva"        { *+f,*+v,*+s,*+z,*+l,*+r,*+S,*+Z,*+L,*+R,*+h }
QS  "R_Fricativa"               { *+f,*+v,*+s,*+z,*+S,*+Z }
QS  "R_Fricativa_Sonora"        { *+v,*+z,*+Z }
QS  "R_Fricativa_Surda"         { *+f,*+s,*+S }
QS  "R_Laterais"                { *+l,*+L }
QS  "R_Vibrantes"               { *+r,*+R,*+h }
QS  "R_Bilabiais"               { *+p,*+b,*+m }
QS  "R_Labiodentais"            { *+f,*+v }
QS  "R_Linguodentais"           { *+t,*+d,*+n }
QS  "R_Alveolares"              { *+s,*+z,*+l,*+r }
QS  "R_Palatais"                { *+S,*+Z,*+L,*+J }
QS  "R_Velares"                 { *+k,*+g,*+R,*+h }
QS  "R_a"       { *+a }
QS  "R_b"       { *+b }
QS  "R_k"       { *+k }
QS  "R_t"       { *+t }
QS  "R_i"       { *+i }
QS  "R_s"       { *+s }
QS  "R_S"       { *+S }
QS  "R_d"       { *+d }
QS  "R_f"       { *+f }
QS  "R_u"       { *+u }
QS  "R_m"       { *+m }
QS  "R_e~"      { *+e~ }
QS  "R_h"       { *+h }
QS  "R_Z"       { *+Z }
QS  "R_l"       { *+l }
```

```
QS  "R_n"       { *+n }
QS  "R_a~"      { *+a~ }
QS  "R_o"       { *+o }
QS  "R_R"       { *+R }
QS  "R_e"       { *+e }
QS  "R_O"       { *+O }
QS  "R_r"       { *+r }
QS  "R_L"       { *+L }
QS  "R_E"       { *+E }
QS  "R_z"       { *+z }
QS  "R_g"       { *+g }
QS  "R_J"       { *+J }
QS  "R_v"       { *+v }
QS  "R_p"       { *+p }
QS  "R_u~"      { *+u~ }
QS  "R_o~"      { *+o~ }
QS  "R_i~"      { *+i~ }
QS  "L_V-Oral"      { a-*,e-*,i-*,o-*,u-*,O-*,E-* }
QS  "L_V-Front"     { i-*,E-*,e-* }
QS  "L_V-Central"   { a-* }
QS  "L_V-Back"      { u-*,o-*,O-* }
QS  "L_V-Aberta"    { a-*,E-*,O-* }
QS  "L_V-Fechada"   { i-*,e-*,o-*,u-* }
QS  "L_V-Nasal"     { a~-*,e~-*,i~-*,o~-*,u~-* }
QS  "L_Oral"                    { p-*,b-*,f-*,v-*,t-*,d-*,s-*,z-*,l-*,r-*,S-*,Z-*,L-*,k-*,g-*,R-*,h-* }
QS  "L_Nasal"                   { n-*,m-*,J-* }
QS  "L_Oclusiva"                { p-*,b-*,t-*,d-*,k-*,g-*,n-*,m-*,J-* }
QS  "L_Sonora"                  { b-*,d-*,g-*,v-*,z-*,Z-*,r-*,R-*,n-*,m-*,J-*,h-* }
QS  "L_Surda"                   { p-*,t-*,k-*,f-*,s-*,S-* }
QS  "L_Oral_Oclusiva"           { p-*,b-*,t-*,d-*,k-*,g-* }
QS  "L_Oral_Oclusiva_Sonora"    { b-*,d-*,g-* }
QS  "L_Oral_Oclusiva_Surda"     { p-*,t-*,k-* }
QS  "L_Oral_Construtiva"        { f-*,v-*,s-*,z-*,l-*,r-*,S-*,Z-*,L-*,R-*,h-* }
QS  "L_Fricativa"               { f-*,v-*,s-*,z-*,S-*,Z-* }
QS  "L_Fricativa_Sonora"        { v-*,z-*,Z-* }
QS  "L_Fricativa_Surda"         { f-*,s-*,S-* }
QS  "L_Laterais"                { l-*,L-* }
QS  "L_Vibrantes"               { r-*,R-*,h-* }
QS  "L_Bilabiais"               { p-*,b-*,m-* }
QS  "L_Labiodentais"            { f-*,v-* }
QS  "L_Linguodentais"           { t-*,d-*,n-* }
QS  "L_Alveolares"              { s-*,z-*,l-*,r-* }
QS  "L_Palatais"                { S-*,Z-*,L-*,J-* }
QS  "L_Velares"                 { k-*,g-*,R-*,h-* }
QS  "L_a"       { a-* }
QS  "L_b"       { b-* }
QS  "L_k"       { k-* }
QS  "L_t"       { t-* }
QS  "L_i"       { i-* }
QS  "L_s"       { s-* }
QS  "L_S"       { S-* }
QS  "L_d"       { d-* }
QS  "L_f"       { f-* }
QS  "L_u"       { u-* }
QS  "L_m"       { m-* }
QS  "L_e~"      { e~-* }
QS  "L_h"       { h-* }
QS  "L_Z"       { Z-* }
QS  "L_l"       { l-* }
QS  "L_n"       { n-* }
QS  "L_a~"      { a~-* }
QS  "L_o"       { o-* }
QS  "L_R"       { R-* }
QS  "L_e"       { e-* }
QS  "L_O"       { O-* }
QS  "L_r"       { r-* }
```

```
QS  "L_L"       { L-* }
QS  "L_E"       { E-* }
QS  "L_z"       { z-* }
QS  "L_g"       { g-* }
QS  "L_J"       { J-* }
QS  "L_v"       { v-* }
QS  "L_p"       { p-* }
QS  "L_u~"      { u~-* }
QS  "L_o~"      { o~-* }
QS  "L_i~"      { i~-* }

TR 2

TB  600.0 "sp_ST2" {(*-sp,sp+*,*-sp+*).state[2]}
TB  600.0 "sil_ST2" {(*-sil,sil+*,*-sil+*).state[2]}
TB  600.0 "sil_ST3" {(*-sil,sil+*,*-sil+*).state[3]}
TB  600.0 "sil_ST4" {(*-sil,sil+*,*-sil+*).state[4]}
TB  600.0 "a~_ST2" {(*-a~,a~+*,*-a~+*).state[2]}
TB  600.0 "a~_ST3" {(*-a~,a~+*,*-a~+*).state[3]}
TB  600.0 "a~_ST4" {(*-a~,a~+*,*-a~+*).state[4]}
TB  600.0 "Z_ST2" {(*-Z,Z+*,*-Z+*).state[2]}
TB  600.0 "Z_ST3" {(*-Z,Z+*,*-Z+*).state[3]}
TB  600.0 "Z_ST4" {(*-Z,Z+*,*-Z+*).state[4]}
TB  600.0 "S_ST2" {(*-S,S+*,*-S+*).state[2]}
TB  600.0 "S_ST3" {(*-S,S+*,*-S+*).state[3]}
TB  600.0 "S_ST4" {(*-S,S+*,*-S+*).state[4]}
TB  600.0 "R_ST2" {(*-R,R+*,*-R+*).state[2]}
TB  600.0 "R_ST3" {(*-R,R+*,*-R+*).state[3]}
TB  600.0 "R_ST4" {(*-R,R+*,*-R+*).state[4]}
                .
                .
                .
TB  600.0 "b_ST4" {(*-b,b+*,*-b+*).state[4]}
TB  600.0 "a_ST2" {(*-a,a+*,*-a+*).state[2]}
TB  600.0 "a_ST3" {(*-a,a+*,*-a+*).state[3]}
TB  600.0 "a_ST4" {(*-a,a+*,*-a+*).state[4]}

TR 1

CO "tiedlist"
```

Apêndice C

Script para criação do Modelo Acústico

```
# Arquivos iniciais: wav_train.list, wav_test.list e Txts.list

### Monofones ###

# Criando lista de arquivos "mfcc" ...
cat wav_train.list | perl -e 'while (<>) { chomp; s/\.wav$//; $x=$_; print "$x.wav $x.mfc\n"; }' > wav_mfc_train.list
cat wav_test.list | perl -e 'while (<>) { chomp; s/\.wav$//; $x=$_; print "$x.wav $x.mfc\n"; }' > wav_mfc_test.list

# Extraindo parametros
HCopy -A -C confs/hcopy-avite.conf -S wav_mfc_train.list
HCopy -A -C confs/hcopy-avite.conf -S wav_mfc_test.list

mkdir hmms0

# Criando arquivo words.mlf
java -jar scripts/TxtToMLF.jar Txts.list

# Criando arquivo phones.mlf e phonesp.mlf
HLEd -l '*' -d dictionary.dic -i phones0.mlf confs/mkphones.led words.mlf
HLEd -l '*' -d dictionary.dic -i phonesp.mlf confs/mkphones2.led words.mlf

# renomeia os arquivos da lista de ".wav" para ".mfc"
cat wav_train.list | sed s/wav/mfc/g > mfc_train.list
cat wav_test.list | sed s/wav/mfc/g > mfc_test.list

# Criando um prototipo de HMM com 5 estados usando 1 Gauss/Mix de dimensao 39.
java -cp . ufpa.curupira.scripts.htk.CreateHMMPrototype 5 1 39 MFCC_0_D_A proto > hmms0/proto.hmm

# Computando Média e Variâncias globais
HCompV -C confs/hcomp.conf -f 0.01 -m -S mfc_train.list -M hmms0 hmms0/proto.hmm

# Criando hmms para cada fone...
./scripts/make-proto.sh

# Criando hmms.mlf
echo "" > concatenade.hed
HHEd -w hmms.mlf -d hmms0/ concatenade.hed hmmlist.txt

# Criando hmmdefs e macros
x=`wc -l hmms.mlf | awk '{print $1}' `
let y="$x - 3"
tail -n $y  hmms.mlf > temp
mv temp ./hmms0/hmmdefs
head -n 3 hmms.mlf > hmms0/temp
cat hmms0/temp hmms0/vFloors > hmms0/macros
```

```
mkdir hmms1
# HERest - 1a reestimação
HERest -I phones0.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms0/macros -H hmms0/hmmdefs -M hmms1 hmmlist.txt

mkdir hmms2
# HERest - 2a reestimação
HERest -I phones0.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms1/macros -H hmms1/hmmdefs -M hmms2 hmmlist.txt

mkdir hmms3
# HERest - 3a reestimação
HERest -I phones0.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms2/macros -H hmms2/hmmdefs -M hmms3 hmmlist.txt

# Adicionando short-pause (sp)
mkdir hmms4
cp hmms3/* hmms4
mkdir hmms5
echo "sp" >> hmmlist.txt
java -jar scripts/Makesp.jar hmms4/hmmdefs
HHEd -H hmms4/macros -H hmms4/hmmdefs -M hmms5/ util/sil.hed hmmlist.txt

mkdir hmms6
# HERest - 4a reestimação
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms5/macros -H hmms5/hmmdefs -M hmms6 hmmlist.txt

...

mkdir hmms14
# HERest - 12a reestimação
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms13/macros -H hmms13/hmmdefs -M hmms14 hmmlist.txt

### Trifones ###

# Cross-Word triphones
#HLEd -n trifone -l '*' -i wintri.mlf util/mkCrossWord.led phonesp.mlf

# Word-Internal triphones
HLEd -n trifone -l '*' -i wintri.mlf util/mktri.led phonesp.mlf

cp util/alltrifones.list trifone

# Clonando Trifones
mkdir hmms18
perl scripts/MAmktie.perl
HHEd -B -H hmms14/macros -H hmms14/hmmdefs -M hmms18 mktri.hed hmmlist.txt
rm mktri.hed

# Reestimando trifones
mkdir hmms19
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S mfc_train.list -H hmms18/macros -H hmms18/hmmdefs -M hmms19 trifone

...

# Reestimando trifones
mkdir hmms22
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S mfc_train.list -H hmms21/macros -H hmms21/hmmdefs -M hmms22 trifone

### Tying de estados por Árvore de Decisão ###

mkdir hmmsTree

# Copia a árvore de decisão
cp util/bestTree.hed tree.hed

# Fazendo Tying dos estados
HHEd -B -H hmms22/macros -H hmms22/hmmdefs -M hmmsTree tree.hed trifone
```

```
# Re-estimando
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0  -S mfc_train.list -H hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0  -S mfc_train.list -H hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0  -S mfc_train.list -H hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0  -S mfc_train.list -H hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist

### Incrementando o número de componentes Gaussianas ###

# Incrementando --> 2 Gauss/mix
mkdir 2G
HHEd -H hmmsTree/macros -H hmmsTree/hmmdefs  -M 2G util/mix2.hed tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H 2G/macros -H 2G/hmmdefs -M 2G tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H 2G/macros -H 2G/hmmdefs -M 2G tiedlist

# Incrementando --> 4 Gauss/mix
mkdir 4G
HHEd -H 2G/macros -H 2G/hmmdefs -M 4G util/mix4.hed tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H 4G/macros -H 4G/hmmdefs -M 4G tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H 4G/macros -H 4G/hmmdefs -M 4G tiedlist

...

# Incrementando --> 14 Gauss/mix
mkdir 14G
HHEd -H 12G/macros -H 12G/hmmdefs -M 14G util/mix14.hed tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H 14G/macros -H 14G/hmmdefs -M 14G tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H 14G/macros -H 14G/hmmdefs -M 14G tiedlist
```

Apêndice D

Script para criação do Modelo de Linguagem

```
# Inicialização das variáveis
TRAIN=databases/train.xml  # corpus de treino
TEST=databases/test.xml    # corpus de teste
VOC=60k.voc                # vocabulário do ML
N=2                        # valor de "N" no N-gram

cp databases/vocs/"$VOC" .

# Conta as ocorrências de todos os n-gramas de ordem N e realiza o smoothing
ngram-count -kndiscount -interpolate -order $N -text $TRAIN -vocab $VOC -lm bigram

# Calcula a perplexidade do ML
ngram -lm bigram -ppl $TEST

# Cria a Rede
HBuild -A -T 1 -s "<s>" "</s>" -n bigram $VOC network_"$VOC"
```