PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Programa de Pós-graduação em Engenharia Elétrica

Willian Gomes de Almeida

RECONHECIMENTO AUTOMÁTICO DE FALA CONTÍNUA COM OTIMIZAÇÃO DOS PARÂMETROS DO DECODIFICADOR ATRAVÉS DOS ALGORITMOS GENÉTICOS MULTIOBJETIVOS

Belo Horizonte
2015

Willian Gomes de Almeida

RECONHECIMENTO AUTOMÁTICO DE FALA CONTÍNUA COM OTIMIZAÇÃO DOS PARÂMETROS DO DECODIFICADOR ATRAVÉS DOS ALGORITMOS GENÉTICOS MULTIOBJETIVOS

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Profr[a]. Dr[a]. Flávia Magalhães Freitas Ferreira
Coorientador: Profr[a]. Dr[a]. Zélia Myriam Assis Peixoto

Belo Horizonte
2015

FICHA CATALOGRÁFICA
Elaborada pela Biblioteca da Pontifícia Universidade Católica de Minas Gerais

Almeida, Willian Gomes de
A447r Reconhecimento automático de fala contínua com otimização dos parâmetros do decodificador através dos algoritmos genéticos multiobjectivos / Willian Gomes de Almeida. Belo Horizonte, 2015.
128 f. : il.

Orientadora: Flávia Magalhães Freitas Ferreira
Coorientadora: Zélia Myriam Assis Peixoto
Dissertação (Mestrado) – Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Engenharia Elétrica.

1. Reconhecimento automático da voz. 2. Processamento de sinais - Técnicas digitais. 3. Sistemas de reconhecimento de padrões. 4. Algoritmos genéticos. I. Ferreira, Flávia Magalhães Freitas. II. Peixoto, Zélia Myriam Assis. III. Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Engenharia Elétrica. IV. Título.

CDU: 621.391

Willian Gomes de Almeida

RECONHECIMENTO AUTOMÁTICO DE FALA CONTÍNUA COM OTIMIZAÇÃO DOS PARÂMETROS DO DECODIFICADOR ATRAVÉS DOS ALGORITMOS GENÉTICOS MULTIOBJETIVOS

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétricada Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do Título de Mestre em Engenharia Elétricado Programa de Pós-graduação em Engenharia Elétrica.

Profr[a]. Dr[a]. Flávia Magalhães Freitas Ferreira - PUC Minas (Orientador)

Prof. Dr. Abraham Alcaim - PUC Rio (Banca Examinadora)

Prof. Dr. Gustavo Luíz Soares - PUC Minas (Banca Examinadora)

Belo Horizonte, 22 de Junho de 2015

Aos meus pais, Maria dos Anjos e João Luiz irmãos, Ed Carlos e Ednaldo e sobrinhas, Jennifer e Pamella

AGRADECIMENTOS

À minha querida professora Flávia pela orientação, incentivo e paciência.

À professora Zélia pela orientação e amizade.

Ao professor Abraham por me ter introduzido ao tema de reconhecimento automático de voz e pelos recursos gentilmente disponibilizados.

Ao Willian Antônio e Harison Silva grande apoio no desenvolvimento do trabalho.

Aos meus pais, irmãos, avós, tios e primos por todo apoio, carinho, paciência e confiança sobre todas as escolhas que fiz durante minha trajetória acadêmica e pessoal.

À Isabel Siqueira e Isabel Novais, por serem tão amáveis e gentis.

Ao Alan Moreira pelo imprescindível auxilio e suporte técnico, os quais foram decisivos para a conclusão do trabalho.

Aos meus queridos amigos Virgínia, Camila e Amarildo, pelo carinho e atenção mesmo quando estive distante.

À Whirlpool Latim America, FAPEMIG e PUC Minas, por terem proporcionado a fantástica experiência de participar do time do desenvolvimento de produtos em uma empresa de reconhecimento internacional, a partir do programa Talentos Mineiros.

E a todos que contribuíram de forma direta ou indireta no desenvolvimento desse trabalho.

"A leitura após certa idade distrai excessivamente o espírito humano das suas reflexões criadoras. Todo o homem que lê de mais e usa o cérebro de menos adquire a preguiça de pensar." Albert Einstein.

RESUMO

No presente trabalho será apresentada uma abordagem para a otimização dos parâmetros do decodificar AVite utilizando o algoritmo genético NSGA-II. Foram construídos modelos acústicos para reconhecimento de voz contínuo e dependente do locutor, utilizando as ferramentas de código aberto contidas no HTK. Foi também construído um modelo de linguagem n-grama, utilizando as ferramentas do pacote SRILM. O método de otimização dos parâmetros do decodificador empregado, utiliza somente 20 sentenças para a avaliação da função de aptidão dos indivíduos e foram conseguidos bons resultados utilizando somente 10 gerações. Com o método proposto foi possível obter uma solução que apresenta redução no RTF de até 68,7% sem degradação do WER quando comparado com os resultados obtidos com um conjunto de parâmetros encontrados na literatura e até uma redução no WER de até 9,5% com RTF melhor.

Palavras-chave: HMM. NSGA-II. Reconhecimento Automático de Fala. Otimização Multiobjetivo.

ABSTRACT

An approach to optimization of parameters AVite decode using genetic algorithm NSGA-II is presented in this work. Acoustic models were built for speaker independent continuous speech recognition, using the open source tools contained in HTK. Was also builded a model of n-gram language using the tools of the SRILM package. The method of optimizing the parameters of the decoder uses only 20 sentences used for evaluating the fitness function of the individuals and good results were achieved using only 10 generations. With the proposed method it was possible to obtain a solution which showed a RTF reduction of up to 68,7% in without degradation of the WER when compared with the results obtained with a set of parameters found in the literature and to a WER reduction of up to 9.5% better than that obtained with parameters founded in the literature with better RTF.

Keysords: HMM. NSGA-II. Automatic Speech Recognition. Multiobjective Optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE GRÁFICOS

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de reconhecimento automático de fala (Automatic Speech Recognition - ASR) têm se tornado cada vez mais presentes no cotidiano das pessoas. Já é possível encontrar tais sistemas em dispositivos como telefones celulares, aparelhos de televisão, eletrodomésticos, computadores, entre outros. As aplicações, entretanto, diferem-se com relação aos requisitos de desempenho, tais como taxa de erro e velocidade de processamento, que podem ser quantificadas através de métricas apropriadas, sendo as mais utilizadas: a taxa de erro do reconhecimento (*Word Error Rate* – WER) e o fator de tempo real (*Real Time Factor* - RTF) (Huang, Acero, Hon, et al., 2001). Sistemas que utilizam comando por voz para o controle de eletrodomésticos ou máquinas industriais devem ter o WER extremamente baixo, pois a ativação indevida de algumas funções pode trazer uma condição de risco ao usuário. Por outro lado, algumas aplicações exigem prioritariamente um baixo RTF (resposta mais rápida), permitindo para isso sacrificar o WER. Em sistemas de transcrição de chamadas telefônicas, por exemplo, todas as chamadas gravadas durante um dia de operação devem ser processadas no período da noite, de forma que o sistema esteja disponível para operação no dia seguinte (Satoshi et al., 2012). Desse modo, é necessário entender as características da aplicação para a correta definição do ponto ótimo de ajuste do ASR.

O módulo dos sistemas de reconhecimento automático de voz responsável pela conversão do áudio para texto é chamado decodificador. Nos decodificadores de voz do estado-da-arte, é comum encontrar uma quantidade razoável de parâmetros, os quais devem ser ajustados em função do desempenho requerido no processo de reconhecimento. De forma geral são encontradas basicamente três categorias de parâmetros: parâmetros relacionados ao processo de detecção dos trechos voz/silêncio; parâmetros relacionados à redução do espaço de busca e parâmetros relacionados ao modelo de linguagem.

Os algoritmos para detecção de voz/silêncio são utilizados para remover os trechos de silêncio no início e fim das palavras ou sentenças pronunciadas, enviando para o decodificador somente o trecho que contém a palavra pronunciada. Com a eliminação dos

trechos de silêncio é possível reduzir a carga computacional necessária para o reconhecimento das sentenças, e portanto o consumo de potência, o que é muito importante quando se consideram sistemas embarcados alimentados por baterias. Além disso, também pode proporcionar uma redução do WER.

O processo de busca pela melhor sequência de palavras envolve a enumeração de todas as possíveis sequências de palavras, e consequentemente todas as possíveis sequências de estados, o que caracteriza o espaço de busca. Para a determinação da melhor sequência de palavras, o decodificador deve então buscar a sequência de estados que forneça a maior probabilidade ou verossimilhança para a sentença em avaliação. O algoritmo mais utilizado para a busca da melhor sequência de estados no reconhecimento de discurso contínuo é o time *synchronous Viterbi beam search* (Ney & Ortmanns, 1999). Nesse algoritmo o espaço de busca é construído sincronizadamente com o tempo e são utilizadas heurísticas para reduzir o espaço de busca, eliminando hipóteses com probabilidade abaixo de um limiar. Dessa forma não é necessário avaliar o espaço de busca completo, reduzindo assim o RTF do sistema. Tais heurísticas constituem a categoria de parâmetros responsável pela redução do espaço de busca.

Quando é empregado um modelo de linguagem no sistema de reconhecimento automático de voz, é uma prática comum o ajuste das probabilidades relacionadas ao modelo de linguagem, a fim de aumentar a taxa de acerto. Tal ajuste é realizado pela aplicação de fatores, os quais são destinados à redução de erros causados, principalmente, pela inserção indevida de palavras e pela tendência do decodificador fornecer resultados constituídos de poucas palavras longas ou muitas palavras curtas. Os parâmetros do decodificador responsáveis por esses ajustes estão na categoria dos parâmetros relacionados ao modelo de linguagem.

Os parâmetros citados diferem-se nos diversos decodificadores e têm grande dependência dos modelos utilizados, de forma que os melhores resultados devem ser obtidos encontrando-se a melhor configuração dos parâmetros para cada configuração de modelos. Em muitos casos esses parâmetros são ajustados manualmente através da experiência do pesquisador, ou automaticamente através de uma busca em grade (algoritmo de busca

em força bruta, dentro de faixas de valores com resoluções especificadas para cada um dos parâmetros). O ajuste manual não é uma tarefa fácil, uma vez que os parâmetros podem estar correlacionados, de forma que o ponto ótimo para um parâmetro pode estar condicionado ao valor de outro parâmetro. A busca em grade nem sempre é eficiente, uma vez que é necessária a definição da faixa de variação dos valores de cada parâmetro e a precisão do resultado estará condicionada ao número de possibilidades para cada parâmetro. Dessa forma, o problema motivador desse trabalho é a determinação de um método capaz de estimar automaticamente, e de forma eficiente, os parâmetros do decodificador, a fim de encontrar soluções que minimizem o WER e o RTF simultaneamente.

Kacur e Korosi (2007) utilizaram a abordagem dos algoritmos genéticos mono-objetivo para escolher os valores de um conjunto de parâmetros do decodificador AVite (*The ATK Real-Time API for HTK*, 2014), contido no pacote ATK, de forma a minimizar apenas o WER. Foram escolhidos pelos autores somente os parâmetros do decodificador relacionados ao algoritmo de detecção de voz/silêncio, além de um único parâmetro relacionado ao modelo de linguagem, chamado *insertion penalty*. A otimização dos parâmetros proporcionou uma redução relativa de 18,6% no WER, que passou de 23% para 18,7%.

El Hannani e Hain (2010), por sua vez, estavam interessados em determinar os valores para um conjunto de parâmetros relacionados à redução do espaço de busca, de forma a minimizar o WER para qualquer valor de RTF. Para isso foi utilizado um processo iterativo, onde o ponto de partida foi definido a partir da medição do RTF quando nenhuma heurística é utilizada para reduzir o espaço de busca. Nesse ponto, o RTF é máximo, ou seja, trata-se da pior situação em termos de velocidade de processamento. A cada iteração do algoritmo, são determinados vários conjuntos de parâmetros, que eram definidos de forma a levar o sistema a um RTF menor que o RTF da iteração anterior. Então um dos conjuntos de parâmetros é escolhido de forma a minimizar uma função de custo, a qual está relacionada ao WER. Dessa forma é construída uma curva WERxRTF buscando o melhor WER para cada RTF encontrado. Observa-se, entretanto, que RTF e WER não são minimizados simultaneamente pelo algoritmo.

Como proposta de solução ao problema motivador, neste trabalho serão utilizados

algoritmos genéticos multiobjetivo para encontrar conjuntos de parâmetros que minimizam o RTF e o WER simultaneamente, no reconhecimento de fala contínua. Serão otimizados somente os parâmetros relacionados à redução do espaço de busca e aos modelos de linguagem. Não serão otimizados os parâmetros relacionados à detecção voz/silêncio, pois os mesmos fazem pouco sentido no cenário de reconhecimento de discurso contínuo, uma vez que os trechos de silêncio podem ser modelados como unidades fonéticas.

O escopo do projeto consiste dos seguintes itens:

a) levantamento bibliográfico acerca do estado da arte da tecnologia em reconhecimento de discurso contínuo para português brasileiro;

b) construção de modelos acústicos utilizando as ferramentas de código aberto contidos no HTK;

c) construção de modelos de linguagem utilizando ferramentas de código aberto contidas no SRILM;

d) avaliação dos modelos construídos utilizando o decodificador de voz de código aberto AVite;

e) levantamento bibliográfico acerca dos métodos utilizados para otimização automática dos parâmetros de um decodificador;

f) levantamento bibliográfico acerca da otimização multiobjetivo;

g) implementação de um sistema para otimização dos parâmetros do decodificador AVite;

h) verificação da robustez dos resultados encontrados.

Este trabalho está organizado como se segue: No Capítulo 2 será feita uma revisão bibliográfica acerca dos fundamentos e da tecnologia em reconhecimento automático de fala, abordando a extração de atributos acústicos, a modelagem acústica utilizando modelos estocásticos, os modelos de linguagem e as considerações práticas que devem ser observadas para a implementação de um sistema reconhecimento automático de fala. No Capítulo 3 serão apresentados os conceitos da otimização multiobjetivo. Serão definidos os

conceitos de dominância e fronteira de Pareto e serão apresentados os fundamentos dos algoritmos genéticos multiobjetivo, dando especial atenção ao algoritmo genético NSGA-II. No Capítulo 4 serão apresentados os trabalhos correlatos e no Capítulo 5 serão apresentadas as ferramentas utilizadas para a modelagem acústica e de linguagem, bem como para a construção do decodificador que será utilizado nos experimentos. No Capítulo 6 serão apresentados os detalhes do modelo acústico e de linguagem construídos para o experimento, bem como o método proposto para a otimização dos parâmetros do decodificador, seguindo-se uma discussão sobre os resultados alcançados. No Capítulo 7 serão avaliados a influência do número de gerações e do número de Gaussianas sobre o desempenho do ASR cujo decodificador foi otimizado utilizando-se a proposta desta pesquisa e finalmente, no Capítulo 8 serão apresentadas as conclusões do trabalho.

2 SISTEMAS DE RECONHECIMENTO AUTOMÁTICO DE FALA

Neste capítulo será apresentada a teoria básica do ASR. Será apresentado um pequeno histórico sobre a evolução dos sistemas de reconhecimento automático de voz, seguido de uma breve discussão acerca dos tipos de ASR. Em seguida, serão apresentados os principais atributos acústicos utilizados para representar um sinal de voz em ASR, seguido dos fundamentos da modelagem acústica utilizando os modelos ocultos de Markov e os fundamentos dos modelos de linguagem n-grama. No final do capítulo serão apresentados alguns detalhes práticos que devem ser observados quando da construção de um sistema para reconhecimento de discurso contínuo.

2.1 Breve histórico dos sistemas de reconhecimento automático de fala

A voz é o principal meio de comunicação entre pessoas. A pesquisa em reconhecimento automático de fala tem atraído grande atenção nas últimas seis décadas, por razões que vão desde o interesse científico pelos mecanismos mecânicos no trato vocal para a produção da voz, até o desejo de automatizar tarefas simples, que inerentemente requerem interação entre homem e máquina.

A tecnologia de ASR também tem despertado interesse da população em geral, se tornando popular a partir de sucessos do cinema das décadas de 1960 e 1970, como o longametragem “2001: Uma odisséia no espaço”. Nesse filme, um computador inteligente chamado “HAL” fala em linguagem sonora natural e é capaz não somente de reconhecer e entender fala fluente, como também de responder de forma adequada. Outro exemplo pode ser encontrado na famosa saga “Star Wars”, onde George Lucas estende a capacidade de inteligência das máquinas, criando então os andróides “R2D2” e “C3PO”, que eram capazes de falar e entender discursos naturais, bem como mover-se e interagir com o ambiente ao seu redor, com outros androides e com a população em geral.

No ano de 1952, Davis, Biddulph e Balashek, dos Laboratórios Bell, criaram um dos

primeiros sistemas para reconhecimento de dígitos isolados pronunciados por um locutor único. O sistema criado foi fundamentado no princípio da fonético-acústica e os dígitos eram reconhecidos utilizando as "frequências formantes", que eram medidas em segmentos (trechos) de voz que correspondiam às vogais presentes em cada um dos dígitos. Essas frequências medidas eram então utilizadas como padrões de referência para a identificação da elocução de um dígito desconhecido. Ainda dentre os primeiros ASRs propostos na década de 1950, Olson e Belar, dos Laboratórios RCA, construíram um sistema para reconhecer 10 sílabas de um único locutor e Forgie, do MIT Lincols Lab, construiu um sistema independente de locutor para o reconhecimento de 10 vogais.

Na década de 1970, Atal e Itakura, independentemente, formularam os conceitos fundamentais da codificação preditiva linear (Linear Predictive Coding – LPC)(Atal & Hanauer, 1971), que simplificou enormemente a estimação da resposta do trato vocal a partir da forma de onda de voz. Em meados de 1970, as ideias de aplicar os conceitos fundamentais da tecnologia de reconhecimento de padrões para reconhecimento de voz, baseado em método LPC, foram propostas por Itakura (Itakura, 1975), Rabiner e Levinson (Rabiner, Levinson, Rosenberg, & Wilpon, 1979) e outros. Na década de 80, devido ao rápido desenvolvimento dos métodos baseados em modelagem estocástica, sendo o modelo oculto de Markov (Hidden Markov Model – HMM) a abordagem de maior destaque, causou certo grau de convergência dos sistemas. Hoje em dia, segundo (Gaikwad, Gawali, & Yannawar, 2010) e (Prabhakar & Sahu, 2013), a maioria dos sistemas práticos de reconhecimento de fala são baseados em modelos estocásticos desenvolvidos na década de 1980 e que sofreram alguma melhoria na década de 1990.

Os sistemas atuais de ASR para reconhecimento de discurso contínuo utilizam o HMM para o modelamento de unidades fonéticas tais como monofones, trifones ou sílabas. Essas unidades são então utilizadas para construir palavras e sentenças. Alguns sistemas, sobretudo os sistemas para reconhecimento de palavras isoladas, geralmente utilizam as próprias palavras ou, até mesmo, sentenças inteiras como unidades fonéticas.

2.2 Tipos de sistemas de reconhecimento automático de fala

Os sistemas de ASR podem ser classificados quanto ao modo da fala, quanto ao tamanho do vocabulário a ser reconhecido e quanto à dependência ou independência de locutor (Alcaim & Oliveira, 2011).

Ao se considerar o modo de fala, há basicamente três tipos de sistemas. As diferenças são na maior parte conceituais, mas de forma geral, eles são classificados como sistemas dedicados ao reconhecimento de:

a) palavras isoladas: os sistemas reconhecem uma palavra dentre um grupo de palavras. É importante que exista uma pausa considerável entre as elocuções das palavras e, na maioria das vezes, utilizam-se as próprias palavras como unidades fonéticas modeladas, embora unidades menores possam ser utilizadas;

b) palavras conectadas: os sistemas buscam reconhecer uma sequência formada por um número fixo de palavras. Assim como no reconhecimento de palavras isoladas, geralmente utiliza-se um vocabulário reduzido e a palavra é utilizada como unidade fonética;

c) fala contínua: são sistemas mais complexos e difíceis de serem implementados. Precisam ser capazes de lidar com elocuções de comprimentos variáveis, palavras desconhecidas e variações de pronúncia. Exploram as relações estatísticas entre as palavras, os conhecimentos léxicos e sintáticos. Como precisam lidar com um número grande de palavras, geralmente utilizam unidades fonéticas menores que as palavras, de forma que as palavras sejam formadas a partir da concatenação dessas subunidades. Dentre essas unidades fonéticas destacam-se: os monofones, que são as menores unidades sonoras utilizadas na construção de palavras, os difones, que são constituídos pela união de dois monofones consecutivos, e os trifones, que são constituídos pela união de três monofones consecutivos. Ressalta-se que os difones e os trifones nada mais são do que monofones contextualizados, ou seja, modificados pelos monofones vizinhos.

O vocabulário de um ASR é o conjunto das possíveis palavras que o mesmo é capaz de reconhecer. No cenário de reconhecimento de palavras isoladas, o vocabulário

geralmente coincide com as unidades fonéticas cujos modelos são construídos. Já no reconhecimento de palavras conectadas e fala contínua, as palavras do vocabulário são formadas pela concatenação de unidades fonéticas menores. É fácil concluir que quanto maior o vocabulário do sistema, menor a taxa de acerto de reconhecimento. Isso se deve ao fato de que o sistema terá que escolher entre diversas alternativas e, com um número maior de alternativas, a probabilidade de erro é maior. Nos sistemas para reconhecimento de vocabulário reduzido, onde o dicionário tem até 100 palavras, é comum o uso de uma gramática de reconhecimento, ou seja, uma estrutura que define como as possíveis sentenças podem ser formadas utilizando as palavras do vocabulário, de forma a restringir o número de alternativas a serem avaliadas pelo decodificador, aumentando com isso a taxa de acerto. Já em um sistema para vocabulário extenso, onde o dicionário pode ter mais de 1.000 palavras, é comum utilizar um modelo estatístico que forneça a probabilidade apriori de que uma determinada palavra seja pronunciada depois de uma sequência de palavras já observada. Com isso, não é colocada nenhuma limitação sobre o que o locutor pode dizer, a não ser a limitação do número de palavras no vocabulário.

Quanto à dependência ou independência de locutor, há também três classificações básicas (Alcaim & Oliveira, 2011):

a) sistemas Dependentes do Locutor: são aqueles nos quais o reconhecimento é realizado apenas com os locutores que estão presentes na base de dados de treinamento. Apesar de poderem ser utilizados para reconhecer a fala de indivíduos que não estavam presentes na base de dados de treinamento, o desempenho será muito pior do que aquele para os indivíduos presentes na mesma;

b) sistemas Independentes do Locutor: são aqueles capazes de reconhecer, com um desempenho aceitável, a fala de locutores que não estavam presentes na base de dados de treinamento. Para serem construídos, são necessárias bases de dados constituídas da elocução de diversas sentenças, pronunciadas por diversos indivíduos;

c) sistemas Adaptados ao Locutor: são aqueles em que um sistema independente de locutor é ajustado de forma a melhorar o desempenho para um determinado locutor

que não estava presente na base de dados de treinamento. Para fazer esse ajuste, é necessária a gravação de um conjunto de dados que geralmente é bem menor do que a quantidade necessária para o treinamento de um modelo dependente de locutor. Por esse motivo, são os sistemas mais adequados à implementação comercial.

2.3 Extração de atributos de fala

Na maioria dos ASRs, são necessárias algumas suposições a respeito das características dos sinais de voz para que se proponha uma modelagem acústica. Uma das mais importantes é que o sinal pode ser considerado estacionário em um intervalo curto de tempo, o que, em outras palavras, que dizer que suas características espectrais são constantes nesse intervalo. No entanto, para que essa suposição possa ser satisfeita, é necessário segmentar o sinal de voz em blocos de curta duração. Empiricamente foi constatado que um tamanho adequado para esses blocos está em torno de 20 a 40ms. A fim de preservar as características temporais do sinal de fala na modelagem acústica, é necessária uma sobreposição entre blocos consecutivos, sendo que a quantidade geralmente utilizada de amostras sobrepostas fica em torno de 25% a 75% do total das amostras do bloco. A título de exemplo, suponha um sinal divido em blocos de 20ms e com sobreposição de 50%. Nesse caso, um novo bloco deve ser tomado a cada 10ms e terá 50% da informação do bloco anterior.

Como efeito dessa segmentação do sinal de voz em blocos, ocorre uma grande descontinuidade nas bordas de cada bloco, o que ocasiona um fenômeno espectral indesejável chamado fenômeno de Gibs. Para contornar esse problema, aplica-se uma janela a cada bloco, sendo que a mais utilizada é a janela Hamming (Huang et al., 2001). A janela Hamming tem o papel de suavizar as bordas de cada bloco, e é descrita pela Equação 1, sendo n a n-ésima amostra e T o número de amostras em um bloco de análise.

$$Hamming(n) = 0,54 - 0,46 \cos\left(\frac{2\pi(n-1)}{T-1}\right) \tag{1}$$

Uma operação comum a ser aplicada ao sinal janelado é uma filtragem chamada pré-ênfase. O filtro de pré-ênfase é um filtro passa-altas, ou seja, tem o efeito de atenuar as baixas frequências do sinal, de modo a compensar a atenuação das altas frequências do sinal de voz pelos lábios. A função de transferência do filtro de pré-ênfase mais utilizado é mostrada na Equação 2, onde o parâmetro é chamado coeficiente de pré-ênfase, assumindo normalmente o valor 0,97.

$$H(z) = 1 - \alpha z^{-1} \tag{2}$$

Uma vez realizados os processos de segmentação em blocos, janelamento e aplicação do filtro de pré-ênfase, são realizadas estimativas espectrais em cada bloco e então essas estimativas passam por uma série de transformações, até que sejam obtidos os atributos acústicos correspondentes a cada bloco. Os atributos acústicos têm por objetivo oferecer ao ASR uma representação compacta e consistente do sinal de voz, buscando evidenciar as características que discriminam os diferentes sons da fala, ao mesmo tempo em que elimina as informações menos relevantes. Dentre os atributos acústicos mais populares, citam-se os atributos oriundos da análise preditiva linear (Linear Predictive Coding - LPC) (Atal & Hanauer, 1971), os atributos mel cepestrais (Mel Frequency Cepstral Coeficients - MFCC)(Davis & Mermelstein, 1980) e os atributos perceptuais lineares preditivos (Perceptual Linear Predictive – PLP) (Hermansky, 1990). A seguir serão descritos os processos de obtenção dos atributos LPC e MFCC, por serem bastante utilizados e servirem de base para muitos outros atributos.

2.3.1 Atributos LPC

A ideia básica do modelo LPC é que a amostra de voz no instante de tempo n , aqui denominada $s(n)$, pode ser aproximada como a combinação linear de p amostras anteriores, tal que:

$$s(n) \approx \tilde{s}(n) = a_1 s(n-1) + a_2 s(n-2) + \ldots + a_p s(n-p) \tag{3}$$

onde os coeficientes $a_1, a_2, \ldots, a_p$ são considerados constantes em um bloco de análise (os pesos são determinados para cada bloco). A Equação 3 pode ser expressa de forma mais compacta e, considerando a presença de um sinal de excitação externo, como:

$$s(n) = \sum_{i=1}^{p} a_1 s(n-1) + Gu(n) \tag{4}$$

onde u é uma excitação normalizada e G é o ganho de excitação. Expressando a Equação 4 no domínio-z, obtém-se:

$$S(z) = \sum_{i=1}^{p} a_i z^{-i} S(z) + GU(z) \tag{5}$$

que leva à função de transferência:

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 - \sum_{i=1}^{p} a_1 z^{-i}} = \frac{1}{A(z)} \tag{6}$$

Considerando o sinal $\tilde{s}(n)$ descrito como uma combinação linear de amostras passadas, definido como:

$$\tilde{s}(n) = \sum_{i=1}^{p} a_i s(n-i) \tag{7}$$

o erro de predição, $e(n)$, pode ser definido como:

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{i=1}^{p} a_i s(n-i) \tag{8}$$

O problema básico da análise preditiva linear é determinar um conjunto de coeficientes a_i que minimiza o erro médio quadrático E (Atal & Hanauer, 1971):

$$E = \sum_{n} e^2(n) \tag{9}$$

que, se escrito em termos de $s(n)$, torna-se:

$$E = \sum_{n} \left[s(n) - \sum_{i=1}^{p} a_i s(n-i) \right]^2 \tag{10}$$

Para encontrar os coeficientes de predição que minimizam o erro médio quadrático expresso pela Equação 10 em cada bloco, é necessário derivar E em relação a cada a_k e igualar o resultado a zero, ou seja:

$$\frac{\partial E}{\partial a_i} = 0, \quad i = 1, 2, \ldots p \tag{11}$$

o que resulta em

$$\sum_{n} s(n-k)s(n) = \sum_{i=1}^{p} a_i \sum_{n} s(n-k)s(n-i) \tag{12}$$

Se o termo $\sum_{n} s(n-k)s(n-i)$ for expresso como

$$\phi(k,i) = \sum_{n} s(n-k)s(n-i) \tag{13}$$

a Equação 12 pode ser expressa usando a notação compacta

$$\phi(k,0) = \sum_{i=1}^{p} a_i \phi(k,i) \tag{14}$$

que é um conjunto de p equações e p variáveis, conhecido como equações de Yule-Walker (Huang et al., 2001).

Para se calcular os coeficientes de predição a_i , na Equação 14, deve-se calcular $\phi(k,i)$ para $1 \leq k \leq P$ e $0 \leq i \leq P$, e então resolver o sistema de equações lineares resultante. O conjunto de equações indicado pela Equação 14 pode ser resolvido por inversão de matriz ou por algum outro método para solução de equações lineares. No entanto, existem métodos eficientes, tais como os métodos da covariância e da autocorrelação (Huang et al., 2001). Esses métodos não serão descritos aqui, ficando a cargo do leitor consultar as referências para maiores informações.

2.3.2 Atributos MFCC

Os coeficientes mel-cepestrais (Mel Frequency Cepstral Coeficients – MFCC) (Davis & Mermelstein, 1980) surgiram devido a estudos da área da piscoacústica, que mostraram que a percepção auditiva humana de tons puros (frequências puras) do sinal de voz não segue uma escala linear. Dessa forma, existe uma escala que mapeia as frequências subjetivas dos tons puros, chamada de mel, na escala Hertz (Hz) de frequência. Para a determinação da relação entre a escala mel e a escala Hz, foi realizado um experimento onde foi definida a frequência de 1.000Hz, com 40dB SPL (Sound Pressure Level) acima do limiar de audição humana, como 1.000mels. Os outros valores foram encontrados pedindo-se que ouvintes ajustassem a frequência física (em Hz) de tom em tom, até que a frequência percebida fosse duas vezes a frequência de referência. Em seguida o mesmo experimento foi realizado para 10 vezes a frequência de referência, e assim por diante. Dessa forma, as frequências encontradas teriam valores de 2.000mels e 10.000mels, respectivamente. De forma semelhante, o processo é realizado pedindo-se que os ouvintes ajustem a frequência para metade e para um décimo da frequência de referência, obtendo-se dessa forma as frequências de 500mels e 100mels, respectivamente. Com esse mapeamento foi possível verificar que a relação entre a frequência em Hz e a frequência percebida é aproximadamente linear abaixo de 1.000Hz, mas que apresenta um comportamento logaritmico acima desse valor. A Equação 15 apresenta uma aproximação do comportamento da frequência percebida B, em mel, em termos da frequência f, expressa em Hz.

$$B(f) = 1125 \ln(1 + f/700) \tag{15}$$

A primeira etapa do processo de geração dos coeficientes MFCC consiste em calcular a transformada discreta de Fourier (*Discrete Fourier Transform* – DFT) das amostras $s[n]$ de um bloco de voz, definida como

$$S_a[k] = \sum_{n=0}^{N-1} s[n] e^{-j2\pi nk/N}, \quad 0 \leq k < N \tag{16}$$

onde N é o número de pontos da transformada e $s[n]$ a n-ésima amostra do sinal de entrada.

Para a extração dos atributos MFCC é utilizado um banco de filtros triangulares e igualmente espaçados na escala mel, com resposta em frequência definidas a partir da Equação 17.

$$H_m(k) = \begin{cases} 0, & k < f[m-1] \\ \frac{(k-f[m-1])}{f[m]-f[m-1]} & f[m-1] \leq k \leq f[m] \\ \frac{(f[m+1]-k)}{f[f+1]-f[m]} & f[m] \leq k \leq f[m+1] \\ 0 & k > f[m+1] \end{cases} \tag{17}$$

Na Equação 17, H_m representa o m-ésimo filtro do banco de filtros, $f[m]$ representa a frequência central em Hz (dado em termos dos índices da Transformada de Fourier) do m-ésimo filtro do banco de filtros, que é calculada a partir da Equação (18), onde f_l e f_h são, respectivamente, as frequências máxima e mínima do sinal considerado, N é o número de pontos da Transformada de Fourier,M é o número de filtros no banco filtros e F_s é a frequência de amostragem do sinal.

$$f[m] = \left(\frac{N}{F_s}\right) B^{-1}\left(B(f_l) + m\frac{B(f_h) - B(f_l)}{M+1}\right) \tag{18}$$

Em seguida, a energia logaritmica de cada filtro é calculada de acordo com

$$S[m] = \ln\left[\sum_{k=0}^{N-1} |X_a[k]|^2 H_m[k]\right], \quad 0 < m \leq M \tag{19}$$

Finalmente os coeficientes MFCC são obtidos pela aplicação da transformada discreta do cosseno (Discrete Sine Transform – DCT) sobre as M saídas dos filtros:

$$c[m] = \sum_{m=0}^{N-1} S[m]\cos(\pi n(m-1/2)/M) \quad 0 \leq n < M \tag{20}$$

A DCT tem o efeito de comprimir a informação de energia do banco de filtros, de forma que, no domínio DCT, os coeficientes MFCC de maior energia correspondem aos coeficientes de menores índices n . Além disso, a DCT causa um efeito benéfico de

produzir coeficientes MFCC praticamente descorrelatados. Em ASR são utilizados valores para M que variam entre 24 a 40, e geralmente os vetores de atributos MFCC de cada bloco do sinal são formados pelos primeiros 13 coeficientes $c[n]$.

2.3.3 Medição da Energia

Em adição aos atributos acústicos obtidos com o LPC, MFCC ou outro tipo de atributo, é muito comum a adição da energia de cada bloco ao vetor de atributos. A energia é calculada como o logaritmo da energia do sinal, ou seja, para as amostras $s[n]$, com $1 \leq 1 \leq N$, onde N é o número de amostras, a energia pode ser obtida a partir da Equação (21).

$$E = \ln \sum_{n=1}^{N} s[n]^2 \tag{21}$$

2.3.4 Coeficientes Dinâmicos

O desempenho do sistema de reconhecimento automático de voz pode ser significativamente melhorado através da adição das derivadas no tempo do sinal aos vetores de atributos. Esses novos componentes dos vetores acústicos, também chamados de componentes dinâmicos, em contraste aos componentes estáticos mostrados anteriormente, tem o objetivo de incorporar informação da variação temporal do sinal de voz aos vetores de atributos. Os atributos dinâmicos mais utilizados são a derivada primeira, também conhecidos como coeficientes delta, e a derivada segunda, também chamada coeficientes aceleração. Os coeficientes delta podem ser calculados como se segue

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \tag{22}$$

onde d_t é o vetor de coeficientes delta no tempo t, calculado em termos dos coeficientes

estáticos $c_{t-\Theta}$ a $c_{t+\Theta}$ (vetores de atributos nos tempos $t-\Theta$ e $t+\Theta$ respectivamente) , e Θ é o tamanho da janela de análise. A mesma fórmula pode ser utilizada para o cálculo dos coeficientes aceleração, exceto que no lugar dos coeficientes estáticos são utilizados os coeficientes delta.

2.4 Modelagem estocástica do sistema de reconhecimento automático de fala

O problema básico dos ASRs consiste em encontrar a sequência de palavras $\mathbf{W}$ que maximize a probabilidade a posteriori $P(\mathbf{W}|\mathbf{O})$ de uma determinada sequência de observações $\mathbf{O}$, em que cada observação da sequência corresponde a um vetor de atributos (por exemplo, atributos LPC ou MFCC) que caracteriza espectralmente um bloco de voz, ou seja:

$$\widehat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{O}) = \arg\max_{\mathbf{W}} \frac{P(\mathbf{W})P(\mathbf{O}|\mathbf{W})}{P(\mathbf{O})} = \arg\max_{\mathbf{W}} P(\mathbf{W})P(\mathbf{O}|\mathbf{W}) \qquad (23)$$

onde a probabilidade a posteriori $P(\mathbf{O}|\mathbf{W})$ é a probabilidade da sequência de observações $\mathbf{O}$ dado que se conhece a sequência de palavras $\mathbf{W}$e está relacionada ao modelo acústico desenvolvido na fase de treinamento do ASR, utilizando-se uma base de dados de voz. A probabilidade a priori $P(\mathbf{W})$ é a probabilidade da sequência de palavras $\mathbf{W}$ e está associada ao modelo de linguagem, construído a partir de uma base de dados escrita (normalmente, livros ou jornais). O termo $P(\mathbf{O})$foi omitido da Equação 23 devido ao fato do mesmo se manter constante para todas as hipóteses $\mathbf{W}$.

2.4.1 Modelo acústico

Em ASR, o modelo acústico tem o papel de representar de uma forma consistente as características temporais e espectrais do sinal de voz. O modelo estocástico mais

utilizado para esse fim é o Modelo Oculto de Markov (Hidden Markov Model – HMM). A teoria básica do HMM vem da década de 70, mas a sua primeira aplicação em ASR foi realizada na década de 80. Desde então, o HMM tem se tornado a principal abordagem para ASR, com aplicações que vão desde sistemas de reconhecimento de palavras isoladas para um vocabulário reduzido e dependente do locutor, até sistemas para reconhecimento de discurso contínuo com vocabulário extenso e independente do locutor.

2.4.1.1 Modelos ocultos de Markov

O Modelo Oculto de Markov (Hidden Markov Model – HMM) (Rabiner, 1989) é uma máquina de estados finitos que realiza uma transição de estado a cada unidade de tempo. O HMM aparece como uma extensão à Cadeia de Markov. Na Cadeia de Markov, cada estado, ou saída, é diretamente ligado a um evento físico, isto é, existe uma relação entre o estado e o evento observado. Já no HMM, a saída observada é uma função probabilística do estado. Dessa forma, o HMM pode ser definido como um processo duplamente estocástico, onde um dos processos, que gera a sequência de estados, não é diretamente observado, ou seja, é escondido, mas pode ser relacionado com outro processo estocástico, o qual gera a sequência de observações.

Um HMM pode ser considerado contínuo, semicontínuo ou discreto. No HMM contínuo, o alfabeto de observações não é finito, de forma que é necessária a utilização de funções densidade de probabilidade contínuas para modelar as probabilidades de saída (emissão de símbolos ou observação) dos estados. Já no HMM discreto, os símbolos possíveis de saída são oriundos de um alfabeto finito, de forma que é possível calcular a probabilidade de cada símbolo ser gerado em um determinado estado. No HMM semicontínuo, também chamado de estado compartilhado, as funções de densidade de probabilidade de saída são compartilhadas entre os estados, de forma que a função de densidade de probabilidade de saída em um determinado estado pode ser uma combinação linear das funções de outros estados.

Um HMM discreto pode ser especificado pelos seguintes elementos:

a) N, o número de estados no modelo. O conjunto de estados possíveis é $S = \{s_1, s_2, \ldots, s_N\}$, e o estado no tempo t é denotado por q_t . Nos ASRs, cada tempo t está associado a um bloco de voz, ou seja, a atualização do estado ocorre normalmente a cada 10ms;

b) M, o número de símbolos de observação possíveis em cada estado. O símbolo de observação corresponde à saída física do modelo. No caso de ASR, cada símbolo de observação é um vetor de atributos obtido a partir de um bloco do sinal de fala. O conjunto de símbolos possíveis é $V = \{v_1, v_2, \cdots, v_M\}$, e a observação no tempo t é denotada por O_t;

c) $A = \{a_{ij}\}$, a matriz de probabilidades de transição entre os estados, onde

$$a_{ij} = P\left[q_{(t+1)} = s_j \mid q_t = s_i\right] \geq 0, \quad 1 \leq i, j \leq N \tag{24}$$

d) $B = \{b_j(k)\}$, a matriz de probabilidades de emissão do símbolo de observação v_k no estado s_j, onde

$$b_j\left(k\right) = P\left[O_t = v_k \mid q_t = s_j\right], \quad 1 \leq j \leq N, \quad 1 \leq k \leq M \tag{25}$$

e) $\pi = \{\pi_i\}$, a matriz de probabilidades do estado inicial, onde

$$\pi_i = P\left[q_1 = s_i\right], \quad 1 \leq i \leq N \tag{26}$$

Se $a_{ij} > 0$ para todo par i, j , ou seja, é permitida a transição entre quaisquer estados, dizemos que o HMM é ergódigo ou completamente conectado. Quando há a limitação da transição de um estado para o anterior, ou seja $a_{ij} > 0$ somente quando $i > j$, ($a_{ij} = 0$ quando $i <= j$), chamamos o HMM de esquerda-direita.

A especificação completa de um HMM discreto envolve a especificação dos parâmetros empíricos do modelo, N e M , a especificação dos símbolos de observação e a

especificação das três matrizes de probabilidades, A, B e π . Por conveniência, usa-se a notação $\lambda = (A, B, \pi)$ para representar o conjunto completo de parâmetros do modelo.

Definidos os elementos que constituem um HMM, há três problemas básicos cujas soluções têm aplicação prática. Eles são chamados de "os três problemas do HMM" e estão listados a seguir:

1. Sejam uma sequência de observações $O = \{O_1O_2, \ldots, O_T\}$ de comprimento T e um modelo λ , como calcular de forma eficiente $P(O|\lambda)$, a probabilidade da sequência de observações, dado um certo modelo?

2. Dada a sequência de observações $O = \{O_1O_2, \ldots, O_T\}$ de comprimento T e o modelo λ , como encontrar a sequência de estados correspondente, $Q = \{q_1q_2, \ldots, q_T\}$, ótima segundo algum critério?

3. Como ajustar os parâmetros do modelo $\lambda = (A, B, \pi)$ de forma que a probabilidade $P[O|\lambda]$ seja maximizada?

2.4.1.1.1 Solução do problema 1: Algoritmo Forward

No problema 1, deseja-se calcular a probabilidade de uma sequência de observações $O = \{O_1O_2, \ldots, O_T\}$ ter sido gerada pelo modelo $\lambda = (A, B, \pi)$, isto é, $P[O|\lambda]$. A solução desse problema tem utilidade prática para a determinação de qual modelo, dentre vários disponíveis, tem maior probabilidade de ter gerado a sequência de observações ocorrida, considerando-se todas as possíveis sequências de estados.

A forma mais direta de solucionar esse problema é enumerar todas as possíveis sequências de estados Q_p com comprimento T, $Q_p = \{q_{p_1}q_{p_2} \cdots q_{p_T}\}$, determinar $P\left[O, Q_p|\lambda\right]$, ou seja, a probabilidade de que cada sequência de estados Q_p tenha gerado a sequência de observações $O = \{O_1O_2 \cdots O_T\}$ (probabilidade conjunta entre O e Q_p) e em seguida, calcular $P\left[O|\lambda\right] = \sum_p P\left[O, Q_p|\lambda\right]$. A probabilidade $P\left[O, Q_p|\lambda\right]$pode ser escrita da seguinte forma:

$$P\left[O, Q_p|\lambda\right] = P\left[O|Q_p, \lambda\right] P\left[Q_p|\lambda\right] \tag{27}$$

$$P\left[O|Q_p,\lambda\right] = P\left[O^{1\to T}|Q_p^{t\to T},\lambda\right] = \prod_{t=1}^{T} P\left[O_t|O^{1\to(t-1)},Q_p^{t\to T},\lambda\right] \tag{28}$$

onde $O = O^{1\to T} = O_1O_2\cdots O_T$ e $Q_p = Q_p^{1\to T} = q_{p_1}q_{p_2}\cdots q_{p_T}$.

Assume-se que o processo de Markov é de primeira ordem, ou seja, a probabilidade de ocorrência do estado $q_t = s_j$ só depende do estado $q_{(t-1)}$. Assume-se também a hipótese de que observações consecutivas são estatisticamente independentes, ou seja, a probabilidade de um dado símbolo de observação O_t só depende do estado q_t e é condicionalmente independente das observações nos tempos anteriores.

Como O_t e O_{t-1} são estatisticamente independentes e a probabilidade de um dado símbolo de observação O_t só depende do estado q_t:

$$P\left[O_t|O^{1\to(t-1)},Q_p^{1\to T},\lambda\right] = P\left[O_t|q_t,\lambda\right] \tag{29}$$

e então:

$$P\left[O|Q_p,\lambda\right] = \prod_{t=1}^{T} P\left[O_t|q_{p_t},\lambda\right] = b_{q_{p1}}(O_1)b_{q_{p2}}(O_2)\cdots b_{q_{pT}}(O_T) \tag{30}$$

em que

$$b_{q_{p1}}(O_1)b_{q_{p2}}(O_2)\cdots b_{q_{pT}}(O_T)$$

são elementos da matriz**B**.

Utilizando a suposição de modelo de Markov de Primeira ordem, temos:

$$\begin{aligned} P\left[Q_p|\lambda\right] = P\left[Q_p^{t\to T}|\lambda\right] &= P\left[q_{p_1}|\lambda\right]\prod_{t=2}^{T} P\left[q_{p_t}|q_{p_{(t-1)}},\lambda\right] \\ &= \pi_{q_{p_1}}a_{q_{p_1}q_{p_2}}\cdots a_{q_{p_{(T-1)}}q_{p_{(T)}}} \end{aligned} \tag{31}$$

em que $\pi_{q_{p_1}}$ são elementos da matriz π e $a_{q_{p_1}q_{p_2}}\cdots a_{q_{p_{(T-1)}}q_{p_{(T)}}}$ são elementos da matriz $\mathbf{A}$. Portanto:

$$\begin{aligned} P\left[O|\lambda\right] &= \sum_p P\left[Q_p|\lambda\right]P\left[O|Q_p,\lambda\right] \\ &= \sum_p \pi_{q_{p_1}}b_{q_{p_1}}(O_t)a_{q_{p_1}q_{p_2}}b_{q_{p_2}}(O_2)\cdots a_{q_{p_{(T-1)}}q_{p_T}}b_{q_{p_T}}(O_T) \end{aligned} \tag{32}$$

O cálculo de $P\,[O|\lambda]$, de acordo com a definição em 32, envolve um número de cálculos da ordem de $2T \cdot N^T$, onde T é o tamanho da sequência de observações e N é o número de estados do modelo. Felizmente, existe um procedimento eficiente para esse cálculo, o qual necessita um número de cálculos da ordem de N^2T, chamado procedimento Forward. Considere a variável forward $\alpha_t(i)$, definida como:

$$\alpha_t(i) = P(O_1 O_2 \ldots O_t | q_t = s_i | \lambda) \tag{33}$$

isto é, $\alpha_t(i)$ é a probabilidade conjunta da ocorrência da sequência parcial de observações nos instantes de 1 a t, $O_1 O_2 \ldots O_t$, e do estado s_i no instante t , ou seja, de $q_t = s_i$. Pode-se calcular $\alpha_t(i)$ por indução, da seguinte maneira:

a) Inicialização:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N \tag{34}$$

b) Indução:

$$a_{(t+1)}(j) = \left[\sum_{i=1}^{N} a_t(i) a_{ij}\right] b_j(O_{(t+1)}), \quad 1 \leq t \leq T-1 \qquad 1 \leq j \leq N \tag{35}$$

c) Término:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{36}$$

2.4.1.1.2 Solução do Problema 2: Algoritmo Viterbi

No problema 2, o objetivo é encontrar uma sequência de estados $Q = Q^{1\to T} = q_1 q_2 \ldots q_T$ que melhor se adapta ao modelo λ , dada uma sequência de observações $O = O^{1\to T} = O_1 O_2 \ldots O_T$. Na prática, deseja-se encontrar a sequência de estados Q que maximiza $P[Q|O, \lambda]$, que é equivalente a maximizar $P[Q, O|\lambda]$. A técnica formal para se encontrar essa sequência é chamada de algoritmo de Viterbi.

O algoritmo de Viterbi tem aplicação prática idêntica à do problema 1, mas com uma simplificação: enquanto no problema 1 consideram-se todas as possíveis sequências de estados Q_p, o problema 2 foca em encontrar a sequência de estados $Q_p^{1\to T}$ mais provável

para uma determinada sequência de observações $O^{1\to T}$, para cada modelo λ. Para isso, é necessário definir a quantidade $\delta_t(i)$:

$$\delta_t(i) = \max_{q_1,q_2,\ldots q_{(t-1)}} P\left[q_1, q_2, \ldots, q_t = s_i, O_1, O_2, \ldots O_t|\lambda\right] \tag{37}$$

isto é, a máxima probabilidade conjunta da sequência de observações parciais $O^{1\to t} = \{O_1 O_2 \ldots O_t\}$ e da sequência de estados $Q^{1\to(t-1)} = q_1 q_2 \ldots q_{(t-1)}$, considerando-se que o estado no instante t seja $q_t = s_i$, para um modelo λ.

Por indução, pode-se afirmar que:

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij}\right] b_j(O_{(t+1)}) \tag{38}$$

Para recuperar a sequência de estados, é necessário armazenar o argumento i que maximiza a Equação 38, para todo j e t, na matriz $\psi_t(i)$. Assim, o procedimento completo para se encontrar a sequência de estados Q que maximiza $P[Q|O,\lambda]$ é:

a) inicialização:

$$\begin{aligned} &\delta_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N \\ &\psi_1(i) = 0 \end{aligned} \tag{39}$$

b) recursão:

$$\begin{aligned} \delta_t(j) = \max_{1\le i\le N} \left[\delta_{(t-1)}(i) a_{ij}\right] b_j(O_t), \quad & 2 \le t \le T \\ & 1 \le j \le N \\ \psi_t(j) = \arg\max_{1\le i\le N} \left[\delta_{(t-1)}(i) a_{ij}\right], \quad & 2 \le t \le T \\ & 1 \le j \le N \end{aligned} \tag{40}$$

c) término:

$$\begin{aligned} P^* &= \max_{1\le i\le N}[\delta_T(i)] \\ q_T^* &= \arg\max_{1\le i\le N}[\delta_T(i)] \end{aligned} \tag{41}$$

d) recuperação sa sequência de estados:

$$q_t^* = \psi_{(t+1)}(q_{(t+1)}^*), \quad t = T-1, T-2, \ldots, 1 \tag{42}$$

A solução do problema 2, além de fornecer a sequência de estados que melhor se casa a uma determinada sequência de observações, fornece também uma medida proximidade $P^* = P[O, Q|\lambda]$, a qual pode ser utilizada em substituição à probabilidade da sequência de observações $P[O|\lambda]$, que é o objeto de estudo do problema 1. Uma das vantagens de se calcular a probabilidade da sequência de observações utilizando $P^* = P[O, Q|\lambda]$, medida que também é também chamada de verossimilhança, é que seu cálculo é mais eficiente computacionalmente, uma vez que as operações de produto encontradas no algoritmo forward são substituídas por operações de soma no algoritmo de Viterbi e as operações de somatório são substituídas por operações de maximização.

2.4.1.1.3 Solução do Problema 3: Algoritmo Baum-Welch

No problema 3, o objetivo é estimar os parâmetros do modelo λ, que são $\mathbf{A}$, $\mathbf{B}$ e π de forma a maximizar a probabilidade $P[O|\lambda]$. Não há um modo conhecido de se resolver esse problema de forma analítica, ou seja, não existe uma solução ótima para uma dada sequência finita de observações $O^{1\rightarrow T}$. Pode-se, no entanto, estimar os parâmetros de forma que $P[O|\lambda]$ seja localmente maximizado. Na prática, essa estimação é realizada a partir de um procedimento iterativo chamado de método de Baum-Welch.

Antes de tudo, é necessária a definição da variável backward:

$$\beta_t(i) = P(O_{(t+1)}O_{(t+2)} \dots O_T | q_t = s_i, \lambda) \tag{43}$$

que é a probabilidade de que uma sequência de observações $O_{(t+1)}O_{(t+2)} \dots O_T$ ocorra entre um instante t qualquer e o instante T, dado que o estado no tempo t é $q_t = s_i$ e que o modelo λ é conhecido. A partir das variáveis forward e backward pode-se calcular $P(O|\lambda)$ como:

$$P[O|\lambda] = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \tag{44}$$

A probabilidade backward pode ser calculada iterativamente da seguinte maneira:

a) Inicialização:

$$\beta_T(i) = 1 \quad 1 \leq i \leq N \tag{45}$$

b) Indução:

$$\begin{aligned} \beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{(t+1)}) \beta_{(t+1)}(j) \quad t = T-1, T-2, \ldots, 1 \\ 1 \leq i \leq N \end{aligned} \tag{46}$$

Define-se então a variável $\xi_t(i,j)$ como a probabilidade de que o estado no instante t seja $q_t = s_i$ e o estado no instante $(t+1)$ seja $q_{(t+1)} = s_j$, dado o modelo e a sequência de observações, ou seja:

$$\xi_t(i,j) = P\left(q_t = s_i \ , \ q_{t+1} = s_j \mid O \ , \ \lambda\right) = \frac{P(q_t = s_i, q_{t+1} = s_j, O|\lambda)}{P(O|\lambda)} \tag{47}$$

que pode ser escrita em termos das variáveis forward e backward como:

$$\begin{aligned} \xi_t(i,j) &= \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{(t+1)})\, \beta_{(t+1)}(j)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{(t+1)})\, \beta_{(t+1)}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i)\, a_{ij}\, b_j(O_{(t+1)})\, \beta_{(t+1)}(j)} \end{aligned} \tag{48}$$

É necessário também definir $\gamma_t(i)$ como a probabilidade do estado no instante t ser igual s_i, sendo conhecida a sequência completa de observações O e o modelo λ, isto é:

$$\gamma_t(i) = P(q_t = s_i|O, \lambda) \tag{49}$$

Pode-se relacionar $\xi_t(i,j)$ e $\gamma_t(i)$ da seguinte forma:

$$\gamma_t(i) = \sum_{j=1}^{N} P(q_t = s_i, q_{(t+1)} = s_j|O, \lambda) = \sum_{j=1}^{N} \xi_t(i,j) \tag{50}$$

Se a variável $\gamma_t(i)$ é somada ao longo do índice t, obtém-se um resultado que pode ser interpretado como sendo o número esperado de vezes que o estado s_i foi visitado. Se esse somatório for realizado até o tempo $(T-1)$, o resultado obtido pode ser interpretado como sendo o número esperado de transições a partir do estado s_i. De forma similar,

quando a variável $\xi_t(i,j)$ é somada ao longo do intervalo entre $t=1$ e $(T-1)$, o resultado representa o número esperado de transições do estado s_i para o estado s_j. Dessa forma,

$$\sum_{t=1}^{(T-1)} \gamma_t(i) = \text{numero de transicoes a partir de } s_i \tag{51}$$

e

$$\sum_{t=1}^{(T-1)} \xi_t(i,j) = \text{numero de transicoes de s}_i \text{ para } s_j \tag{52}$$

Utilizando as equações acima e o conceito de frequência de ocorrência relativa, pode-se deduzir as equações para a estimação dos parâmetros de π e $\mathbf{A}$ como:

$$\pi_i = \lambda_1(i) = \textit{numero mdio de ocorrencia do estado } s_i \textit{ no tempo } t=1 \tag{53}$$

Como $a_{ij} = P(q_{t+1} = s_j | q_t = s_i) = \frac{P(q_{t+1}=s_j, q_t=s_i)}{P(q_t=s_i)}$, então:

$$a_{ij} = \frac{\textit{numero esperado de transicoes do estado } s_i \textit{ para o estado } s_j}{\textit{numero esperado de transicoes a partir do estado } s_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{54}$$

Para o caso onde o alfabeto de símbolos de saída é discreto, a função densidade de probabilidade de emissão de símbolos $b_j(k)$ também será discreta (ou seja, $b_j(k)$ será uma medida de probabilidade), de forma que os elementos de B podem ser calculados da seguinte maneira:

$$b_j(k) = P\left(O_t = v_k | q_t = s_i\right) = \frac{P(O_t = v_k, q_t = s_j)}{P(q_t = s_j)} \tag{55}$$

então:

$$b_j(k) = \frac{numero\ esperado\ de\ observacoes\ do\ simbolo\ v_k\ no\ estado\ s_j}{numero\ esperado\ de\ vezes\ em\ que\ se\ ocupa\ o\ estado\ s_j} = \frac{\sum_{=1}^{T}{}_{quando\ O_t=v_k} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{56}$$

Como esse procedimento é iterativo, é necessário definir valores iniciais para os parâmetros dos modelos. Pode ser mostrado que após uma iteração, o modelo resultante será melhor ou igual ao modelo anterior. Quando o modelo resultante é melhor, o modelo antigo deve ser substituído por ele e então o procedimento deve ser repetido até não haver mais alterações de uma iteração para a outra.

A teoria aqui descrita para a estimação dos parâmetros do HMM é a teoria básica. Alguns detalhes relativos à implementação prática foram omitidos, tais como: a normalização das variáveis forward e backward para evitar problemas com precisão numérica; o uso do logaritmo da variável de Viterbi, também para evitar problemas com precisão numérica; o uso de múltiplos vetores de observação para a estimação dos parâmetros dos modelos e formas de inicialização dos parâmetros do modelo.

2.4.2 Reconhecimento de palavras isoladas utilizando HMM

A base teórica apresentada nas seções anteriores pode ser facilmente empregada na construção de um sistema para reconhecimento de palavras isoladas. A título de exemplo, considere um vocabulário contendo V palavras a serem reconhecidas. Cada uma dessas palavras deve ser modelada por um HMM distinto, se a palavra é utilizada como unidade fonética. Para isso, assume-se que esteja disponível uma base de dados de voz (conjunto de treinamento) constituída de k ocorrências de cada palavra (faladas por um ou mais locutores). Cada ocorrência gera uma sequência de observações (normalmente uma observação a cada 10ms), onde essas observações são alguma representação eficiente do sinal de fala (MFCC ou LPC, por exemplo). O processo de treinamento pode ser realizado a partir

do algoritmo de Baum-Welch, como descrito na solução do problema 3. Para a identificação de uma palavra desconhecida a ser reconhecida, a sequência de observações $O^{1\rightarrow T}$ deve ser avaliada de forma a se obter a medida de probabilidade ou verossimilhança para cada modelo. Essas medidas de probabilidade ou verossimilhança podem ser conseguidas através da aplicação do algoritmo forward, ou o algoritmo de Viterbi. Finalmente, a identidade da palavra desconhecida é aquela do modelo que apresentar a maior probabilidade ou verossimilhança.

2.4.3 Modelo de linguagem

Até o momento só foi abordado o problema do modelo acústico, nesta seção serão abordados os conceitos de modelo de linguagem, incluindo a gramática de estado finito, a gramática de livre contexto e os modelos de linguagem n-grama.

2.4.3.1 Gramatica de estado finito e gramática de livre contexto

Em muitas aplicações de reconhecimento automático de fala, existe um pequeno número de possíveis sentenças a serem reconhecidas. Nesse tipo de aplicação o conceito de gramática de estado finito (*Finite State Grammar* – FEG) pode ser utilizado. Uma FEG é uma estrutura determinística, a qual descreve a estrutura gramatical das sentenças a serem reconhecidas.

Muitos problemas podem ser resolvidos pela aplicação das FEGs. No entanto, se a estrutura da gramática for muito complexa, exige-se o compartilhamento de diferentes subgramáticas e o suporte a modificações dinâmicas. Essa FEG não determinística resultante é muito parecida com uma gramática de livre contexto (*Context-Free Grammar* - CFG) em termos de implementação.

A CFG consiste em um série de regras que expandem nós não-terminais em uma

sequência de nós terminais e não-terminais. Nós não-terminais geralmente representam classes de palavras, como datas, números e nomes, enquanto nós terminais representam palavras do vocabulário.

Tanto as FEGs como as CFGs são estruturas muito úteis para definir a estrutura gramatical das sentenças a serem reconhecidas. No entanto, tais estruturas não podem ser empregadas quando não se sabe o domínio da aplicação. Nesses casos, onde não se conhece o domínio da aplicação previamente, pode ser utilizado um modelo estatístico, o qual consegue representar indiretamente tanto as estruturas sintáticas, quanto as estruturas semânticas da linguagem. Tais modelos são conhecidos como modelos n-gramas e são descritos na subseção a seguir.

2.4.3.2 Modelo de linguagem n-grama

A probabilidade a priori $P(\mathbf{W})$ de uma sequência de palavras $\mathbf{W}$ ser pronunciada é calculada de acordo com a Equação 57, onde $P(w_i|w_1w_2\ldots w_{i-1})$ é a probabilidade de que a palavra w_i ocorra logo após a sequência $w_1w_2\ldots w_{i-1}$. Se considerado um vocabulário de tamanho V, pode-se concluir que há V^{i-1} possíveis históricos para a palavra w_i(sequências de palavras pronunciadas antes de w_i), e para calcular $P(w_i|w_1w_2\ldots w_{i-1})$ são necessários um total de V^i valores a serem estimados.

$$\begin{aligned} P(\mathbf{W}) &= P(w_1, w_2, \ldots, w_n) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\cdots P(w_n|w_1, w_2, \ldots, w_{n-1}) \\ &= \prod_{i=1}^{n} P(w_i|w_1, w_2, \ldots, w_{i-1}) \end{aligned} \tag{57}$$

Na prática, as probabilidades $P(w_i|w_1w_2\cdots w_{i-1})$ são impossíveis de serem estimadas até para valores moderados de i, uma vez que algumas sequências $w_1w_2\cdots w_{i-1}$ não ocorrem, ou ocorrem apenas algumas vezes. Uma forma de amenizar esse problema é a realização do cálculo das probabilidades condicionais truncando o histórico $w_1w_2\cdots w_{i-1}$ em $N-1$ palavras anteriores à palavra w_i, $w_{i-N+1}w_{i-N+2}\cdots w_{i-1}$, o que leva aos modelos

n-grama. Se a palavra depende de duas palavras anteriores, temos o modelo tri-grama: $P(w_i|w_{i-2}, w_{i-1})$. De forma similar, se a palavra depende de apenas uma palavra anterior, temos o modelo bi-grama: $P(w_i|w_{i-1})$.

Para calcular $P(w_i|w_{i-1})$, que é a probabilidade da palavra w_i ocorrer logo após a palavra w_{i-1}, basta contar quantas vezes a sequência w_{i-1}, w_i ocorre e normalizar esse valor pelo número de ocorrências de w_{i-1}, como mostra a Equação 58, em que a letra C foi usada para designar a quantidade de ocorrências observadas.

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})} \tag{58}$$

Para tornar possível o cálculo de $P(w_i|w_{i-1})$, mesmo na situação de $i = 1$, é necessária a criação de um símbolo especial para o início da sentença $< s >$. Além disso, para que a soma das probabilidades de cada uma das sequências seja 1, é necessária a criação de um símbolo especial para o final da sentença, aqui escolhido como sendo $< /s >$.

Para calcular as probabilidades tri-grama $P(w_i|w_{i-2}w_{i-1})$, ou seja, a probabilidade da palavra w_i ocorrer logo após a sequência $w_{i-2}w_{i-1}$, basta contar a ocorrência do trio $C(w_{i-2}w_{i-1}w_i)$ e do par $C(w_{i-1}w_i)$, então usar a Equação 59.

A base de dados utilizada para o treinamento dos modelos de linguagem é chamada de corpora de texto. Para se obter uma boa estimação, a quantidade de dados utilizados chega a milhões de sentenças.

$$P(w_i|w_{i-1}, w_{i-2}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})} \tag{59}$$

Considere um pequeno exemplo. Seja o conjunto de treinamento S, formado pelas sentenças abaixo:

Quadro 1 - Conjunto de treinamento para o modelo de linguagem

Luiz leu seu livro
Eu li um livro diferente
Luiz leu um livro do Paulo Coelho

Fonte: Elaborado pelo autor

Deseja-se então calcular a probabilidade de ocorrência da sentença “Luiz leu um livro”. Tal probabilidade pode ser calculada a partir das probabilidades bi-grama abaixo:

$$\begin{aligned}
&P(Luiz| < s >) = \tfrac{C(<s>,Luiz)}{C(<s>)} = \tfrac{2}{3} \\
&P(leu|Luiz) = \tfrac{C(Luiz,leu)}{C(leu)} = \tfrac{2}{2} \\
&P(um|leu) = \tfrac{C(leu,um)}{C(leu)} = \tfrac{1}{2} \\
&P(livro|um) = \tfrac{C(um,livro)}{C(um)} = \tfrac{2}{2} \\
&P(< /s > |livro) = \tfrac{C(livro,</s>)}{C(livro)} = \tfrac{1}{3}
\end{aligned} \tag{60}$$

de forma que

$$\begin{aligned}
&P(Luizleuumlivro) = \\
&P(Luiz| < s >)P(leu|Luiz)P(um|leu)P(livro|um)P(< /s > |livro) \\
&\approx 0.1111
\end{aligned} \tag{61}$$

O mesmo conceito pode ser aplicado para o cálculo da probabilidade da sentença utilizando as probabilidades tri-grama. Com a utilização das probabilidades tri-grama, um problema já presente quando no uso das probabilidades bi-grama fica mais evidenciado. Como o cálculo das probabilidades é baseado na frequência relativa de ocorrência, pode acontecer de determinados pares ou trios não aparecerem na base de treinamento, de forma que algumas probabilidades receberão valor zero. Com isso, sentenças que teriam uma boa probabilidade de ocorrência podem apresentar probabilidade nula, como é o caso da sentença “Paulo leu um livro”. Tal situação pode ser contornada através de uma suavização do modelo de linguagem.

O nome suavização se deve ao fato que técnicas que a empregam tendem a tornar a distribuição de probabilidade do modelo de linguagem mais uniforme, ajustando para cima as probabilidades baixas como zero e reduzindo as probabilidades mais altas. Tais métodos não somente previnem a ocorrência de probabilidades zero, como também aumenta a eficácia do modelo como um todo. De forma geral, os algoritmos de suavização empregam uma estratégia de retrocesso (*back-off*) ou uma interpolação. As estratégias de retrocesso

utilizam probabilidades n-grama de menor ordem para a estimação da probabilidade n-grama que não ocorrem. Sendo assim, a maioria dos algoritmos podem ser descritos como se segue:

$$
\begin{aligned}
&P_{suavizado}(w_i|w_{i-n+1}\dots w_{i-1}) \\
&= \begin{cases} \alpha(w_i|w_{i-n+1}\dots w_{i-1}) & se\ C(w_{i-n+1}\dots w_{i-1}) > 0 \\ \gamma(w_{i-n+1}\dots w_{i-1})P_{suavizado}(w_i|w_{i-n+2}\dots w_{i-1}) & se\ C(w_{i-n+1}\dots w_{i-1}) = 0 \end{cases}
\end{aligned} \tag{62}
$$

Ou seja, se algum n-grama apresentar contagem diferente de zero, é utilizada a distribuição $\alpha(w_i|w_{i-n+1}\dots w_{i-1})$. Caso contrário, são utilizados n-gramas de ordens menores, onde o fator de escala $\gamma(w_{i-n+1}\dots w_{i-1})$ é escolhido de modo a fazer a distribuição condicional somar um. Os algoritmos de suavização que empregam a estrutura da Equação 62 são chamados modelos de retrocesso (*back-off models*).

Os algoritmos que empregam estratégias de interpolação geralmente expressam a probabilidade n-grama suavizada, $P_{suavizado}(w_i|w_{i-n+1}\dots w_{i-1})$, em termos de uma interpolação linear, como se segue:

$$
\begin{aligned}
&P_{suavizado}(w_i|w_{i-n+1}\dots w_{i-1}) \\
&= \lambda P(w_i|w_{i-n+1}\dots w_{i-1}) + (1-\lambda)P_{suavizado}(w_i|w_{i-n+2}\dots w_{i-1})
\end{aligned} \tag{63}
$$

onde λ é o peso de interpolação, o qual depende de $w_{i-n+1}\dots w_{i-1}$. Tais modelos são ditos modelos interpolados.

Entre os diversos algoritmos que empregam uma abordagem ou outra, podem ser destacados: *Deleted Interpolation Smoothing*, Katz *Smoothing* e Knezer-Ney *Smoothing* (Huang et al., 2001). A descrição dos algoritmos citados não será dada aqui, por fugir do escopo da pesquisa, devendo o leitor verificar as referências para mais informações.

2.4.3.3 Medição da perplexidade do modelo de linguagem

A forma mais natural de se medir a qualidade de um modelo de linguagem é através da medição da taxa de acerto do sistema. Embora essa abordagem forneça uma boa estimativa, não é muito prática, pois conta com a participação do sistema de reconhecimento. Felizmente há uma forma de calcular a qualidade do modelo de linguagem sem envolver o sistema de reconhecimento. Tal medida é derivada da entropia cruzada e é conhecida como perplexidade.

Considere um modelo de linguagem que atribui a probabilidade $P(\mathbf{W})$ para uma sequência de palavras $\mathbf{W}$. Pode-se derivar um algoritmo de compressão que codifica a sequência de palavras $\mathbf{W}$ utilizando $-\log_2(\mathbf{W})$ bits.

A entropia cruzada $H(\mathbf{W})$ do modelo $P(w_i|w_{i-n+1}\dots w_{i-1})$ utilizando os dados $\mathbf{W}$, com uma sequência de palavras suficientemente longa, pode ser definida como:

$$H(\mathbf{W}) = -\frac{1}{N_w}\log_2 P(\mathbf{W}) \tag{64}$$

onde N_w é o número de palavras contidas na sequência de palavras $\mathbf{W}$ e portanto, a entropia cruzada é uma medida que representa a quantidade média de bits necessária para codificar cada palavra da sequência de palavras $\mathbf{W}$. A partir da entropia cruzada pode-se definir a perplexidade como

$$PP(\mathbf{W}) = 2^{H(\mathbf{W})} \tag{65}$$

A perplexidade está intimamente relacionada com o fator de ramificação do texto de acordo com o modelo de linguagem. De forma grosseira, pode-se dizer que a perplexidade representa o número médio de palavras (equiprováveis) que podem preceder uma determinada palavra. Por exemplo, na tarefa de reconhecimento contínuo de dígitos (considerando os dígitos de 0 a 10), onde todos os dígitos possuem a mesma probabilidade de ocorrência, a entropia-cruzada calculada de acordo com a Equação 64 é $3,3219$ e por consequência a perplexidade, segundo a Equação 65, é 10. Pode-se dizer que quanto menor

a perplexidade do modelo de linguagem, menor será a taxa de erro do ASR. A Equação 66 fornece uma aproximação do WER em função da perplexidade (E. Silva, Pantoja, Celidônio, & Klautau, 2004).

$$WER \approx -12,37 + 6,48\log_2(PP(\mathbf{W})) \tag{66}$$

A Tabela 1 apresenta os valores de perplexidade para diversas tarefas de reconhecimento automático de fala encontrados na literatura. É possível notar que a perplexidade é fortemente relacionada ao tamanho do vocabulário, sobretudo quando a probabilidades n-grama de todas as palavras são iguais, como o caso do reconhecimento de números ou dígitos. Em se tratando de bases de dados para reconhecimento de discurso contínuo, o tamanho do vocabulário ainda tem influência na perplexidade, mas a perplexidade também está relacionada à complexidade do texto. Espera-se que o conteúdo de uma conversa espontânea tenha uma complexidade bem menor que o conteúdo de um jornal, o que faz com que a base de dados *Conversational Speech* apresente perplexidade menor que as bases *Wall Street Journal* e *Broadcast News*.

Tabela 1 – Perplexidade e WER associados a diferentes tarefas de reconhecimento automático de fala

Base de dados	Tamanho do vocabulário	Perplexidade	WER
TI Digits	11	11	0,0%
OGI Alphadigits	36	36	8%
Resource Management (RM)	1000	60	4%
Air Travel Information Service (ATIS)	1800	12	4%
Wall Street Journal	>20000	200-250	15%
Broadcast News	>80000	200-250	20%
Conversational Speech	>50000	100-150	30%

Fonte: (Teruszkin & Vianna, 2006)

É mostrado (Huang et al., 2001) que modelos de linguagem empregando quaisquer dos métodos de suavização citados anteriormente apresentam menor perplexidade que modelos de linguagem sem nenhuma forma de suavização, levando assim a sistemas de reconhecimento com maiores taxas de acerto. Com isso fica evidente a necessidade do emprego de tais técnicas, mesmo quando há abundância de dados de treinamento.

2.5 Métricas de desempenho de um ASR

Para a medição do desempenho de um ASR é necessário o emprego de métricas adequadas. Dentre as métricas de desempenho mais utilizadas se encontram a taxa de erro de palavras (Word Error Rate – WER) e o fator de tempo real (Real Time Factor – RTF).

O WER é a métrica mais comum utilizada para quantificar o desempenho do sistema de reconhecimento de fala. Para seu cálculo é necessária a comparação das transcrições obtidas como saída do ASR com a transcrição real das sentenças em teste. A partir dessa comparação são contabilizadas as seguintes quantidades: o número de palavras substituídas (S), o número de palavras inseridas incorretamente (I) e o número de palavras deletadas (D). Com isso o WER é calculado conforme a Equação 67:

$$WER = \frac{S + I + D}{N} \times 100\% \tag{67}$$

onde N é o número de palavras na sequência de palavras de referência.

O RTF, por outro lado, é uma medida que reflete a velocidade de processamento do sistema. É calculado pela razão entre o tempo necessário para se processar um conjunto de sentenças (T_s) e a duração real das sentenças (T_d), mostrado na Equação 68.

$$RTF = \frac{T_s}{T_d} \tag{68}$$

2.6 Refinamento na estrutura básica do ASR

Na seção anterior foi introduzida a teoria básica do HMM, o que inclui a definição formal de seus parâmetros e a enumeração dos problemas que precisam ser resolvidos para que o HMM possa ser utilizado em aplicações práticas. Foi mostrado também como aplicar o HMM no reconhecimento de palavras isoladas. Na presente seção serão apresentados

refinamentos à estrutura básica do HMM, bem como o algoritmo mais utilizado para o reconhecimento de discurso contínuo baseado em HMM.

2.6.1 Modelo de Misturas de Gaussianas

Na formulação básica do HMM é assumido que os possíveis símbolos de observação são oriundos de um alfabeto finito de símbolos. Isso significa que os símbolos de observação devem ser discretos por natureza ou oriundos de um processo de discretizaçao. Entretanto, como cada elemento do vetor de atributos pode assumir uma grande gama de valores, o número de vetores distintos que podem ser formados acaba sendo exageradamente grande, de forma que a probabilidade de ocorrência desse vetor seria extremamente baixa.

Uma alternativa seria utilizar uma quantização vetorial, de forma a mapear todos os possíveis vetores de atributos em um número reduzido de vetores. Existem várias métodos de quantização vetorial (Huang et al., 2001), dentre eles citam-se os métodos LGB e K-means .

Embora a quantização vetorial resolva parcialmente o problema do número muito grande de vetores possíveis, essa quantização acaba introduzindo um ruído nos vetores de atributo, reduzindo assim a qualidade do sistema. Diante disso faz-se necessário o emprego de uma alternativa à quantização, o que é conseguido com o modelo de mistura de gaussianas (*Gaussian Misture Model* – GMM).

O GMM consiste em estimar uma função densidade de probabilidade (fdp) do vetor de emissão (vetor de atributos acústicos), caracterizada como a fdp conjunta de emissão de símbolos (cada um dos coeficientes MFCC do vetor de atributos) em cada estado, ao invés de calcular a probabilidade de se observar cada um dos possíveis vetores de emissão, em cada estado. Como o nome indica, a GMM é constituída da soma ponderada de M gaussianas (M funções normais N), como mostrado na Equação 69. E então, a probabilidade de observação de um determinado vetor de atributos é calculada avaliando-se o valor da fdp para esse vetor. O uso de mistura de fdp gaussianas é interessante, pois

com o número adequado de componentes é possível aproximar qualquer fdp (Rabiner, 1989).

$$b_j(O_t) = \sum_{m=1}^{M} c_{jm} N(O_t, \mu_{jm}, U_{jm}) \tag{69}$$

onde c_{jm} é o m-ésimo coeficiente de ponderação da mistura no estado j e c_{jm} é a função densidade de probabilidade conjunta de um vetor gaussiano O_t (ou a probabilidade conjunta das componentes do vetor de atributos), com vetor média μ e matriz de covariância U, ou seja:

$$N(O_t, \mu, U) = \frac{1}{\sqrt{(2n)^n \, |U|}} e^{-\frac{1}{2}(O_t-\mu)^T U^{-1}(O_t-\mu)} \tag{70}$$

onde n é a dimensão do vetor O_t (quantidade de atributos) e $|U|$ é o determinante da matriz de covariância.

Pode ser mostrado que as equações para a reestimação dos parâmetros da mistura são:

$$c_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_t(j,k)} \tag{71}$$

$$\mu_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) O_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{72}$$

$$U_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)(O_t - \mu_{jk})(O_t - \mu_{jk})^T}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{73}$$

onde $\gamma_t(j,k)$ é a parcela da probabilidade do estado no instante t ser igual a s_j, $P[q_t = s_j]$, sendo conhecida a sequência completa de observações O, presente na k-ésima componente da mistura. Ou seja:

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)}\right]\left[\frac{c_{jk}N(O_t,\mu_{jk},U_{jk})}{\sum_{m=1}^{M}c_{jm}N(O_t,\mu_{jm},U_{jm})}\right] \quad (74)$$

A Equação 74 resume-se à Equação 49 quando considerada apenas uma componente na mistura, ou seja, $M = 1$, ou quando se considera uma distribuição discreta.

2.6.2 Escolha das unidades fonéticas

Foi mencionado anteriormente que se o vocabulário do sistema não é muito grande, pode-se utilizar a própria palavra como unidade fonética, de forma que cada HMM representaria uma palavra. A palavra é a unidade fonética mais natural, uma vez que nos comunicamos através de palavras. No entanto, à medida em que o vocabulário do sistema aumenta, o treinamento de modelos utilizando a palavra como unidade fonética vai se tornando pouco prático. Isso deve-se a basicamente dois fatores: pouca generalização e baixa treinabilidade. O termo generalização está relacionado à possibilidade de criar novas unidades fonéticas a partir de um conjunto pequeno de unidades. Já o termo treinabilidade está relacionado à disponibilidade de dados para a realização do treinamento de uma determinada unidade fonética. Visto que para treinar cada modelo são necessárias várias amostras de cada unidade fonética, seria necessária uma base de dados contendo diversas repetições de cada palavra. Tal base de dados é extremamente difícil de ser construída, uma vez que exigiria que o locutor pronunciasse uma quantidade enorme de sentenças a fim de cobrir todas as palavras do vocabulário que se deseja incluir no sistema de reconhecimento, em todos os contextos que as palavras puderem aparecer. Dizemos que a palavra é uma unidade fonética que fornece pouca generalização, pois não podemos formar palavras novas utilizando modelos de palavras já construídas, e é pouco treinável, pois é muito difícil conseguir uma base de dados que apresente todos os contextos (palavras vizinhas) de cada palavra do vocabulário do sistema.

Uma solução para o problema mencionado seria a utilização de unidades fonéticas

menores que a palavra, e, a partir delas, representar as palavras. Uma das primeiras alternativas nesse sentido é o uso de fonemas. O fonema é menor unidade fonética utilizada para construir palavras. Seu uso é interessante, pois como o seu número é reduzido, seria fácil treinar modelos para cada um deles. A execução mais simples de um fonema é denominada monofone, ou fonema independente do contexto. A título de exemplo pode-se representar todos os sons do português brasileiro com somente monofones. No Quatro 2 é apresentado o conjunto de monofones utilizados neste trabalho (Neto, Patrick, Klautau, & Trancoso, 2011), juntamente com exemplos de palavras que os empregam.

Quadro 2 – Lista dos fonemas utilizados nesse trabalho e exemplos de palavras que os empregam

Monofone	Exemplos
Vogais Orais	
A	lápis, jatobá, ábaco, capacete, cabeça, caça, lua, pedia
E	é, médico, pajé, épico, Pelé, pele, ferro, velho
e	capacete, resolver, respeito
i	justiça, pais, saia, lápis, idiota, aqueles, ele, pele
O	ópio, cópia, jogos, docas, sozinho, forte
o	resolver, jogo, golfinho, bolo, cor
U	baiacu, Raul, culpa, baú, cururu, logo, consolo, tijolo
Vogais Nasais	
a	avião, campeão, andar, tampar, canção, cama
e	então, consciência, tempo, bem, menos, dente
i	ninho, tinta, latina, importa
o	onda, campeões, somos, homem, fronha
u	um, muito, umbigo
Semi-Vogais	
w	natal, fácil, voltar, eu, chapéu, quase, jaula
j	fui, pai, sei, foi, caracóis, hotéis, micróbio, pátria

w	não, cão
j	muito, bem, parabéns, compõe
Fricativas não Sonoras	
f	festa, fanfarrão, afta, afluente
s	sapo, caçar, crescer, sessão, lápis, tórax, capaz, disco, casca, desço, excesso
S	chá, xaveco, cachorro
Fricativas Sonoras	
z	casa, coisa, quase, exato
v	vovó, vamos, avião
Z	geladeira, trovejar
Africativas	
tS	tia, pacote, constituinte, Tijuca
dZ	dia, cidade, disco
Plosivas	
b	barba, absinto
d	dados, cidade, dominar, administrar
t	todos, pato, constituinte
k	casa, casca, quero, quanto
g	guerra, gato, agüentar, agnóstico
p	papai, psicológico, apto, rapto
Liquidas	
I	laranja, palafita, leitão
L	calhar, colheita, melhor
R	carro, rua, rato, carga, germe
X	casar, certo, harpa, arco
r	carona, garoto, frango, graxa
Consoantes Nasais	
m	mamãe, ema, emancipar, marmota

n	nome, atenuar, encanação
j	casinha, galinha
Silêncios	
sil	silêncio no início/fim de uma sentença
sp	silêncio no meio de uma sentença (pausa)

Fonte: (Teruszkin & Vianna, 2006)

Diferentemente da palavra, os monofones são unidades fonéticas que oferecem muita generalização, ou seja, podemos construir qualquer palavra utilizando um pequeno grupo de monofones, e também é facilmente treinável, pois podemos facilmente encontrar um grupo de sentenças que apresente todos os monofones.

Entretanto, da mesma forma que as palavras, os monofones são muito afetados pelo seu contexto. Nos monofones o problema é ainda mais acentuado devido à sua duração, que é muito pequena, o que torna a sua aplicação pouco prática.

Uma vez que a palavra é uma unidade fonética que fornece pouca generalização e os monofones generalizam demais, é necessário um meio termo entre eles. Isso é conseguido com a utilização dos fonemas dependentes do contexto, sendo os seus representantes mais comuns os difones e os trifones. Os difones são fonemas dependentes do contexto somente da direita ou esquerda. Os trifones, por outro lado, são fonemas dependentes do contexto da esquerda e da direita. A título de exemplo consideremos o monofone hipotético x. As partir desse fonema pode-se formar um grupo de difones da forma $l - x$, onde l seria qualquer monofone à esquerda de x, e $x + r$, onde r seria qualquer monofone à direita de x. Da mesma forma pode-se formar o grupo de trifones $l - x + r$, onde l seria qualquer monofone à esquerda de x e r, qualquer monofone à direita de x.

Algumas unidades são consideradas livres de contexto, o que significa que não são formadas unidades dependentes do contexto a partir das mesmas e elas não aparecem em nenhum contexto. Um exemplo de unidade livre de contexto comumente utilizada é a pausa curta (*short pause* - sp). O sp é utilizado apenas para indicar o pequeno intervalo de silêncio que ocorre entre as palavras. Existem também as unidades independentes do

contexto, as quais podem aparecer no contexto de variáveis dependentes do contexto, mas não são construídas unidades dependentes do contexto a partir delas. Um exemplo de tal unidade independente do contexto é o silêncio (sil). Essa unidade é utilizada para indicar o intervalo de silêncio no início ou fim das sentenças.

2.6.3 Concatenação de modelos

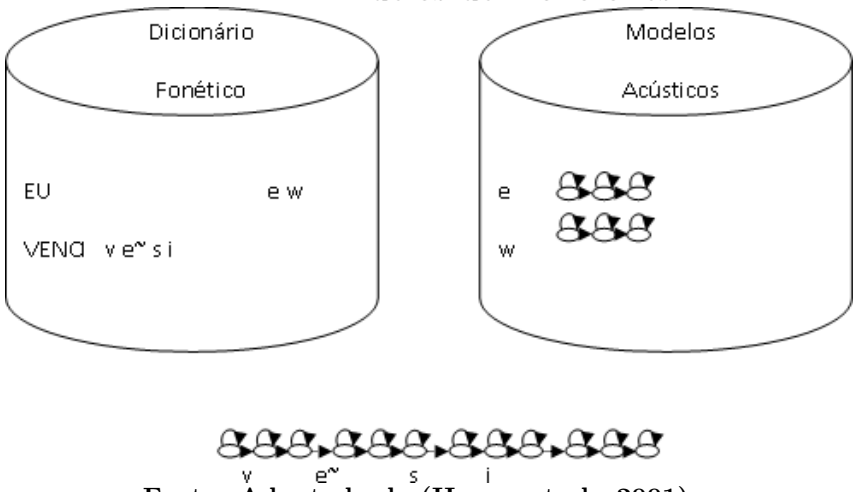
Tanto na fase de treinamento quanto na avaliação dos modelos HMMs utilizando unidades menores que a palavra, será necessária a realização da concatenação desses modelos de unidades menores (baseados em fonemas) para formar unidades maiores (palavras). Para a realização dessa tarefa, é utilizado um dicionário fonético com a transcrição de cada palavra em termos dessas unidades. Para facilitar a concatenação de modelos, geralmente são utilizados estados especiais de entrada e saída, os quais não emitem observação, e cujo único papel é servir como uma cola entre o estado final de um modelo e o estado inicial de outro.

Recorrendo a essa estrutura, é possível construir supermodelos para representar não somente palavras, mas também sentenças completas. Então cada sentença pode ser considerada simplesmente um grande HMM, e, por conseguinte, ser avaliada como uma palavra isolada.

Quando existe um HMM para cada palavra de uma sentença, a mesma pode ser formada pela substituição direta dos modelos das palavras. Quando as palavras são formadas por unidades menores, podem ser realizados dois tipos de expansão: a expansão intra-palavra e a expansão entre-palavras. Na expansão intra-palavra, todas as unidades fonéticas só podem ser formadas através dos contextos presentes dentro da palavra, podendo ser monofones, bifones ou trifones. Dessa forma, para formar uma sentença, cada palavra deve ser expandida de acordo com o dicionário fonético e então os modelos de palavras são juntados para formar os modelos para as sentenças. Na expansão entre-palavras são utilizados somente trifones e os monofones das palavras vizinhas são utilizados para a

formação dos trifones. Assim, para a expansão entre-palavras de uma sentença, é necessário considerar as palavras vizinhas da palavra a ser expandida, de forma a ser possível identificar qual o contexto a ser utilizado para formas os trifones nas bordas das palavras. Na Figura 1 é mostrado como uma palavra isolada pode ser formada a partir da concatenação de modelos de monofones.

Figura 1 – Formação de uma palavra isolada a partir da concatenação de modelos de monofones



Fonte: Adaptado de (Huang et al., 2001).

2.6.4 Treinamento do HMM para reconhecimento de discurso contínuo

Na formulação básica do HMM foi apresentada uma forma de estimar os seus parâmetros. No entanto, a forma apresentada assume que se conhecem as fronteiras da unidade a ser reconhecida, ou seja, as unidades devem estar segmentadas. No caso de discurso contínuo, onde são utilizadas unidades menores que a palavra, seria necessária a utilização de bases de dados contendo diversas sentenças segmentadas em termos dos monofones ou trifones. Tal abordagem não seria prática, uma vez se leva muito tempo para fazer essa segmentação manual da base de dados.

Uma solução para o problema é a utilização de uma versão alternativa ao algoritmo Bawn-Welch, a qual é denominada *Embedded* Bawn-Welch. O ponto positivo desse novo algoritmo é que não é necessário o conhecimento das fronteiras entre as unidades, pois elas serão determinadas através do alinhamento dos estados com a sequência de observação.

O algoritmo *Embedded* Bawn-Welch consiste na criação de supermodelos para cada sentença de treinamento, utilizando os modelos inicializados e o dicionário fonético. Então o algoritmo Bawn-Welch é aplicado nesse modelo composto e os parâmetros do HMM de cada unidade presente no modelo composto são atualizados de forma a criar novos modelos para cada unidade. Se os modelos resultantes forem melhores que os anteriores, isto é, apresentarem maior probabilidade de gerar a sentença modelada, o mesmo substitui o modelo anterior. Esse processo é repetido iterativamente até que os novos modelos não propiciem aumento de probabilidade, ou então um número máximo de iterações foi alcançado.

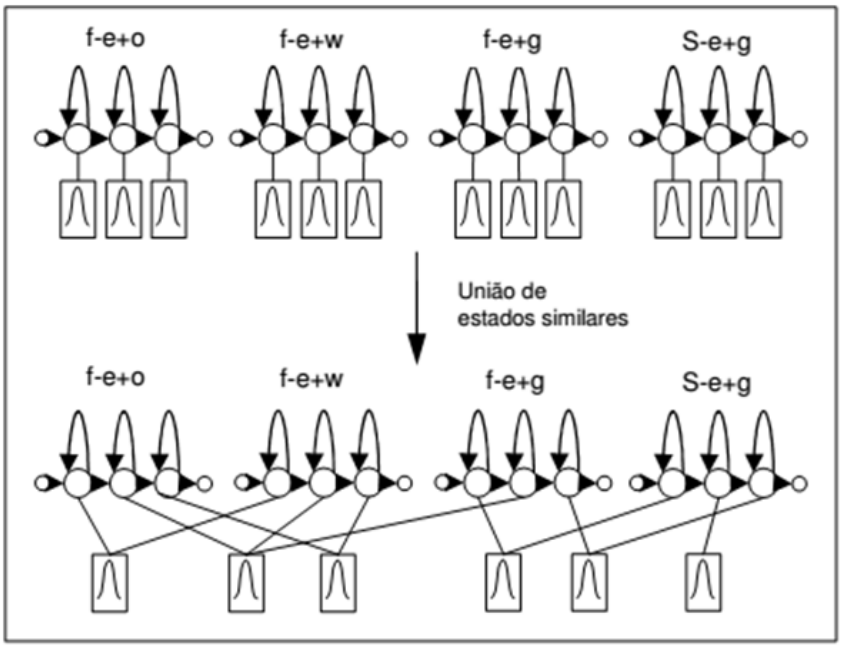
2.6.5 Compartilhamento de estados

Embora a utilização de fonemas dependentes do contexto forneça um equilíbrio entre a generalização e a treinabilidade, eles ainda não conseguem resolver o problema completamente. Na construção de sistemas utilizando expansão entre-palavras, por exemplo, por vezes faz-se necessário o treinamento de todos os possíveis trifones. Em uma conta simples, podemos concluir que para o caso de N monofones haverá um total de N^3 trifones. Para o caso do português brasileiro, onde geralmente são utilizados 38 monofones, chega-se a um total de 54.872 trifones. Embora seja um número muito grande de modelos, acontece que muitos trifones são muito infrequentes ou não ocorrem na prática e vários outros são foneticamente semelhantes entre si. Isso dificulta a estimação dos modelos para os trifones.

Uma técnica muito utilizada para lidar com o problema da escassez de dados para o treinamento de fonemas dependentes do contexto é o uso de estados compartilhados (Huang et al., 2001), ou união de estados (*state-tying*), conforme exemplo mostrado na Figura 2, que apresenta quatro trifones diferentes, mas todos associados ao monofone "E". A união de estados visa unir estados acusticamente indistinguíveis (também chamados senones). Com a união dos estados, trifones pouco frequentes podem ter seus estados

ligados a trifones mais frequentes, de forma a possibilitar a estimação eficiente de seus parâmetros. No exemplo da Figura 3, a priori, para cada estado de cada trifone oriundo do monofone "E", seria necessário modelar a fdp de emissão do vetor de atributos. Mas, após a aplicação do processo de união de estados, os trifones passam a compartilhar estados entre si, o que reduz a quantidade de parâmetros do modelo acústico a serem determinados.

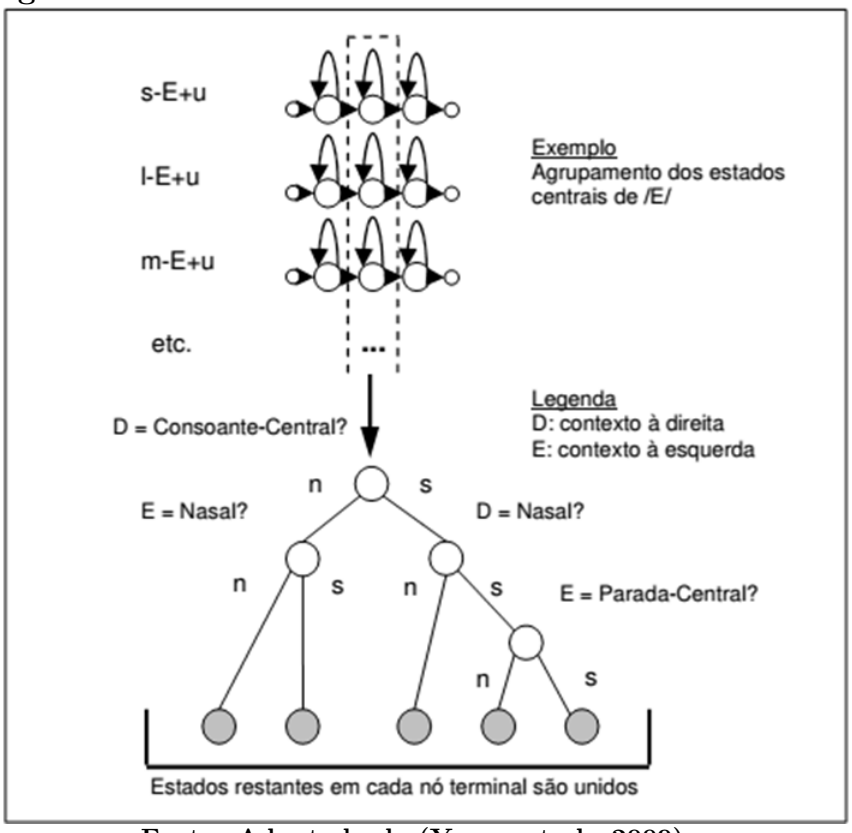
Figura 2 – União de estados foneticamente semelhantes



Fonte: Adaptado de (Young et al., 2009).

O método mais utilizado no processo de determinação de quais estados devem ser unidos é a arvore de decisão fonética. Esse método consiste em construir uma árvore binária para cada monofone. Em cada nó da árvore existe uma pergunta fonética cuja resposta é sim ou não, como por exemplo: "o contexto à esquerda é oral?". Inicialmente, todos os trifones oriundos de um mesmo trifone são posicionados no nó raiz da árvore. Então, para cada estado, os mesmo são divididos de acordo com as respostas às perguntas, até que não haja mais aumento de probabilidade decorrente da divisão dos grupos. A pergunta fonética utilizada para a divisão em cada nó é escolhida de forma a maximizar a diferença da soma das probabilidades dos conjuntos resultantes e a probabilidade do grupo antes da divisão. No final do processo, todos os estados que estão no mesmo nó terminal da árvore são unidos. Na Figura 3 é dado um exemplo desse processo, onde os estados centrais do fonema "E" são unidos. Nesse exemplo, os nós terminais são indicados pelos

Figura 3 – União de estados utilizando arvore de decisão



Fonte: Adaptado de (Young et al., 2009).

círculos escuros. Nota-se que o trifone "s-E+u" irá fazer parte do primeiro nó terminal, uma vez que o seu contexto à direita não é uma consoante-central e o seu contexto à esquerda não é nasal.

Além de poder ser utilizada para uma estimação robusta dos parâmetros das misturas de Gaussianas da função de probabilidade de emissão, as árvores de decisão permitem sintetizar trifones que nem ao menos apareceram na base de dados de treinamento, bastando para isso que esses trifones sejam colocados no topo da árvore e que sejam definidos os nós terminais aos quais ele pertencem, a partir da aplicação das perguntas fonéticas.

2.6.6 Reconhecimento de discurso contínuo

Como visto anteriormente, o HMM pode ser facilmente aplicado para o reconhecimento de palavras isoladas, através da avaliação de uma palavra de teste utilizando os cada modelo de palavra presente no vocabulário e então escolhendo o modelo que oferecer

a maior probabilidade para a palavra em teste. O modelo HMM de cada palavra pode ser construído utilizando a palavra como unidade fonética, ou então pode ser gerado a partir da concatenação dos modelos das unidades fonéticas (monofones ou trifones) que constituem a palavra, de acordo com o dicionário fonético, constituindo, dessa forma, um supermodelo. No reconhecimento de palavras isoladas é assumido que elas sejam pronunciadas pausadamente, de modo que o sistema pode identificar suas fronteiras de forma adequada.

Uma extensão natural do reconhecimento de palavras isoladas é o reconhecimento de palavras conectadas e, finalmente, o reconhecimento de discurso contínuo. Em tais sistemas não é necessária nenhuma atenção na forma de se pronunciar as palavras e o sistema fica responsável por determinar adequadamente as fronteiras entre as palavras, bem como as palavras contidas na sentença em avaliação.

De uma forma simplificada, para se identificar uma sequência de palavras em uma locução de teste é necessária a avaliação de todas as possíveis sequências de palavras. Para cada sequência possível é construído um HMM composto a partir da união dos modelos de monofones ou trifones para formar palavras, e das palavras para formar sentenças. É evidente que o tamanho do espaço de busca aumenta com o número de palavras no vocabulário. Em muitos sistemas, o número de palavras no vocabulário pode passar de dezenas de milhares, de modo que o espaço de busca se torna grande o suficiente para tornar o custo computacional proibitivo. No entanto, quando o número de estados é grande, ocorre de haver uma grande variabilidade na distribuição de probabilidades para os diferentes caminhos. De modo geral, a maioria dos estados possui probabilidade muito menor que a probabilidade máxima para cada instante de tempo. Tal fenômeno é explorado por uma técnica de redução do espaço de busca chamada Busca em Feixe (ou *Beam Search*)(Huang et al., 2001). Na Busca em Feixe, os estados que em um determinado instante apresentam verossimilhança menor que a verossimilhança máxima, menos um limiar, são descartados e não são considerados no restante do processo de busca. O limiar escolhido determina a largura do feixe de busca, ou o número de hipóteses mantidas para o restante do processo de busca. Uma escolha adequada desse limiar pode reduzir

consideravelmente o custo computacional, com pequeno ou nenhum acréscimo na taxa de erro do sistema. O processo de construção gradativa do espaço de busca, combinado com a técnica de poda por largura de feixe, é conhecido como *time syncronous Viterbi Beam Search* (Ney & Ortmanns, 1999) e forma a base dos algoritmos de reconhecimento de discurso contínuo baseados em HMM.

3 OTIMIZAÇÃO MULTIOBJETIVO

Neste capítulo será apresentada uma breve revisão a respeito da otimização multiobjetivo. Serão apresentados os conceitos de dominância e fronteira de Pareto e em seguida, será feita uma breve explanação acerca dos algoritmos evolucionários multiobjetivo, dando foco especial ao algoritmo genético multiobjetivo NSGA-II, empregado no contexto desta pesquisa.

3.1 Dominância e fronteira de Pareto

Como em (Soares, Guimaraes, Maia, Vasconcelos, & Jaulin, 2009), considerando as variáveis de decisão $\mathbf{x} \in \mathbf{X} \subset \mathbb{R}^n$, o vetor k-dimensional das funções objetivo é definido como $\mathbf{f}(\mathbf{x}) = \mathbb{R}^n \rightarrow \mathbb{R}^k$. Assim, um problema de otimização multiobjetivo sem restrições pode ser escrito como:

$$\min_{\mathbf{x} \in \mathbf{X}} \mathbf{f}(\mathbf{x}) \tag{75}$$

A solução da Equação 75 consiste em encontrar um conjunto de minimizadores $\mathbf{X}^*$, tal que:

$$\mathbf{X}^* = \{\mathbf{x}^* \in \mathbf{X} | \nexists \mathbf{x} \in \mathbf{X}, \mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}^*)\} \tag{76}$$

sendo o termo $\prec$ o operador de dominância usado para simplificar as comparações de vetores $\mathbf{f}(\mathbf{x}) \leq \mathbf{f}(\mathbf{x}^*)$e $\mathbf{f}(\mathbf{x}) \neq \mathbf{f}(\mathbf{x}^*)$. E a Fronteira de Pareto $\mathbf{Y}^*$ é definida como:

$$\mathbf{Y}^* = \mathbf{f}(\mathbf{X}^*) \tag{77}$$

Frequentemente não é possível avaliar todo $\mathbf{x} \in \mathbf{X}$. Assim, os algoritmos multiobjetivo buscam por uma aproximação da Fronteira de Pareto e soluções os mais uniforme-

mente distribuídas e espaças umas das outras.

3.2 Algoritmos evolucionários multiobjetivo

O termo algoritmos evolucionários (*Evolutionary Algorithms* – EA) está relacionado a uma classe de métodos de otimização estocásticos que simulam o processo de evolução natural (Zitzler, Laumanns, & Bleuler, 2004). Sua origem foi por volta da década de 1950 e pouco tempo depois foram propostas várias metodologias, dentre as quais, caracterizam-se como principais os algoritmos genéticos, a computação evolucionária, as estratégias de evolução e a programação genética. O ponto comum entre todas essas abordagens é que elas operam sobre um grupo de candidatos à solução (indivíduo ou cromossomo), que passam pelos processos de seleção e variação. O processo de seleção está relacionado ao princípio da competição pela reprodução e por recursos entre os indivíduos vivos, enquanto a variação está relacionada à capacidade de criação de novos indivíduos a partir de um grupo anterior, seja por recombinação (*crossover*), seja por mutação. A seguir serão apresentados os fundamentos dos algoritmos evolucionários, bem como uma breve descrição dos principais algoritmos evolucionários multiobjetivo encontrados na literatura.

3.2.1 Representação do indivíduo

Em algoritmos evolucionários, um indivíduo ou cromossomo é definido como um candidato à solução do problema de otimização. Existem várias formas de se representar um indivíduo, sendo que a representação adequada depende da natureza do problema. Dentre as possíveis formas de representação, as mais empregadas são a representação por cadeias binárias e a representação por vetores de números reais.

Na representação por cadeia binária, as variáveis de entrada são convertidas para

o formato binário e combinadas em uma ordem determinada para formar uma cadeia binária de comprimento fixo (Tabela 2).

Tabela 2 – Representação de individuos por cadeia binária

Cromossomo A	10101110111100010010
Cromossomo B	00110000111111000111

Fonte: Elaborado pelo autor.

Quando as variáveis, ou genes, podem assumir uma gama muito grande de valores, a representação por cadeia binária torna-se pouco viável. Em tais casos a representação dos indivíduos por vetores de números reais apresenta-se como uma boa alternativa. Na representação por vetores de números reais, cada gene de um indivíduo é representado por um número real de um vetor (Tabela 3).

Tabela 3 – Representação de indivíduos por vetores de números reais

Cromossomo A	17	456	9,9	34
Cromossomo B	45	38	3	71,1

Fonte: Eleborado pelo autor.

3.2.2 Função de aptidão

A função de aptidão (ou *fitness*) tem o objetivo de fornecer uma medida de qualidade dos indivíduos frente ao problema de otimização.

A função de aptidão mais natural é a própria função objetivo do problema. No entanto há situações onde a função de aptidão deve ser reconstruída. Um exemplo clássico é a conversão de um problema de maximização para um problema de minimização. Dessa forma pode-se minimizar uma função-objetivo $f(x)$ através da maximização da função $g(x) = -f(x)$, onde $g(x)$ é a função de aptidão.

3.2.3 Seleção dos indivíduos

Na teoria dos algoritmos evolucionários, a seleção está relacionada ao processo de seleção natural que ocorre entre os seres vivos. Nesse processo, os indivíduos mais aptos tem maior probabilidade de sobreviver e gerar descendentes. Como dito anteriormente, a função de aptidão é utilizada para determinar quais são os indivíduos mais aptos. Dentre vários métodos de seleção aplicáveis aos algoritmos genéticos, sendo os métodos da roleta e o torneio simples estão no grupo dos mais utilizados.

O método da roleta é uma forma de seleção proporcional, onde o indivíduo com melhor aptidão tem maior probabilidade de ser escolhido. A aptidão de cada indivíduo em uma geração é normalizada em termos do somatório S das aptidões de todos os indivíduos nessa geração e então é construída uma lista com aptidões relativas acumuladas dos indivíduos, conforme mostrado na Tabela 4.

Tabela 4 – Cálculo das aptidões relativas para o método da roleta

Indivíduo	1	2	3	4	5	6	7	8	9	10
Aptidão	8	2	17	7	2	12	11	7	3	7
Aptidão relativa (%)	10,5	2,6	22,4	9,2	2,6	15,8	14,5	9,2	3,9	9,2
Aptidão rel. acu. (%)	10,5	13,1	35,5	44,7	47,3	63,1	77,6	86,8	90,7	99,9

Fonte: Elaborado pelo autor.

Em seguida é sorteado um número aleatório r, a partir de uma distribuição uniforme, no intervalo $[0, S]$. O primeiro indivíduo que apresentar aptidão acumulada maior que r é selecionado. Na Figura 4 é mostrado de forma gráfica o processo de seleção pelo método da roleta. A porção da roleta ocupada por cada indivíduo é proporcional à sua aptidão relativa. Na Tabela 6 são mostrados alguns indivíduos selecionados a partir do método da roleta.

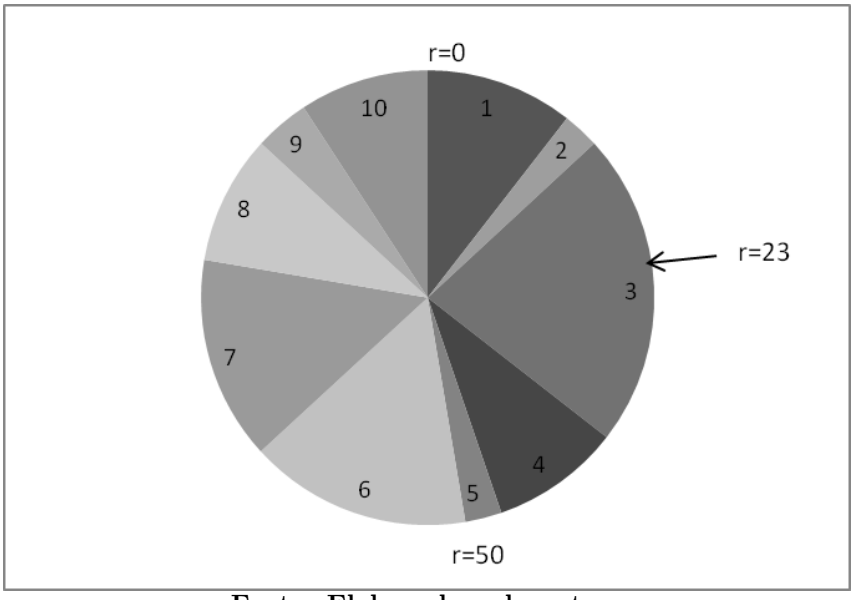
Tabela 5 – Seleção utilizando o método da roleta

Nº aleatório r	23	49	76	30	1	38	57
Indivíduo selecionado	3	6	6	2	1	4	5

Fonte: Elaborado pelo autor.

O método do torneio, por sua vez, opera sobre um grupo de indivíduos que, após serem sorteados aleatoriamente, têm suas aptidões calculadas. O indivíduo que tiver a

Figura 4 – Representação gráfica da seleção do indivíduo 3 a partir do método da roleta



Fonte: Elaborado pelo autor.

maior aptidão é selecionado para reprodução.

3.2.4 Reprodução

A reprodução é o processo que garante a renovação dos indivíduos das próximas gerações, mantendo algumas características adquiridas em gerações anteriores. A fim de imitar o processo de reprodução na natureza, são utilizados basicamente dois tipos de operadores: os operadores de recombinação (*crossover*) e os operadores de mutação. Esses operadores constituem os principais mecanismos dos algoritmos evolucionários para recombinar informação existente e inserir diversidade.

3.2.4.1 Operadores de Recombinação

O *crossover* consiste em utilizar um par de indivíduos pais para gerar um ou dois indivíduos filhos. Dentre os métodos utilizados, destacam-se os métodos: "1 ponto", "2 pontos" e "disperso (*scattered*)".

No método "1 ponto" é escolhido um número inteiro aleatório n, $1 \leq n \leq B$, em que B é o número de bits na sequência binária (quando utilizando cadeia de bits) ou então o número de variáveis (quando utilizando representação por vetores de números reais). O indivíduo filho é formado pela porção com índice $k < n$ do primeiro pai e $k > n$ do segundo pai.

No método "2 pontos" são escolhidos dois números inteiros aleatórios n e m, da mesma forma que no caso anterior. Nesse caso, o indivíduo filho é formado pelos elementos com índice $k \leq n$ do primeiro pai, elementos com índice $n + 1 \geq k \geq m$ do segundo pai, e elementos com índice $k > m$ novamente do primeiro pai.

No método "*scattered*", é gerada uma sequência binária aleatória de comprimento B. O indivíduo filho é formado tomando os elementos do primeiro pai nas posições correspondentes da sequência binária com valor 0 e os elementos do segundo pai nas posições correspondentes da sequência binária com valor 1.

3.2.4.2 Operadores de Mutação

A mutação consiste em realizar uma perturbação em um indivíduo a fim de evitar soluções locais. O operador de mutação é um operador unário, ou seja, é aplicado em somente um indivíduo. Quando a representação binária dos indivíduos está sendo utilizada, o método mais empregado de mutação consiste em selecionar aleatoriamente um bit do indivíduo e então tomar o seu complemento. Quando a representação por números reais é a adotada, a abordagem mais utilizada é a adição de um número aleatório a cada gene do indivíduo. Esse número é obtido geralmente de uma função gaussiana de média 0 e desvio padrão constante ou variável. Quando se opta por utilizar desvio padrão constante, o desvio padrão definido é utilizado em todas as gerações. Quando a opção é por se utilizar um desvio padrão variável, a abordagem mais empregada é definir um desvio padrão inicial e então esse valor é decrescido uniformemente até a ultima geração. A justificativa da utilização de desvio padrão variável é que, ao passar das gerações, as

soluções encontradas já estão próximas das soluções ótimas e necessitam, portanto, de pequenas modificações.

Se existem restrições nos valores que as variáveis que constituem o indivíduo podem assumir, o método de mutação factível adaptativo (*adaptive feasible*) (Kumar, 2010) pode ser utilizado. No método factível adaptativo, para cada indivíduo selecionado para mutação, é gerada de forma aleatória uma direção e uma largura de passo (perturbação) inicial. Após a realização da mutação, se o indivíduo resultante não estiver dentro da região factível (que obedeça às restrições impostas), a largura de passo é reduzida e então a mutação é realizada novamente. Esse processo é repetido até que o indivíduo resultante esteja contido na região factível.

3.2.5 Algoritmos evolucionários multiobjetivo baseados nos algoritmos evolucionários mono-objetivo

A otimização multiobjetivo utilizando algoritmos evolucionários multiobjetivo tem sido alvo de muitas pesquisas. A maioria dos algoritmos genéticos multiobjetivo encontrados na literatura podem ser considerados uma variante dos algoritmos genéticos mono-objetivo. Pode-se dizer que a principal diferença entre os algoritmos genéticos mono-objetivo e multiobjetivo encontra-se na forma como o operador de seleção trata as soluções não-dominadas e a forma de garantir a diversidade de soluções. Dentre os principais métodos de otimização multiobjetivo estão o VEGA (*Vector Evaluated Genetic Algorithm*) (Schaffer, 1984), o MOGA (*Multiobjective Genetic Algorithm*) (Fonseca, Fleming, et al., 1993), o NPGA *(Niched Pareto Genetic Algorithm*) (Horn, Nafpliotis, & Goldberg, 1994), o NSGA (*Nondominated Sorting Genetic Algorithm*) (Srinivas & Deb, 1994), o NSGA-II (Deb, Pratap, Agarwal, & Meyarivan, 2002) e o SPEA (*Strength Pareto Evolutionary Algorithm*) (Zitzler, 1999). A diferença básica entre esses algoritmos encontra-se na estratégia de seleção dos indivíduos.

Foi mostrado em (Deb et al., 2002), utilizando um conjunto de problemas de otimi-

zação multiobjetivo destinado à avaliação de algoritmos evolucionários, que o algoritmo NSGA-II geralmente apresenta soluções melhores que as conseguidas com os outros algoritmos avaliados. Esse fato, aliado à sua simplicidade de implementação, o tornou um dos algoritmos genéticos multiobjetivo mais populares e por isso ele vem sendo utilizado na solução de problemas de otimização em diversas áreas do conhecimento. Na seção seguinte, o algoritmo NSGA-II será descrito de forma detalhada, uma vez que o mesmo será utilizado neste trabalho.

3.2.6 NSGA-II

O NSGA-II (Deb et al., 2002) veio como uma evolução do NSGA (Srinivas & Deb, 1994). Ambos são implementações baseadas nas sugestões de Goldberg et al. (1989). O NSGA–II traz em sua estrutura diversos componentes desejáveis para um algoritmo de otimização multiobjetivo. Entre elas, citam-se: (i) um procedimento de ordenação dos indivíduos através do critério da dominância; (ii) uma estratégia simples e eficaz para garantir o elitismo, ou seja, dificultar que bons indivíduos sejam perdidos durante o processo de evolução e (iii) uma forma de manter uma diversidade próxima de uma distribuição uniforme, entre as soluções na fronteira de Pareto.

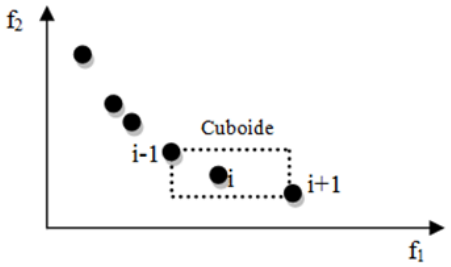
A ordenação utilizando o critério da dominância no NSGA-II vale-se do conceito dos vários níveis de soluções não dominadas. São calculadas duas quantidades: o contador de dominância n_p, que contabiliza o número de soluções que dominam a solução x_p, e S_p, o conjunto das soluções que a solução x_p domina.

As soluções que possuem o contador de dominância igual a zero farão parte do primeiro nível de não-dominância, F_1. Então, para cada solução com $n_p = 0$, ou seja, para cada solução x_p contida no nível F_1, encontram-se as soluções x_q pertencentes a S_p. Cada solução x_q será visitada e seu contador de não-dominância n_q será decrescido de um. Se o contador de não-dominância de algum elemento x_q se tornar zero, esse elemento é colocado em uma lista separada Q. Todos os elementos pertencentes a Q formam

então o segundo nível de não-dominância, F_2. O mesmo procedimento é realizado com os membros de Q e o terceiro nível é identificado. Esse processo continua até não existir nenhum elemento em Q.

A distância de agrupamento é uma medida que busca fornecer uma estimativa da densidade de soluções próximas a uma determinada solução. O cálculo da distância de agrupamento exige a ordenação prévia das soluções x_p pertencentes a cada nível de dominância. Para cada função-objetivo, aos indivíduos com maior e menor valor de saída, são atribuídas distâncias de agrupamento infinitas. Tal procedimento garante a preservação das bordas. Para os outros indivíduos, a distância de agrupamento é calculada como sendo a soma das distâncias entre seus vizinhos superior e inferior, em relação a cada função objetivo. A distância de agrupamento para a i-ésima solução pode ser representada pela área de um cuboide cujos vértices são formados pelas soluções vizinhas em relação a cada função, como mostrado na Figura 5.

Figura 5 – Cálculo da distância de agrupamento



Fonte: Adaptado de (Deb et al., 2002).

Uma vez definida a distância de agrupamento é possível definir o operador *crowded-comparison* ($\prec_n$), dizendo-se que uma solução $\mathbf{y}^1$ é preferível que uma solução $\mathbf{y}^2$($\mathbf{y}^1 \prec_n \mathbf{y}^2$) , se $\mathbf{y}^1$ está em um nível menor de não-dominância que $\mathbf{y}^2$. Se as duas soluções estão no mesmo nível de não-dominância, a solução que tiver a maior distância de agrupamento é preferível.

No loop principal do NSGA-II, primeiro é criada uma População Pai inicial,P_0, constituída de N elementos. A cada solução é atribuído um valor de aptidão (ou *ranking*) igual ao seu nível de dominância (o nível '1' é o melhor nível, seguido do nível '2', e assim por diante). Em seguida, assim como nos algoritmos genéticos mono-objetivo, são aplicados os operadores de seleção, reprodução e mutação, para criar uma População Filha (*offspring*) Q_0 de tamanho N, que será usada, conjuntamente com a população P0, para

a geração da população da próxima geração.

Para a determinação das populações das gerações seguintes a partir das populações P e Q na geração atual com a garantia do elitismo, primeiramente é formada uma população combinada $R_t = P_t \cup Q_t$, onde t é a geração corrente. A população R_t tem tamanho igual a $2N$. Em seguida a população R_t é ordenada de acordo com o critério de não-dominância. Então, se o número de elementos do primeiro nível de não-dominância F_1 for menor que N, todos os elementos de F_1 são escolhidos para formar a nova população P_{t+1}. O restante dos elementos de P_{t+1}são escolhidos a partir dos níveis de não-dominância subsequentes, F_2, F_3, e assim por diante, até não ser mais possível incluir conjuntos. Seja o conjunto F_l, o último conjunto depois do qual não seria mais possível acomodar conjuntos em P_{t+1}. Geralmente o número de elementos oriundos da união dos conjuntos F_1 a F_l é maior do que N. Então os elementos de F_l são ordenados em ordem descendente de acordo com o operador de *crowded-comparison* e os melhores indivíduos são escolhidos até completar o tamanho máximo da população.

Por fim são aplicados os operadores de seleção, *crossover* e mutação nos elementos da população P_{t+1}, de modo a formar a população filha Q_{t+1}.

3.2.7 Métricas de desempenho

Em muitas situações é desejável uma forma objetiva de comparação de duas soluções de um problema de otimização multiobjetivo. Uma dessas situações e quando se deseja comparar o conjunto de soluções encontrados por um algoritmo genético multiobjetivo, com a fronteira de Pareto real do problema. Em casos onde não se conhece a fronteira de Pareto real do problema, pode-se querer comparar os resultados de duas execuções do mesmo algoritmo ou execuções de algoritmos diferentes. Em problema com até três funções objetivo, é possível se ter uma ideia da proximidade das soluções a partir da observação dos gráficos das fronteiras de Pareto plotados na mesma figura. No entanto, com um número maior de variáveis essa análise se torna impraticável. A fim de solucio-

nar esse problema, vários pesquisadores desenvolveram métricas de desempenho para os algoritmos genéticos multiobjetivo, algumas das quais serão apresentadas nesse trabalho.

Segundo Zitzler, Deb, e Thiele (2000), os algoritmos evolucionários tem, alem dos objetivos propostos pelo problema, outros três objetivos:

a) encontrar um conjunto de soluções não dominadas o mais próximo possível da fronteira de Pareto;

b) encontrar um conjunto de soluções uniformemente espaçadas ao longo da fronteira;

c) maximizar os pontos extremos da fronteira não dominada para cada objetivo.

Deb et al. (2001), por outro lado, considera como meta extra para os algoritmos genéticos multiobjetivo apenas os dois primeiros itens citados anteriormente. Várias métricas foram propostas nesse sentido, sendo que no presente trabalho serão consideradas prioritariamente as métricas que podem ser utilizadas para medir a distância entre duas fronteiras de Pareto.

3.2.7.1 Métrica C

A métrica C, busca comparar dois conjuntos de soluções não dominadas $\mathbf{A}$ e $\mathbf{B}$ utilizando o critério da dominância. A métrica C é definida formalmente como se segue:

$$C_{\mathbf{A},\mathbf{B}} = \frac{|\{b \in \mathrm{B} | \exists a \in \mathbf{A} : a \prec b\}|}{|\mathbf{B}|} \tag{78}$$

Pode-se perceber que considerando dois conjuntos quaisquer $\mathbf{A}$ e $\mathbf{B}$, o resultado $C_{\mathbf{A},\mathbf{B}}$ não e necessariamente igual a $C_{\mathbf{B},\mathbf{A}}$. Observa-se que se cada solução de $\mathbf{B}$ for dominada por somente uma solução de $\mathbf{A}$, então $C_{\mathbf{A},\mathbf{B}} = 1$, e, quando nenhuma solução de B é dominada por alguma solução de $\mathbf{A}$, temos que $C_{\mathbf{A},\mathbf{B}} = 0$.

3.2.7.2 Métrica do Hipervolume

A métrica do Hipervolume é utilizada tanto para medir a proximidade de um conjunto de soluções não dominadas e a fronteira de Pareto, bem como a uniformidade da distribuição das soluções na fronteira de Pareto. Nessa métrica cada ponto do conjunto de soluções **A** é o vértice de um hipercubo hc_i. O outro vértice, considerando a diagonal principal, é um ponto fixo o qual pode ser a pior solução possível para o problema. O Hipervolume é então construído a partir da união dos hipercubos formados por cada solução do conjunto **A** como indicado pela Equação 79. É usual normalizar o Hipervolume encontrado utilizando um hipervolume de referência, o qual pode ser conseguido calculando o Hipervolume da fronteira de Pareto real, ou então a partir de um conjunto de soluções não dominadas **B**, como indica a Equação 80.

$$HV_{\mathbf{A}} = volume\left(\cup_{i=1}^{|A|} hc_i\right) \tag{79}$$

$$HVR_{\mathbf{A}} = \frac{HV_{\mathbf{A}}}{HV_{\mathbf{B}}} \tag{80}$$

4 OTIMIZAÇÃO DOS PARÂMETROS DE UM DECODIFICADOR DE FALA

A configuração dos parâmetros que controlam o processo de reconhecimento é muito importante para o desempenho geral do sistema. A maioria dos decodificadores de código aberto disponíveis na internet conta com um conjunto de parâmetros, que pode chegar a dezenas, que controlam processos intrínsecos ao reconhecimento de voz, tais como: (i) detecção de pontos terminais ou detecção de atividade vocal (*Voice Activity Detection* – VAD), responsável por identificar os instantes em que o sinal de voz começa e termina; (ii) busca pela melhor sequência de estados; (iii) ajuste da probabilidade do modelo de linguagem etc. Infelizmente esses parâmetros podem estar relacionados um ao outro de forma não linear e a sua alteração conjunta pode modificar tanto positivamente quanto negativamente o WER e o RTF. Na maioria das vezes os parâmetros são escolhidos de forma manual, através da experiência prévia do pesquisador, e o resultado da sua escolha no tocante ao reconhecimento é verificado a partir de experimentos. Embora seja possível obter resultados razoáveis através do ajuste manual, geralmente um ajuste ótimo dos parâmetros não é conseguido dessa maneira.

A forma mais direta de se ajustar os parâmetros do decodificador é através do ajuste individual de cada parâmetro. Nesse procedimento, o desempenho do decodificador é verificado após a modificação de um único parâmetro, mantendo-se os demais invariantes. Dessa forma, é possível traçar curvas para WER e RTF em função de cada um dos parâmetros do decodificador. El Hannani e Hain (2010) mostraram que os parâmetros relacionados ao processo de busca pela melhor sequência de estados possuem pelo menos uma relação monotônica com o RTF e com o WER, ou seja, o decréscimo no valor de um parâmetro alterado de forma independente só pode causar a redução ou nenhum efeito no RTF, podendo causar um aumento ou nenhum efeito no WER. Foi mostrado também que a alteração independente de alguns parâmetros podem ter maior ou menor efeito no WER e/ou RTF, dependendo dos valores dos outros parâmetros.

Uma vez que os parâmetros do decodificador apresentam uma grande correlação (Davenport, Schwartz, & Nguyen, 1999), o seu ajuste deve ser realizado de forma conjunta,

e não individualmente. Tal ajuste de forma conjunta pode se tornar bastante complexo, uma vez que cada parâmetro pode ter um faixa de valores distinta e pode ser representado por números inteiros ou reais. Uma forma direta de otimização conjunta dos parâmetros é a partir de uma busca em grade localizada, através de um procedimento de força bruta (El Hannani & Hain, 2010). Nesse caso, define-se uma faixa de valores para cada parâmetro, bem como a resolução da busca dentro de cada faixa, testando-se todos os possíveis conjuntos de valores. Em seguida, esses conjuntos são avaliados utilizando-se um conjunto razoável de sentenças de teste, sendo escolhido o conjunto ótimo de parâmetros de acordo com o WER e o RTF. Embora tal procedimento seja de simples implementação, os resultados dependem muito da faixa de valores e da resolução escolhida. Dependendo desses fatores, a avaliação de todas as possibilidades pode se tornar impraticável, e não existe garantia de que os resultados encontrados estejam em um ponto ótimo.

Visto que a busca em força bruta não é uma opção viável para a otimização conjunta dos parâmetros do decodificador, outras formas de otimização automática desses parâmetros devem ser buscadas. Há de ser considerado que o problema a ser resolvido é um problema de otimização multivariável, não linear e com restrições. Existem diversos métodos matemáticos que podem lidar de forma eficiente com tais condições, no entanto a maioria deles exige o conhecimento analítico da função objetivo. Em (Kacur & Korosi, 2007) foram utilizados algoritmos genéticos para a otimização de um conjunto de parâmetros do decodificador AVite, o qual está contido no conjunto de ferramentas ATK (Application Tool Kit for HTK) (Young et al., 2009). Os autores selecionaram os parâmetros que julgavam apresentar, de acordo com a experiência, um papel importante em um sistema de reconhecimento de fala contínua em diálogo. Dessa forma, foram escolhidos nove parâmetros relacionados ao algoritmo de detecção de fala, dois parâmetros relacionados aos atributos acústicos, bem como um parâmetro relacionado ao modelo de linguagem. Os experimentos foram realizados com a base de dados MOBILDAT-SK (Rusko, Trnka, & Darjaa, 2006). A base MOBILDAT-SK é uma base de dados constituída de 1.100 locutores, os quais falam a língua eslovaca. Cada locutor pronunciou 50 sentenças, as quais são constituídas por números, nomes, datas, quantias em dinheiro, comandos

e palavras foneticamente balanceadas. Para a avaliação dos parâmetros encontrados pelo método de otimização foi utilizada uma porção da base de dados constituída de repetições de sequências de dígitos de comprimento desconhecido, em que o comprimento refere-se à quantidade de dígitos pronunciados. Para o teste foram utilizados modelos acústicos construídos para trifones, com três estados por modelo e densidade de emissão no estado representada por misturas Gaussianas contendo oito componentes por estado. Como vetor de observações, foram utilizados os atributos MFCC, bem como suas derivadas de primeira e segunda ordem (denominadas componentes delta e aceleração, respectivamente), totalizando 39 componentes por vetor. Todos os parâmetros foram otimizados somente em função do WER, ou seja, foram utilizadas estratégias evolucionárias mono-objetivo. Os indivíduos foram representados através de vetores de números reais e, no mecanismo de mutação utilizado, o indivíduo sofria uma pequena alteração através de um processo estocástico como mostrado na Equação 81, onde x' representa uma mutação do vetor original x, que está passando pelo processo de otimização, e $N(0, \sigma)$ representa uma distribuição Gaussiana com média zero e desvio padrão σ.

$$x^{'} = x + N(0, \sigma) \tag{81}$$

Em complementação ao método de adição estocástica expressa pela Equação 81, onde foi utilizado um desvio padrão σ constante, foram utilizados dois outros métodos de mutação. No primeiro, o desvio padrão σsofreu decréscimos gradativos com o decorrer das gerações, começando em $0,35$ e chegando a $0,02$. No outro, o desvio padrão também passava pelo processo de evolução, como indicado pela Equação 82:

$$\sigma' = \sigma e^{N(0,\sigma_o)} \tag{82}$$

onde σ_o é um valor que se mantém constante ao longo das gerações, σ é o desvio padrão original e σ' é o desvio padrão após sofrer o processo de mutação. Através da abordagem mostrada foi conseguido um WER em torno de 5% abaixo do melhor conseguido quando os parâmetros foram ajustados manualmente. Foi mostrado também que o

método de mutação com decréscimo gradual do desvio padrão, expressado pela Equação 81, oferece melhores resultados ao final das gerações utilizadas.

A abordagem utilizada em (Kacur & Korosi, 2007) se mostrou capaz de encontrar melhores resultados, em termos do WER, que os conseguidos com o ajuste manual dos parâmetros do decodificador. No entanto não se preocupou com a carga computacional (que influi na métrica RTF), preocupando-se somente com os parâmetros que afetam predominantemente o WER. Entretanto, aqui se faz uma crítica: como o cenário de reconhecimento utilizado nos experimentos foi o de diálogo, o RTF deveria ter sido levado em consideração, uma vez que é assumido que um sistema de diálogo deva ser capaz de dar respostas em tempo real.

Em uma abordagem um pouco diferente da apresentada em (Kacur & Korosi, 2007), El Hannani e Hain (2010) apresentaram um método para otimização dos parâmetros do decodificador HDecode (Young et al., 2009), o qual faz parte de um pacote de código aberto distribuído como uma extensão do HTK. O foco é uma otimização multiobjetivo para a busca conjunta dos parâmetros do decodificador. A função objetivo, que é bidimensional, é constituída das métricas RTF e WER (sendo $R(\theta)$ e $W(\theta)$ os valores obtidos com o conjunto de parâmetros θ). Como se trata de um problema multiobjetivo, não é encontrada uma solução única, e sim um conjunto de soluções possíveis, as quais descrevem o WER mínimo para todos os possíveis valores de RTF. Tal conjunto de soluções é dado pela curva descrita pela Equação 83, em que a letra r é usada para designar todos os possíveis valores de RTF e $C(\theta)$ é a função objetivo a ser minimizada para um dado valor de RTF.

$$C_{opt}(r) = \min_{\theta:R(\theta)=r} C(\theta) \tag{83}$$

Uma solução para tal equação é pode ser dada por:

$$\theta_{opt}(r) = \arg\min_{\theta:R(\theta)=r} C(\theta) \tag{84}$$

A implementação direta da Equação 84 é impossível, uma vez que o conjunto

de parâmetros que leva a um RTF específico é desconhecido. Com isso, é utilizada a suposição de que os parâmetros relacionados à busca têm relação monotônica local com o custo computacional, ou seja, com a métrica RTF, para que se possa "rastrear" o conjunto de parâmetros ótimos.

Os parâmetros relacionados à busca são utilizados para fazer a poda dos estados pouco prováveis, portanto com pouca probabilidade de levarem à melhor sequência de estados, reduzindo assim o tempo necessário para se encontrar a sentença mais provável de ter sido pronunciada. Dessa forma, o ponto de partida do algoritmo de otimização pode ser a busca sem nenhuma heurística, ou seja, sem indicador que possibilite a eliminação de estados pouco prováveis. Para isso devem ser escolhidos valores altos para os parâmetros (que neste caso são limiares de medida de probabilidade, abaixo dos quais é realizada a poda), de forma a nenhum estado ser eliminado. Essa solução é reconhecida por ter o menor WER e o maior RTF. Partindo desse ponto, pode-se rastrear $\theta_{opt}(r)$, ou seja, o conjunto de parâmetros que minimiza o WER para cada valor específico de RTF. A cada iteração t, um novo conjunto de parâmetros é encontrado através da Equação 85:

$$\begin{aligned} \theta(t) &= \theta(t-1) - \alpha \left.\frac{\partial\theta}{\partial R}\right|_{t-1} \\ &= \theta(t-1) - \alpha k(t-1) \end{aligned} \tag{85}$$

com o gradiente $k(\cdot)$ e o controle da largura de passo (em relação ao RTF) α. Como essa função por si só não garante que o novo ponto ainda esteja na curva ótima, os parâmetros são escolhidos de forma a minimizar uma função de custo escolhida:

$$\hat{\theta}_{opt}(t) = \arg \min_{k(t-1)\in\left\{\frac{\partial\theta}{\partial R}:R(\theta)=r(t)\right\}} C(t) \tag{86}$$

Foram apresentadas duas funções objetivo: "minWER" e "razão". A função "minWER" implementa a minimização do WER como mostrado na Equação 87.

$$C_{\min WER}(t) = W(\hat{\theta}_{opt}(t-1) - \alpha k(t-1)) \tag{87}$$

onde W é o WER em função do conjunto de parâmetros.

A função "razão" foi escolhida de forma a encontrar o gradiente em termos da métrica RTF, como indica a Equação 88.

$$C_{razo}(t) = \frac{W\left(\hat{\theta}_{opt}(t-1) - \alpha k(t-1)\right) - W\left(\hat{\theta}_{opt}(t-1)\right)}{R\left(\hat{\theta}_{opt}(t-1) - \alpha k(t-1)\right) - R\left(\hat{\theta}_{opt}(t-1)\right)} \tag{88}$$

A largura do passo define a progressão em RTF através de uma função de decaimento, $\alpha(t) = \alpha(0)e^{(-\lambda t)}$, onde $\alpha(0)$ e λ são constantes positivas que definem a velocidade de progressão. Esse método foi escolhido de forma a fazer o decréscimo da largura de passo proporcional ao RTF, uma vez que as mudanças do WER aumentam com um baixo RTF. Os valores de $\alpha(0)$ e λ foram escolhidos de forma empírica como sendo, respectivamente, 2 e $0,5$.

Para o cálculo de $k(t-1)$ é utilizada uma aproximação. Em cada iteração, são calculados um número finito de candidatos à solução (conjunto de parâmetros) através da alteração do conjunto atual de parâmetros em somente uma dimensão, dando origem ao conjunto de parâmetros $\theta_i(t)$, onde i corresponde a uma dimensão. Então os gradientes são calculados a partir da Equação 89:

$$k_i(t-1) = \frac{\theta_i(t-1) - \hat{\theta}_{opt}(t-2)}{R(\theta_i(t-1) - R(\hat{\theta}_{opt}\,(t-2)} \tag{89}$$

O número de gradientes candidatos é igual ao número de parâmetros a serem otimizados. O procedimento pode ser resumido como se segue:

a) inicialização: Defina $\hat{\theta}_{opt}(0)$ e calcule $R_{opt}(0)$. Calcule $\hat{\theta}_i(1)$ através da perturbação de $\hat{\theta}_{opt}(1)$;

b) para cada iteração t, calcule as estimativas k_i utilizando a Equação 89, então use a Equação 86 para encontrar $\hat{\theta}_{opt}(t)$;

c) repita o passo 2 até $R(\hat{\theta}_{opt}(t))$ se tornar suficientemente pequeno.

Para os experimentos foram utilizados 39 atributos PLP (*Perceptual Linear Prediction*) (Hermansky, 1990). Foram construídos modelos para trifones, com estados compartilhados utilizando uma base de contendo 300 horas de duração, a qual foi construída

a partir da união das bases: (i) *Switchboard*-I, a qual é constituída de 2.400 diálogos por telefone, com 543 locutores; (ii) *Switchboard Cellular*, a qual é constituída de 190 locutores de vários gêneros e em diversas condições de ambiente, com os quais foram gravadas mais de 10 conversações com duração de cinco a seis minutos na rede de telefonia GSM e (iii) *Callhome English*, a qual é constituída de 120 conversações por telefone entre locutores nativos da língua inglesa. Os modelos de linguagem utilizados foram baseados nos modelos de linguagem *Broadcast News*. O vocabulário foi constituído das 10.000 palavras mais frequentes da lista de palavras dos modelos de linguagem. Os experimentos foram realizados em uma rede com três computadores com dois processadores dual core Intell 5160, e 8GB de memória RAM, de forma a ter 12 processadores em paralelo.

Foi mostrado que os resultados obtidos utilizando ambas as funções objetivo são praticamente idênticos para altos valores de RTF, apresentando alguma diferença na região de baixo RTF, em que a função objetivo "razão" obteve melhores resultados. Os resultados obtidos com ambos os métodos foram melhores que os melhores até então obtidos pelo ajuste manual dos parâmetros.

Embora tal abordagem tenha apresentado bons resultados na otimização dos parâmetros relacionados à busca, é necessária a escolha manual de um ponto inicial para o algoritmo, ou seja, um conjunto de parâmetros que leve ao menor WER, ou seja $\hat{\theta}_{opt}(0)$. Tal exigência impõe algum empecilho na utilização de tal procedimento para a otimização de outros parâmetros, uma vez que a identificação dos valores que levam ao menor WER geralmente não é trivial e não se sabe se o ponto inicial escolhido é um ponto ótimo.

5 FERRAMENTAS PARA CONSTRUÇÃO DE MODELOS ACÚSTICOS E DE LINGUAGEM E DECODIFICADORES DE FALA

Nos capítulos anteriores foram apresentados os fundamentos de reconhecimento automático de voz e de otimização multiobjetivo, assim como os principais trabalhos encontrados na literatura relacionados ao tema central desta pesquisa, que é a otimização dos parâmetros do decodificador através de algoritmos genéticos multiobjetivo para a minimização conjunta das métricas WER e RTF. Como contribuição técnica desta pesquisa, no presente capítulo será apresentado o processo de construção do modelo acústico e do modelo de linguagem utilizando ferramentas de código aberto distribuídas de forma gratuita na internet. Essas contribuições, que nos permitem fazer a transposição da teoria de reconhecimento de voz para as ferramentas disponíveis, são encontradas na literatura de forma muito dispersa. A carência dessas informações constitui um obstáculo enorme a ser vencido por grupos de pesquisa iniciantes. Portanto, são de relevância não só para o presente trabalho, o primeiro no Programa de Pós-graduação em Engenharia Elétrica da PUC Minas (PPGEE) em reconhecimento de voz, mas também para trabalhos de pesquisa futuros nesse tema. Neste capítulo também será apresentada uma breve descrição do decodificador de fala cujos parâmetros serão otimizados utilizando o método proposto.

5.1 HTK

Para a construção dos modelos acústicos foram utilizadas as ferramentas contidas no HTK (HMM *ToolKit*) (Young et al., 2009). O HTK consiste de um conjunto de ferramentas livres e abertas, destinadas à construção e avaliação de modelos acústicos baseados em HMM e também de modelos de linguagem. A literatura normalmente trata o processo de construção de um sistema de reconhecimento automático de fala dividindo-o em quatro etapas: preparação dos dados, treinamento, decodificação e análise dos resultados. O HTK utiliza essa mesma filosofia, e portanto, o que será mostrado nas subseções seguintes

são as ferramentas HTK, e os parâmetros relacionados, para cada uma dessas etapas.

5.1.1 Preparação de dados

Na fase de preparação de dados, todos os arquivos de áudio, com suas respectivas transcrições obtidas através do dicionário fonético, são pré-processados para servir de entrada para a fase de treinamento. O HTK fornece ferramentas que possibilitam a gravação de áudio, a criação e edição de transcrições e a conversão dos blocos de voz para seus respectivos vetores de atributos.

O processo de extração de atributos é realizado pela ferramenta **HCopy**. Para isso, a ferramenta recebe um conjunto de arquivos a serem convertidos, e realiza a conversão utilizando parâmetros informados através de um arquivo de configuração (todas as ferramentas do HTK suportam a passagem de parâmetros a partir de arquivos de configuração). Nesse arquivo são definidas características dos arquivos de entrada e de saída, tais como: tipo de arquivo de entrada (*Wavfile*, *Flac*, *Ogg* por exemplo), frequência de amostragem do sinal de voz, valor do coeficiente de pré-ênfase, tipo de janela (*Hamming*, *Hanning* etc), tamanho da janela (número de amostras de voz por bloco), taxa de blocos por segundo, tipo de atributo acústico e os parâmetros específicos de cada atributo. O **HCopy** suporta alguns dos mais utilizados atributos acústicos encontrados na literatura, tais como o MFCC, LPC e PLP.

Os arquivos de transcrições podem ser manipulados utilizando a ferramenta de edição de arquivos de transcrições **HLEd**. O **HLEd** suporta uma vasta gama de formatos de transcrições, oriundos de diversas bases de dados disponíveis. O principal papel do **HLEd** é converter os arquivos de transcrições de diversas bases de dados para um formato padrão do HTK. Além da conversão entre formatos de arquivos de transcrição, o **HLEd** também é utilizado para a formatação de transcrições, bem como para juntar diversos arquivos de transcrição em um só arquivo, criando assim o chamado arquivo de transcrição principal (*Master Label File* – MLF).

5.1.2 Treinamento dos modelos acústicos

O processo de treinamento dos modelos acústicos é o processo onde os parâmetros dos modelos HMM são ajustados. No HTK esse processo é realizado de forma gradual, construindo modelos simples e então melhorando esses modelos de forma gradativa em processos subsequentes. O processo de treinamento dos modelos para reconhecimento de discurso contínuo inicia-se pela criação de modelos protótipos para cada fonema independente do contexto (monofone). O protótipo é basicamente um arquivo de texto contendo a estrutura dos modelos relativos a cada monofone: número de estados do HMM (expressado indiretamente pelo tamanho do vetor de média e da matriz de covariância das componentes do vetor de atributos), número de componentes Gaussianas por estado (para expressar a densidade de probabilidade de emissão de símbolos por estado, ou seja, para encontrar a matriz **B** do modelo HMM), tipo de matriz de transição de estados (matriz **A** do modelo HMM) etc. Aqui cabe uma observação: sendo os atributos de voz descorrelatados (premissa dos atributos MFCC), a matriz de covariância é uma matriz diagonal e pode ser completamente definida pelos elementos da diagonal principal, que correspondem às variâncias dos coeficientes. Dessa forma, a matriz de covariância poderá ser substituída por um vetor de variâncias. Os valores de inicialização dos vetores de média e variância, bem como os elementos da matriz de transição, não são importantes. A única condição é que a soma dos elementos de uma mesma linha da matriz de transição seja igual a um. As transições que se deseje forçar como impossíveis (transição de um estado $q_t = s_i$ para o estado $q_{t+1} = s_j$, por exemplo) devem ser inicializadas com um elemento nulo, naquela posição da matriz (elemento a_{ij}). O HTK fornece duas ferramentas para inicialização dos modelos: **HInit** e **HCompV**. A ferramenta **HInit** é utilizada quando se deseja fazer a inicialização de modelos de unidades segmentadas, ou seja, unidades cujos arquivos de fala estão segmentados (com pontos terminais identificados). A ferramenta **HCompV**, por sua vez, permite inicializar os parâmetros do modelo HMM quando as unidades fonéticas não estão segmentadas nos arquivos de áudio. Após a inicialização, o próximo processo é a estimação dos parâmetros dos modelos. O HTK também oferece duas ferramentas

para a realização desse processo: **HRest** e **HERest**. A ferramenta **HRest** implementa o algoritmo Baum-Welch e é utilizada para a estimação dos modelos inicializados com o **HInit**, uma vez que é necessária a transcrição dos arquivos de treinamento contendo os pontos terminais de cada unidade fonética. A ferramenta **HERest**, por outro lado, implementa um algoritmo conhecido como algoritmo de Baum-Welch embutido (*Embedded* Baum-Welch). Nesse algoritmo as transcrições de cada sentença de treinamento são expandidas em termos de unidades menores (a princípio os monofones inicializados pelo **HCompV**), e então é construído um supermodelo a partir da concatenação dos monofones. O algoritmo de Baum-Welch é então aplicado sobre os supermodelos formados pela concatenação de monofones e então as novas estatísticas de todas as unidades fonéticas são estimadas de forma paralela, utilizando os dados de treinamento. O número de iterações a serem realizadas é um dos parâmetros do **HERest**. Esse número é predefinido em três iterações, mas de forma geral, a ferramenta é executada mais de uma a três vezes.

Para auxiliar o processo de treinamento são utilizadas as ferramentas **HDMan** e **HLEd**. A expansão das transcrições em termos de unidades fonéticas menores utiliza o dicionário fonético. A função desse dicionário é fornecer a transcrição de cada palavra presente nas sentenças de treinamento em termos de monofones ou trifones. Enquanto a ferramenta **HDMan** é utilizada para o gerenciamento e edição de dicionários, a ferramenta **HLEd** utiliza o dicionário fonético para a realização da expansão das transcrições em termos de unidades fonéticas. As duas ferramentas operam da seguinte forma: após o treinamento dos modelos dos monofones utilizando a ferramenta **HERest**, esses modelos são utilizados para a construção de modelos trifones. A ferramenta **HDMan** converte a representação das palavras do dicionário fonético de monofones para trifones, e a ferramenta **HLEd** utiliza esse novo dicionário, com representações em termos de trifones, para expandir as transcrições da base de dados de voz em sequências de trifones.

Outra ferramenta muito importante na fase de treinamento dos HMMs é o **HHEd**. A ferramenta **HHEd** é utilizada para edição dos modelos HMM, como clonagem de modelos, compartilhamento de estados, e aumento do número de Gaussianas por estado. Dessa forma, o **HHEd** é utilizado para construir os modelos de cada trifone que apare-

cer nas transcrições em termos de trifone criadas pelo **HLEd**, a partir do monofone dos quais os trifones foram gerados, processo esse que é conhecido como clonagem de modelos. Nesse processo a matriz de transição de estados é compartilhada, ou seja, todos os trifones formados pelos contextos de um mesmo monofone, compartilharão uma mesma a matriz de transição. Após a criação dos modelos trifones com matriz de transição compartilhada, utilizando a ferramenta **HHEd**, a ferramenta **HERest** é utilizada novamente para reestimar todos os modelos trifones utilizando a base de dados de treinamento.

Como a quantidade de modelos trifone é muito grande, o compartilhamento de estados é utilizado para unir estados de modelos com características fonéticas semelhantes, reduzindo de forma considerável o número de parâmetros a serem estimados. O procedimento de compartilhamento de estados semelhantes é realizado pela ferramenta **HHEd**, a qual utiliza um conjunto de perguntas fonéticas a respeito dos estados e estatísticas obtidas durante o treinamento, para a determinação de quais estados serão unidos. Após a realização do compartilhamento de estados, os modelos são novamente reestimados através da ferramenta **HERest**.

A última etapa de refinamento normalmente realizada é a adição de componentes na mistura de Gaussianas, da função da probabilidade de emissão. Para isso é utilizado o **HHEd** em conjunto com o **HERest**. O processo mais comum é aumentar de uma para duas componentes, e em seguida aumentar de duas em duas, até atingir o número desejado pelo usuário de componentes. A cada alteração do número de Gaussianas, os modelos devem ser reestimados utilizando a ferramenta **HERest**.

5.1.3 Decodificação

Os modelos acústicos podem ser avaliados a qualquer momento do processo de treinamento, ou seja, a partir do momento que os modelos foram inicializados, os mesmos já podem ser utilizados no processo de decodificação afim de se obter o seu WER e RTF. O HTK fornece dois decodificadores: **HVite** e **HDecode**. O **HVite** é voltada

ao reconhecimento de palavras isoladas e de discurso contínuo para pequenos e médios vocabulários. É compatível com gramáticas de estado finito e não suporta modelos de linguagem N-grama. O **HDecode**, por outro lado, é aplicada ao reconhecimento de discurso contínuo para vocabulários extensos. A sua principal diferença em relação ao **HVite** é que a **HDecode** suporta modelos de linguagem N-grama e implementa o dicionário fonético em árvore, para permitir estratégias de poda do espaço de busca mais eficientes.

Ambas as ferramentas recebem um conjunto de arquivos de áudio de teste, os modelos acústicos HMM, o dicionário contendo todas as palavras do vocabulário a ser utilizado, uma gramática de reconhecimento ou um modelo de linguagem (no caso do **HDecode**) e um arquivo de configuração que define diversas variáveis que controlam o comportamento do decodificador, como heurísticas de poda do espaço de busca, fatores aplicado aos modelos de linguagem, parâmetros do detecção automática de voz, entre outros. Como resultado, ambas as ferramentas fornecem um arquivo com as transcrições dos arquivos de áudio de teste. Esse arquivo é utilizado como entrada para a etapa de análise de resultados.

5.1.4 Análise de resultados

Nesta etapa, as transcrições obtidas pelo decodificador na etapa anterior são comparadas com as transcrições de referência para então gerar estatísticas acerca das taxas de acerto do sistema. A ferramenta do HTK responsável por essa etapa é o **HResults**. O **HResults** recebe tanto as transcrições de saída do decodificador quanto as transcrições de referência e, a partir delas, gera um arquivo com estatísticas de taxas de erro de acerto de reconhecimento de palavras e sentenças. Com os resultados fornecidos, é possível calcular o WER, o qual será utilizado como um dos indicadores de qualidade dos sistemas de reconhecimento de fala construídos neste trabalho.

5.2 SRILM

O SRILM (Stanford Research Institute Language Modeling toolkit) é um é um conjunto de ferramentas para criação e aplicação de modelos de linguagem estatísticos, principalmente destinados ao reconhecimento automático de voz, à etiquetagem estatística, à segmentação de texto e à tradução por máquina. O SRILM está em desenvolvimento no SRI desde 1995. É constituído de: (i) um conjunto de bibliotecas em C++ que implementam modelos de linguagem; (ii) estruturas de dados de apoio e utilidades de uso geral; (iii) um conjunto de códigos executáveis construídos sobre as bibliotecas em C++ para realizar tarefas tais como treinamento e teste de modelos de linguagem, etiquetagem ou segmentação de texto etc; e (iv) um conjunto de *scripts* que facilitam a realização de tarefas menores. O *tollkit* suporta modelos de linguagem N-grama e pode ser executado em Windows e Linux.

Sua utilização para construção de modelos N-grama é extremamente simples. A ferramenta **ngram-count** é utilizada para a construção de modelos tri-grama a partir de uma base de dados de treinamento, utilizando o modelo de retrocesso Katz Backoff para suavização. A perplexidade do modelo de linguagem construído pode ser avaliada em uma base de dados de teste, utilizando a ferramenta n-gram. Os modelos de linguagem construídos utilizando-se o SRILM são compatíveis com os com decodificadores fornecidos no HTK.

5.3 AVite

O **AVite** é um decodificador de fala para reconhecimento de discurso contínuo compatível com modelos acústicos HMM construídos no HTK e modelos de linguagem N-grama construídos no SRILM. É distribuído junto com o conjunto de bibliotecas de funções para construção de aplicações de reconhecimento automático de voz denominado ATK (*Application Tool Kit for* HTK) (*The ATK Real-Time API for HTK*, 2014). O

ATK foi construído tomando as bibliotecas do HTK como base, de modo a suportar grande parte dos recursos presentes no decodificador do HTK, o **HVite**. Algumas das diferenças do **AVite** em relação ao **HVite** é que a entrada do **AVite** precisa estar no formato WAV e somente suporta HMM com função densidade de probabilidade de emissão contínua, ao passo que o **HVite** suporta funções de densidade de probabilidade de emissão discreta e contínua. O **AVite** apresenta resultados bem semelhantes ao **HVite**, exceto quando utilizando modelos de linguagem n-grama, uma vez que a **HVite** não suporta tais modelos.

Assim como a **HVite**, a utilização do **AVite** exige uma lista com os arquivos de áudio a serem decodificados, os modelos acústicos, o dicionário fonético, a gramática de reconhecimento ou modelo de linguagem n-grama e um arquivo de configuração com os parâmetros que controlam o comportamento do decodificador. O resultado é um arquivo com as transcrições dos arquivos de áudio de entrada.

6 OTIMIZAÇÃO DOS PARÂMETROS DO DECODIFICADOR UTILIZANDO ALGORITMOS GENÉTICOS MULTIOBJETIVO

Como discutido no Capítulo 4, a maioria dos decodificadores de código aberto possui um grande número de parâmetros que devem ser ajustados para melhor eficiência do sistema de reconhecimento de voz em termos da métrica WER, ou da métrica RTF, ou ambas. Entretanto, nenhum desses parâmetros apresenta uma correlação monotônica linear com as métricas WER ou RTF. Na verdade, essas métricas dependem conjuntamente de vários parâmetros, tornando extremamente difícil realizar-se um ajuste manual desses parâmetros visando a um sistema de reconhecimento ótimo.

Por isso, têm sido discutidas várias propostas para a realização do ajuste de forma automática (Kacur & Korosi, 2007; El Hannani & Hain, 2010; Satoshi et al., 2012). Os trabalhos apresentados buscaram otimizar os parâmetros do decodificador em função somente do WER, ou então otimizar o WER para quaisquer valores de RTF. No cenário de reconhecimento de discurso contínuo, sobretudo quando se utiliza vocabulário extenso, muitas vezes é exigido um baixo WER, acompanhado de um RTF da ordem de 1 (aplicações em tempo real), outras no entanto exigem um WER modesto, no entanto com RTF menor que 1 (necessita processar uma sentença em tempo inferior ao à duração da mesma). Ambas as abordagens exigem então uma otimização multiobjetivo das métricas WER e RTF, de forma a possibilitar a escolha da configuração mais adequada para a aplicação.

Como principal contribuição científica deste trabalho, será proposta neste capítulo uma abordagem para a otimização conjunta dos parâmetros do decodificador **AVite**, a fim de obter um conjunto de soluções ótimas, que chamaremos de fronteira de Pareto, derivadas de diversas relações entre WER e RTF. O AVite foi escolhido como estudo de caso, mas o processo descrito pode ser aplicado a qualquer decodificador. Foi dada preferência ao **AVite**, em detrimento ao **HVite**, devido ao primeiro ser um pouco mais flexível e oferecer suporte a modelos de linguagem tri-grama .Para alcançar o objetivo do trabalho, foi utilizado o algoritmo genético multiobjetivo NSGA-II para gerar o conjunto

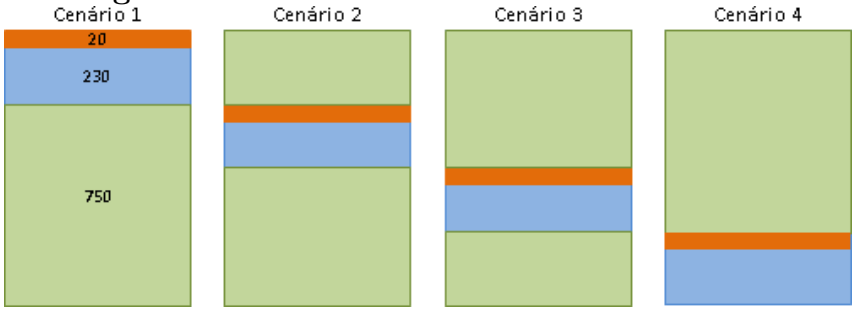
de soluções da fronteira de Pareto.

As subseções seguintes mostrarão os desenvolvimentos e as ferramentas utilizadas e delimitarão o escopo das configurações e parâmetros aplicados. Por fim, serão mostrados e discutidos os resultados dos experimentos realizados.

6.1 Construção dos modelos acústicos e de linguagem

Neste trabalho, o HTK foi utilizado para a construção do modelo acústico dependente do locutor a partir de uma base de dados de voz constituída de 1.000 sentenças diferentes, foneticamente balanceadas (Cirigliano et al., 2005), pronunciadas por um único locutor, amostradas com uma frequência de amostragem de 16kHz e gravadas sem utilização de compressão, no formato Microsoft WAV. A base de dados foi dividida em três partes: (i) 75% (750 sentenças) para treinamento do modelo acústico; (ii) 2% (20 sentenças) para o processo de otimização dos parâmetros do decodificador e 23% (230 sentenças) para a avaliação (teste) de desempenho do sistema de reconhecimento . Foram considerados quatro cenários distintos para as etapas de treinamento do modelo acústico, ajuste do decodificador e teste. Como indica a Figura 6, a diferença entre os cenários é que são utilizadas diferentes porções da base para treinamento do modelo acústico (cor verde), realização do processo de otimização do decodificador (cor laranja) e teste (cor azul).

Figura 6 – Divisão da base de dados acústica



Fonte: Elaborado pelo autor.

Foram criados modelos HMM utilizando trifones intra-palavra com compartilhamento de estados, baseados nos 40 monofones mostrados na Tabela II. Todos os modelos foram construídos utilizando-se 3 estados (além dos estados de entrada e saída, que não

emitem observações e são usados única e exclusivamente para a concatenação de modelos), exceto o modelo de pausa curta, o qual possui somente 1 estado. As funções de densidade de probabilidade de emissão foram modeladas utilizando mistura de densidades de probabilidade gaussianas com 16 componentes por estado, como indica (Teruszkin & Vianna, 2006). Foram utilizados vetores de atributos constituídos de 12 atributos MFCC utilizando o coeficiente de pré-ênfase de 0,97 e janelas *Hamming* de 25ms tomadas a cada 10ms. Juntamente com os 12 coeficientes MFCC foi adicionada a componente de energia e as derivadas de primeira e segunda ordem, formando vetores de 39 atributos.

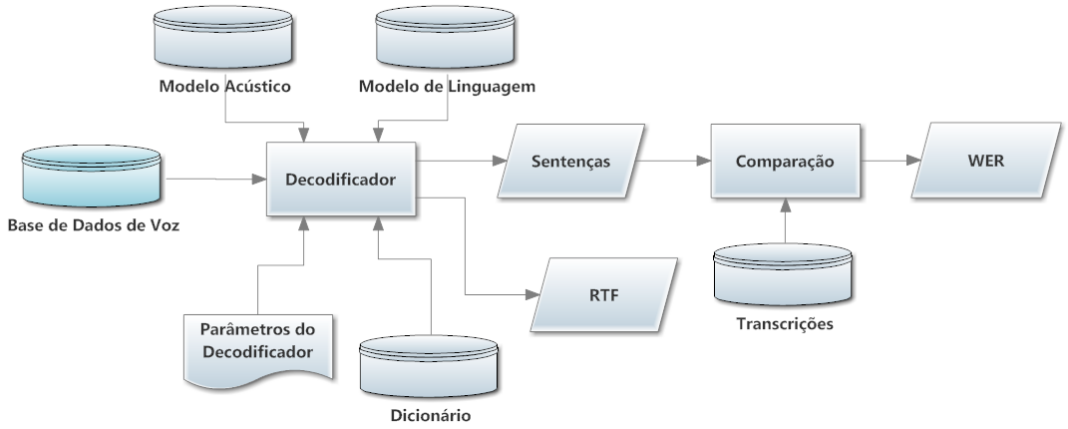
O HTK possui ferramentas para construção de modelos de linguagem, no entanto, para a construção dos modelos de linguagem nesta pesquisa foi utilizado o SRILM. O SRILM foi utilizado em detrimento ao HTK por possuir atualizações mais recentes e ser capaz de gerar modelos de linguagem melhores, ou seja, modelos com menor perplexidade. Dessa forma, o SRILM foi utilizado para construir modelos tri-grama a partir de uma base de dados constituída de 240.000 sentenças extraídas do jornal online Folha de São Paulo. O modelo de linguagem foi avaliado utilizando as transcrições da base de dados de voz, obtendo perplexidade de 253 para um vocabulário de 3.528 palavras.

6.2 Otimização multiobjetivo de parâmetros do decodificador

O método aqui proposto para a otimização multiobjetivo de parâmetros do decodificador consiste em encontrar valores de um subconjunto de parâmetros do decodificador AVite que minimizem simultaneamente as funções WER e RTF, encontrando um conjunto de soluções que pertençam à fronteira de Pareto. Na Figura 7 é mostrado um esquema simplificado do processo de teste (decodificação) dos modelos e da análise dos resultados.O módulo fundamental no processo de reconhecimento de voz é o decodificador propriamente dito. O decodificador recebe parâmetros de configuração e utiliza os modelos acústicos, o modelo de linguagem e o dicionário fonético para reconhecer a fala de um conjunto de sentenças de teste que não foram utilizadas na etapa de treinamento,

gerando a transcrição de cada arquivo de entrada. Essas transcrições são comparadas com as transcrições de referência e então é obtida a métrica WER do sistema. O tempo gasto durante o processo de decodificação é medido e usado para o cálculo da métrica RTF.

Figura 7 – Processo de teste e análise de resultados



Fonte: Elaborado pelo autor.

No presente trabalho o decodificador **AVite** será utilizado para a tarefa de reconhecimento de discurso contínuo dependente de locutor e com vocabulário extenso. O **AVite** possui um grande número de parâmetros que devem ser ajustados em função da eficiência requerida. Na Tabela 6 são mostrados os parâmetros que foram ajustados neste trabalho, bem como a faixa de valores definida para cada parâmetro (Young et al., 2009). Tais parâmetros estão relacionados ao processo de busca pela melhor sequência de estados ou ao modelo de linguagem e foram escolhidos para otimização no âmbito desta pesquisa por serem, dentre os parâmetros disponíveis (*The ATK Real-Time API for HTK*, 2014) no **AVite**, os parâmetros que têm maior influência nos resultados de um sistema para reconhecimento de discurso contínuo. Os parâmetros relacionados ao processo de busca são: NTOKS, GENBEAM, WORDBEAM e NBEAM. Tais parâmetros possuem maior influência no RTF, no entanto podem ter grande impacto no WER quando o RTF é muito baixo. Os parâmetros relacionados ao modelo de linguagem são: NGSCALE, LMSCALE e WORDPEN. Tais parâmetros são utilizados no cálculo da probabilidade a priori da sequência de palavras e podem influenciar tanto o WER quanto o RTF.

Esses parâmetros podem ser ajustados a partir da utilização de algum método de otimização monocritério ou multicritério. No âmbito desta pesquisa, selecionaram-se os algoritmos genéticos multiobjetivo para a otimização desses parâmetros. Nesse caso, a

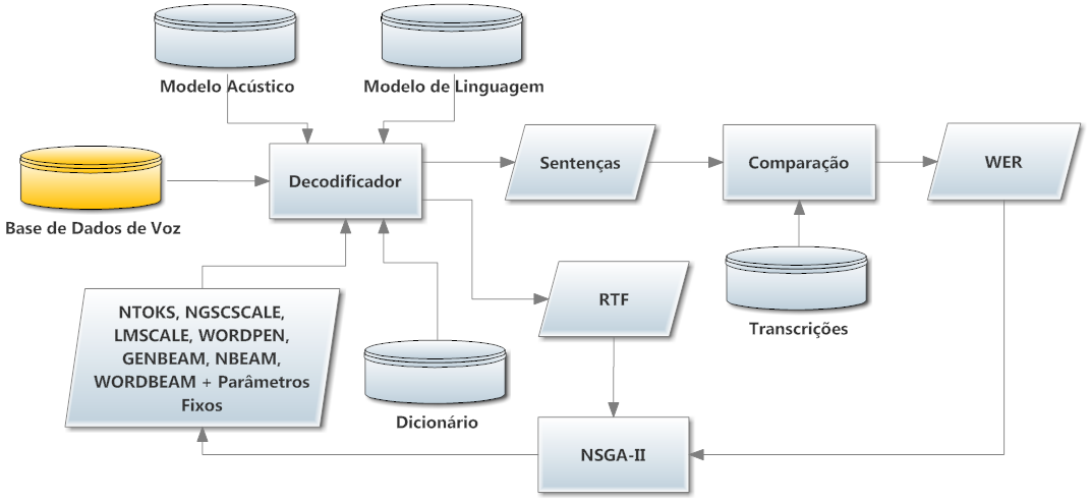
Tabela 6 – Parâmetros escolhidos para otimização e suas respectivas faixas de valores

Parâmetro	Faixa de valores	Tipo
NTOKS	1-15	Inteiro
GENBEAM	1-400	Real
WORDBEAM	1-400	Real
NBEAM	1-400	Real
NGSCALE	1-60	Real
LMSCALE	1-60	Real
WORDPEN	-40-0	Real

Fonte: Dados da pesquisa.

solução do problema retorna a fronteira de Pareto contendo o conjunto de soluções ótimas, a partir da minimização simultânea das métricas WER e RTF. A Figura 8 inclui o bloco responsável por encontrar a solução de Pareto para o problema em questão. No âmbito desta pesquisa, o algoritmo genético multiobjetivo NSGA-II foi o escolhido, por razões já mencionadas no Capítulo 3. Ressaltam-se aqui as características do algoritmo NSGA-II implementado: representação dos indivíduos utilizando vetores de números reais; processo de seleção utilizando o método do torneio simples; mutação a partir do método factível adaptativo e a recombinação pelo método scattered; taxa de recombinação de 80% e taxa de cruzamento de 20%.

Figura 8 – Diagrama do processo de otimização multiobjetivo dos parâmetros do decodificador Avite utilizando o algoritmo NSGA-II



Fonte: Elaborado pelo autor.

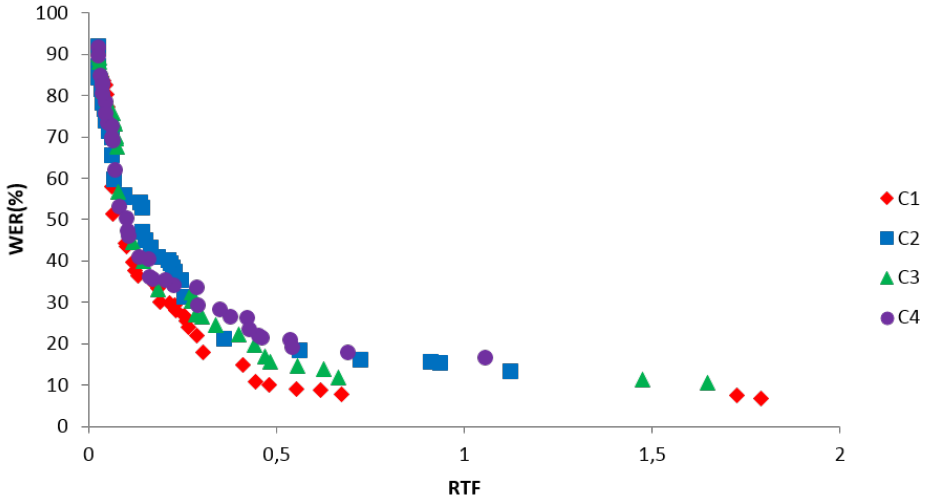
A diferença entre as Figuras 7 e 8, além da inserção do bloco de otimização, é que o conjunto de sentenças faladas que alimentam o decodificador é diferente. No primeiro caso, trata-se das sequências de teste (23% do total), ao passo que no segundo caso trata-

se das sequências utilizadas para a otimização do decodificador (2% do total). Como não existe uma forma analítica para as funções-objetivo WER e RTF, a avaliação de ambas as métricas é feita a partir do processo de decodificação.

Os parâmetros do decodificador determinados a partir da fronteira de Pareto identificada pelo NSAG-II são passados para o decodificador a partir de um arquivo de configuração. Como a variável NTOKS deve ser um número inteiro, a prática aqui adotada foi o arredondamento para o inteiro imediatamente igual ou inferior ao valor determinado pelo algoritmo de otimização.

Na implementação do algoritmo NSAG-II, foi definida uma população de 100 indivíduos e o processo de evolução foi realizado por 10 gerações. Foi definido um número pequeno de indivíduos e gerações devido ao fato que a avaliação de cada indivíduo demanda um tempo considerável, uma vez que é necessária a decodificação de 20 sentenças. Foi observado durante os experimentos realizados no âmbito desse trabalho, que há pouca variação na fronteira de Pareto a partir da sétima geração. O Gráfico 1 mostra as fronteiras de Pareto resultantes do processo de otimização multiobjetivo utilizando o algoritmo NSAG-II, para cada um dos quatro cenários ilustrados na Figura 6, ou seja, foram utilizados 750 sentenças para treinamento dos modelos acústicos e 20 sentenças para o processo de otimização, de forma que os resultados mostrados são referentes a avaliação dos indivíduos utilizando somente 20 sentenças.

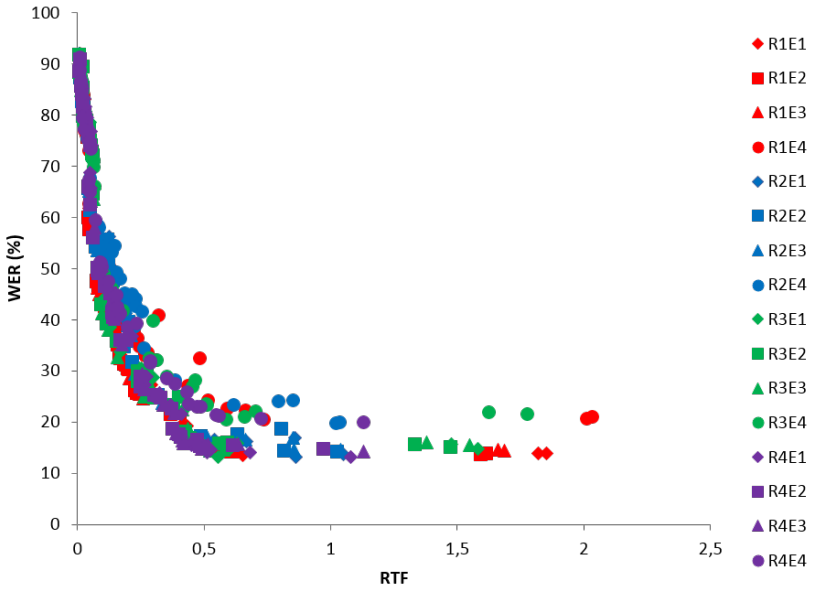
Gráfico 1 - Fronteira de Pareto



Fonte: Elaborado pelo autor.

Para avaliar a robustez dos resultados encontrados, os indivíduos pertencentes à fronteira de Pareto encontrados a partir de cada um dos quatro cenários (que se diferem pelo conjunto de sentenças utilizadas para treinamento do modelo acústico e para otimização dos parâmetros do decodificador) foram avaliados utilizando as sentenças de teste de todos os quatro cenários. O Gráfico 2 mostra os resultados obtidos. Na legenda do Gráfico 2, a letra R é utilizada para designar o cenário de treinamento do modelo acústico e realização do processo de otimização dos parâmetros (ou seja, para delimitar o conjunto de sentenças utilizadas para essas duas etapas) e a letra E é usada para designar o cenário de teste (ou seja, para designar o conjunto de sentenças utilizadas para teste).

Gráfico 2 - Avaliação dos indivíduos da Fronteira de Pareto



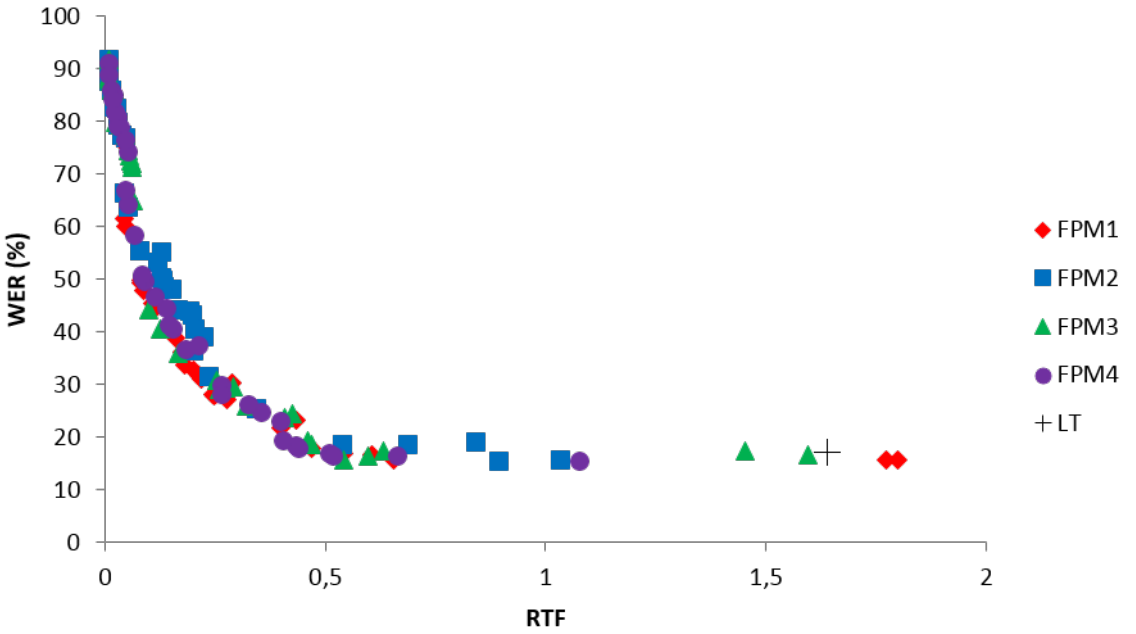
Fonte: Elaborado pelo autor.

Dessa forma, R1E1 refere-se aos resultados da avaliação dos indivíduos oriundos da fronteira de Pareto encontrada a partir do cenário C1, quando aplicados ao decodificador para reconhecer as sentenças de teste também do cenário C1. Da mesma forma, por exemplo, R2E4 refere-se aos resultados da avaliação dos indivíduos oriundos da fronteira de Pareto encontrada a partir do cenário C2, quando aplicados ao decodificador para reconhecer as sentenças de teste do cenário C4. De acordo com os cenários ilustrados na Figura 6, observa-se que os resultados RiEj, com $i = j$, originam-se da utilização de sequências de teste (representadas na Figura 6 pela cor laranja) que não fazem parte do conjunto de

sequências de treinamento. Entretanto, se $i \neq j$, todas as sequências de teste são usadas também para o treinamento. Exemplo: dividindo-se as sentenças da base de voz em quatro quartis, nos resultados R2E4 foram usadas para treinamento do modelo acústico (cenário C2) o 1º, 3º e 4º quartis do total da base de dados e para a otimização do decodificador, foram usadas as primeiras 20 sequências do 2º quartil. Para o teste (cenário C4), foram usadas as 230 últimas sequências do 4º quartil. Ora, todas essas sentenças já haviam sido usadas para o treinamento do modelo acústico.

Em seguida, tomaram-se os resultados médios relativos a cada i-ésimo cenário de treinamento (Ri), ou seja, foi calculada a média aritmética de cada indivíduo dos resultados RiE1, RiE2, RiE3 e RiE4 e o resultado, armazenado na variável FPRi (Fronteira de Pareto obtida com cenário de treinamento Ri) foi plotado no Gráfico 3.

Gráfico 3 - Resultados medios relativos a cada cenário de teste dos indivíduos da Fronteira de Pareto e resultado médio do conjunto de parâmetros encontrados na literatura



Fonte: Elaborado pelo autor.

O Gráfico 3 traz ainda o resultado médio da avaliação dos modelos referentes a cada cenário de teste (Ei) utilizando-se um conjunto de parâmetros encontrados na literatura para aplicações utilizando bases de dados semelhantes (P. Silva, Neto, & Klautau, 2009).

No Gráfico 3, esse resultado está indicado pela variável LT (LiTeratura). Vale a pena ressaltar que o resultado médio obtido com os parâmetros reportados na literatura, quando plotado Gráfico 3, é mapeado para um ponto único no gráfico, já que não se trata de uma fronteira de Pareto, que especifica um conjunto de soluções ótimas sob algum aspecto. Portanto, a fronteira de Pareto é uma das principais contribuições desta pesquisa, pois permite a escolha do par (WER,RTF) ótimo que seja mais adequado à aplicação, apontando, obviamente, quais são os parâmetros do decodificador que levam a esse desempenho do ASR.

A partir Gráfico 3, observa-se que com a abordagem adotada, que permite encontrar uma fronteira de Pareto, foi possível conseguir uma redução relativa de até 69% no RTF sem degradação do WER em relação ao resultado médio conseguido com parâmetros encontrados na literatura, tendo sido encontrado um indivíduo com RTF médio de 0,51 e WER médio 16,88%, ao passo que os resultados com parâmetros da literatura apresentam RTF médio de 1,64 e WER médio de 17,15%. O melhor WER encontrado foi 15,54%, com o respectivo RFT de 1,08, o que caracteriza uma redução de 9,4% no WER com uma redução de 34,1% no RTF em relação ao resultado médio conseguido com os parâmetros encontrados na literatura.

Para verificar a influência do grupo de treinamento dos modelos acústicos e realização do processo de otimização, as fronteiras de Pareto médias mostradas no Gráfico 3 foram comparadas entre si. Para essa comparação foram utilizas duas métricas de desempenho para algoritmos evolucionários multiobjetivo: a métrica C e a métrica do Hipervolume.

Como dito anteriormente, a métrica C compara duas soluções utilizando o critério da dominância. Como existem quatro fronteiras de Pareto diferentes, a métrica C foi aplicada a cada par individualmente. Os resultados são sumarizados na Tabela 7.

De acordo com os resultados mostrados na Tabela 7, pode-se dizer que a fronteira de Pareto FPM1 é melhor que as fronteiras de Pareto FPM2, FPM3 e FPM4, uma vez que $C_{FPM1,FPM2}$ é maior que $C_{FPM2,FPM1}$, $C_{FPM1,FPM3}$ é maior que $C_{FPM3,FPM1}$ e $C_{FPM1,FPM4}$ é maior que $C_{FPM4,FPM1}$. De forma similar pode-se dizer que a fronteira de

Tabela 7 – Resultado da aplicação da métrica C sobre as fronteiras de Pareto mostradas no Gráfico 3

$C_{\mathbf{A},\mathbf{B}}$	FPM1	FPM2	FPM3	FPM4
FPM1	0,31	1,77	1,17	0,88
FPM2	0,22	0,28	0,77	0,25
FPM3	0,34	0,91	0,51	0,48
FPM4	0,34	1,37	1,14	0,22

Fonte: Dados da pesquisa.

Pareto FPM4 é melhor que as fronteiras de Pareto FPM2 e FPM3.

A métrica do Hipervolume é usada para indicar a proximidade de um conjunto de soluções não dominadas, da fronteira de Pareto real ou de uma fronteira de Pareto de referência. Para comparar as fronteiras de Pareto mostradas no Gráfico 3 utilizando a métrica do Hipervolume é necessário definir qual vai ser a fronteira de Pareto de referência, e então calcular o valor da métrica para cada uma das outras fronteiras de Pareto. Na Tabela 8 são mostrados os resultados para cada fronteira de Pareto, utilizando cada uma das fronteiras como referência.

Tabela 8 – Resultado da aplicação da métrica do Hipervolume sobre as fronteiras de Pareto mostradas no Gráfico 3

HVR_A	FPM1	FPM2	FPM3	FPM4
FPM1-R	1,000	0,994	0,998	1,002
FPM2-R	1,006	1,000	1,004	1,008
FPM3-R	1,002	0,996	1,000	1,004
FPM4-R	0,998	0,992	0,996	1,000

Fonte: Dados da pesquisa.

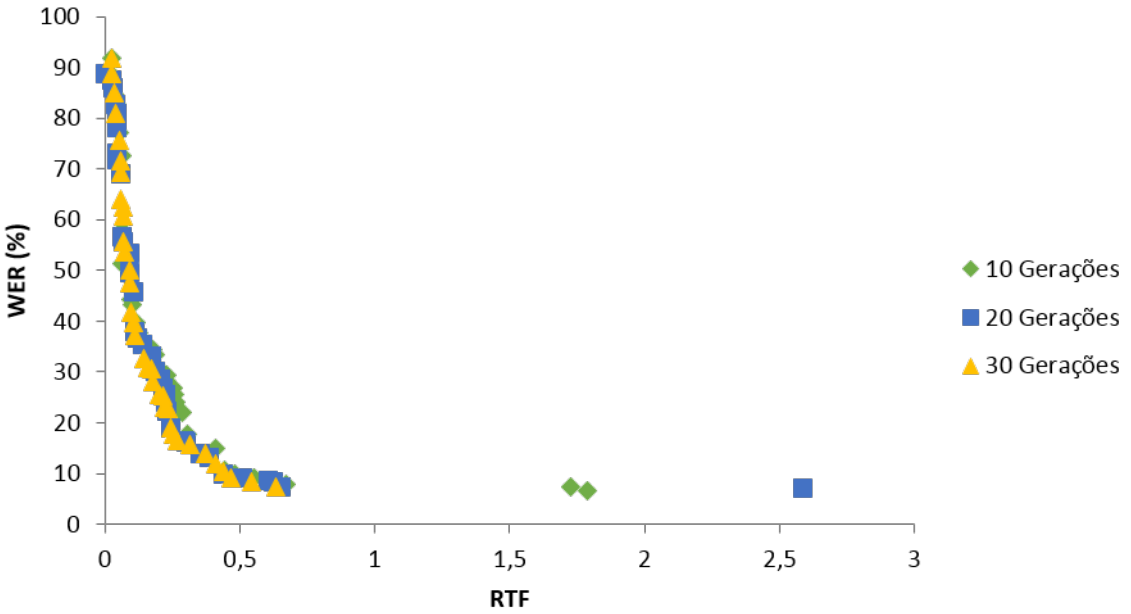
A partir da Tabela 8 é possível confirmar que a fronteira FPM1 é melhor que as fronteiras FPM2,FPM3, mas não se pode confirmar que é melhor que FPM4, uma vez que tomando a fronteira de Pareto 1 como referência, temos que HVR_{FPM2}, HVR_{FPM3} são menores que a unidade e HVR_{FPM4} é maior que a unidade. Observa-se que, segundo a métrica do Hipervolume, a fronteira FPM4 é melhor que as demais. De acordo com a métrica do Hipervolume, podemos concluir que embora a fronteira FPM1 e FPM4 sejam as melhores, podemos dizer que todas as fronteiras de Pareto são bem semelhantes, uma vez que todos os valores encontrados estão próximos da unidade. Dessa forma pode-se dizer que não há influência do Cenário de treinamento do modelos acústico e realização do processo de otimização sobre os resultados encontrados.

7 INFLUÊNCIA DO NÚMERO DE GERAÇÕES DO ALGORITMO GENÉTICO E DO NÚMERO DE COMPONENTES GAUSSIANAS DO MODELO ACÚSTICO SOBRE AS MÉTRICAS DE DESEMPENHO DO ASR COM DECODIFICADOR OTIMIZADO ATRAVÉS DO NSGA-II

No capítulo anterior foram considerados vários cenários para a otimização dos parâmetros do decodificador. Os cenários distinguem-se uns dos outros pela forma de dividir a base de dados acústica para o treinamento dos modelos, para o treinamento do algoritmo genético para a determinação dos parâmetros ótimos do decodificador e para o teste do ASR. Todos os ensaios foram realizados para uma configuração fixa do ASR, escolhendo-se o tipo de atributo acústico, o tipo de unidade fonética utilizada, o número de estados do HMM, a forma de compartilhamento de estados, o número de componentes Gaussianas nas funções de densidade de probabilidade de emissão de símbolos nos estados, o dicionário fonético e o tipo de modelo de linguagem, com base em resultados mostrados na literatura. No presente capítulo será avaliada a influência do número de gerações do algoritmo genético e do número de componentes Gaussianas utilizadas para compor a função de probabilidade de emissão, sobre as métricas de desempenho do ASR com decodificador otimizado através do NSGA-II.

Para a avaliação da influência do número de gerações utilizadas durante o processo de otimização dos parâmetros do decodificador, foi utilizado o modelo acústico construído utilizando a porção da base de dados de voz do Cenário 1 destinada ao treinamento do modelo acústico. Foi utilizada a mesma configuração descrita no capítulo anterior para a construção dos modelos acústicos e do modelo de linguagem. O processo de otimização foi realizado da mesma forma que no capítulo anterior. Foram realizados três processos de otimização dos parâmetros, considerando 10, 20 e 30 gerações no algoritmo genético NSGA-II. As fronteiras de Pareto obtidas são mostradas no Gráfico 4. No Gráfico 5 são mostrados os resultados da avaliação dos indivíduos do Gráfico 4 utilizando a porção da base de dados de voz do Cenário 1 destinada à avaliação dos modelos.

Gráfico 4 - Fronteiras de Pareto Obtidas utilizando 10, 20 e 30 gerações



Fonte: Elaborado pelo autor.

Como feito no Capítulo 6, as fronteiras de Pareto encontradas foram comparadas utilizando as métricas C e Hipervolume. Nas tabelas 9 e 10 se encontram os resultados da comparação dos conjuntos de soluções mostrados no Gráfico 5, utilizando a métrica C e do Hipervolume respectivamente.

Tabela 9 – Resultado da aplicação da métrica C sobre as fronteiras de Pareto mostradas no Gráfico 5

$C_{\mathbf{A,B}}$	G10	G20	G30
G10	0,22	0,26	0,20
G20	1,05	0,09	0,45
G30	0,71	0,14	0,03

Fonte: Dados da pesquisa.

Tabela 10 – Resultado da aplicação da métrica do Hipervolume sobre as fronteiras de Pareto mostradas no Gráfico 5

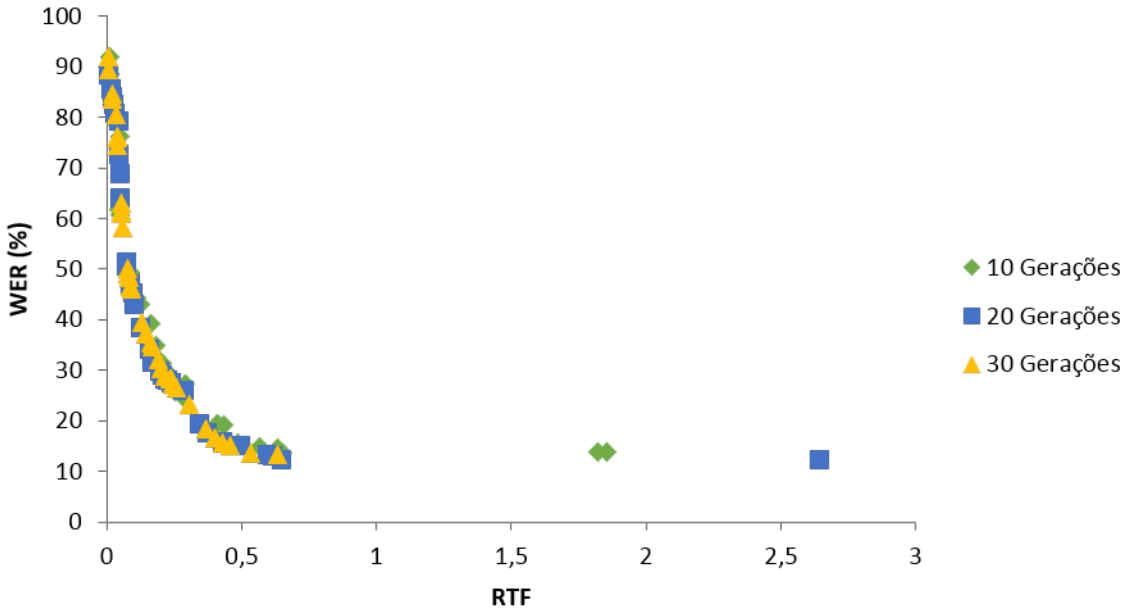
$HVR_{\mathbf{A}}$	G10	G20	G30
G10-R	1,00	1,02	1,00
G20-R	0,98	1,00	0,99
G30-R	1,00	1,01	1,00

Fonte: Dados da pesquisa.

Observando a Tabela 9 verifica-se que os resultados obtidos utilizando 20 gerações no processo de otimização dos parâmetros do decodificador são melhores que os resultados

obtidos utilizando 10 ou 30 gerações, uma vez que $C_{G20,G10}$ é maior que $C_{G10,G20}$ e $C_{G20,G30}$ é maior que $C_{G30,G20}$. Verifica-se também que os resultados conseguidos utilizando 30 gerações são melhores que os obtidos com 10 gerações. Com relação à métrica Hipervolume, confirma-se na Tabela 10 que os resultados utilizando 20 gerações no algoritmo genético são melhores que os conseguidos utilizando 10 ou 30 gerações, uma vez que os valores HVR_{G10} e HVR_{G30} são menores que a unidade quando tomando a fronteira de Pareto conseguida utilizando 20 gerações no algoritmo genético como referência. Os resultados mostrados no Gráfico 5 indicam que há variação no resultado da otimização dos parâmetros com o aumento do número de gerações, principalmente na nos valores extremos e RTF. No entanto, na prática os demais pontos se sobrepõem. Uma vez que para cada geração é necessária a avaliação do conjunto de sentenças de teste para cada indivíduo, considerou-se vantagem executar o procedimento de otimização com 10 gerações

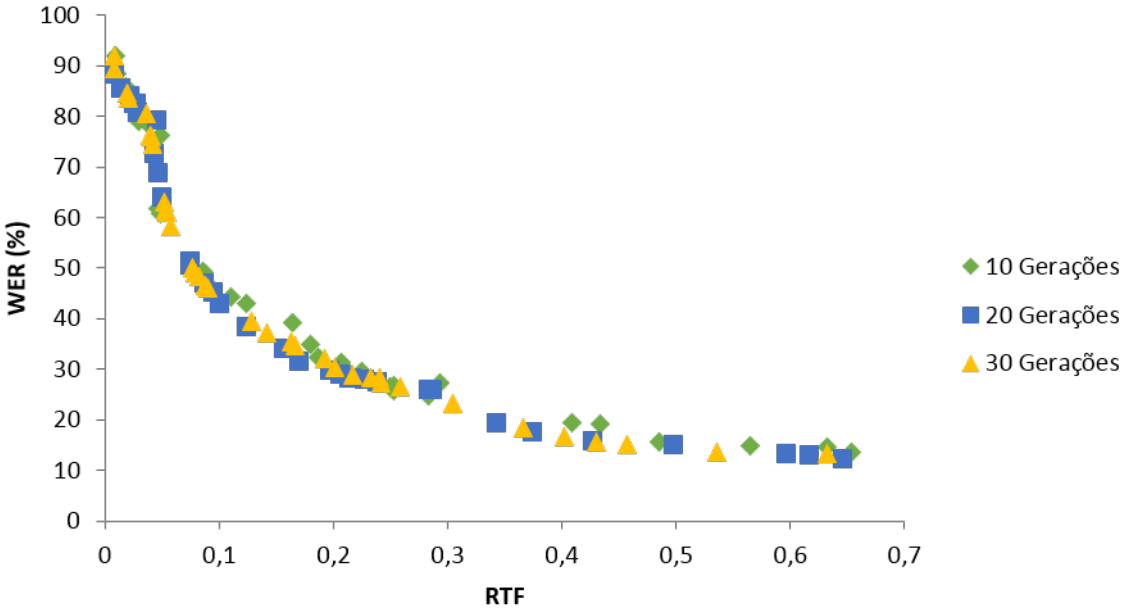
Gráfico 5 - Avaliação dos indivíduos das Fronteiras de Pareto mostradas no Gráfico 4



Fonte: Elaborado pelo autor.

Reduzindo-se a área de plotagem dos resultados do Gráfico 5 de forma a compreender somente resultados com valores de RTF menores que 0,75, é possível observar melhor a semelhança entre os resultados obtidos com a variação do número de gerações, o que pode ser visto Gráfico 6.

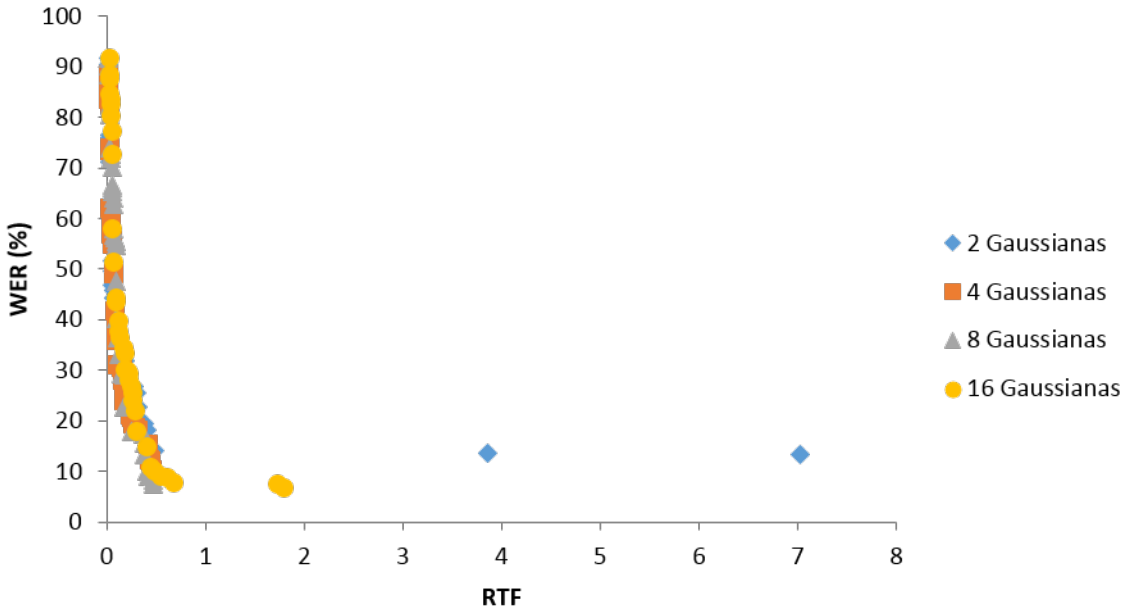
Gráfico 6 - Exibição dos resultados do Gráfico 5 somente para indivíduos com RTF menor que 0,75



Fonte: Elaborado pelo autor.

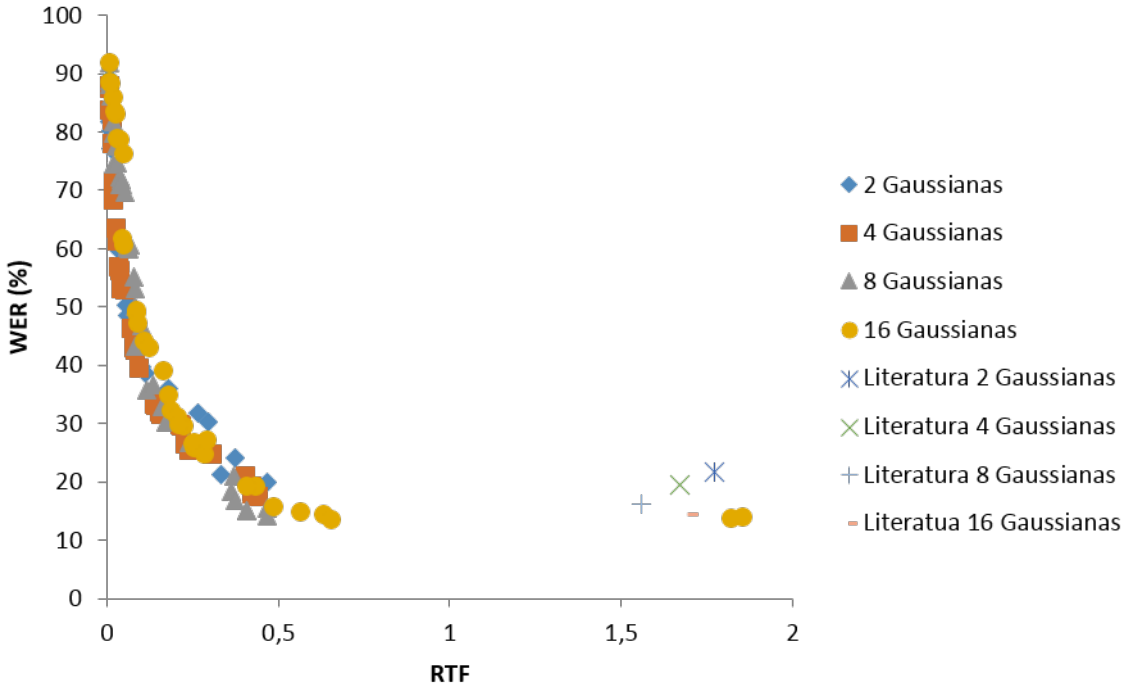
Para avaliar a influência do número de componentes Gaussianas nas funções de densidade de probabilidade de emissão sobre o desempenho do ASR com o decodificador otimizado conforme proposta desta pesquisa, foi considerada a mesma configuração descrita no capítulo anterior para a construção dos modelos acústicos e modelo de linguagem. Foram construídos modelos acústicos utilizando 2, 4, 8 e 16 componentes Gaussianas por estado. Para cada configuração de modelos acústicos foi realizado um processo de otimização dos parâmetros do decodificador. Todos os processos de otimização foram realizados utilizando 10 gerações. No Gráfico 7 são mostradas as Fronteiras de Pareto obtidas e no Gráfico 8 são mostrados os resultados da avaliação dos indivíduos das Fronteiras de Pareto utilizando a porção da base de dados de voz do Cenário 1 destinada à avaliação dos modelos. Juntamente com os resultados da avaliação dos indivíduos, das Fronteiras de Pareto, no Gráfico 8 também são mostrados os resultados da avaliação dos modelos utilizando o conjunto de Parâmetros encontrados na literatura.

Gráfico 7 - Fronteiras de Pareto obtidas utilizando 2, 4, 8, e 16 Gaussianas



Fonte: Elaborado pelo autor.

Gráfico 8 Avaliação dos indivíduos das Fronteiras de Pareto mostradas no Gráfico 7



Fonte: Elaborado pelo autor.

Assim como nos resultados anteriores, as resultados encontrados variando-se o número de Gaussianas por estado foram comparados a partir das métricas C e do Hipervolume. Verifica-se que, segundo a métrica C, os resultados obtidos utilizando 8 Gaussianas

por estados são melhores que os demais, visto que $C_{G4,G2}$ é maior que $C_{G2,G4}$, $C_{G4,G8}$ é maior que $C_{G8,G4}$ e $C_{G4,G16}$ é maior que $C_{G16,G4}$. A métrica do Hipervolume, por outro lado, não confirma tal constatação. De acordo com a métrica do Hipervolume, os melhores resultados são aqueles conseguidos quando utilizadas 16 Gaussianas por estado.

Tabela 11 – Resultado da aplicação da métrica C sobre as fronteiras de Pareto mostradas no Gráfico 8

$C_{\mathbf{A},\mathbf{B}}$	2G	4G	8G	16G
2G	0,40	0,27	2,22	1,85
4G	1,25	0,20	2,25	2,76
8G	0,65	0,37	0,42	1,52
16G	0,45	0,05	0,08	0,23

Fonte: Dados da pesquisa.

Tabela 12 – Resultado da aplicação da métrica do Hipervolume sobre as fronteiras de Pareto mostradas no Gráfico 8

$HVR_{\mathbf{A}}$	2G	4G	8G	16G
2G-R	1,00	1,03	1,07	1,07
4G-R	0,97	1,00	1,04	1,04
8G-R	0,93	0,97	1,00	1,01
16G-R	0,93	0,96	0,99	1,00

Fonte: Dados da pesquisa.

Para facilitar a visualização dos dados, os resultados da avaliação dos indivíduos da Fronteira de Pareto que apresentaram RTF maior que 2 foram suprimidos. Observa-se que com o método proposto foi possível encontrar indivíduos com melhor WER e RTF que os conseguidos com os parâmetros encontrados na literatura com qualquer número de Gaussianas. É possível observar também, que as soluções encontradas utilizando um número maior de Gaussianas não são as melhores para todo valor de RTF. De forma geral, soluções encontradas utilizando um número maior de Gaussianas apresentam melhor WER para os valores mais elevados de RTF. Com os parâmetros do decodificador otimizados utilizando-se a proposta desta pesquisa, a solução encontrada com menor WER (WER = 13,8%) foi obtida a partir do modelo acústico com 16 Gaussianas, com RTF igual a 1,8. A solução de menor WER (WER = 18,9%) para os resultados obtidos com os parâmetros do decodificador otimizados para o modelo acústico com 2 Gaussianas levou ao valor de RTF igual a 7,1. Dessa forma, comparando-se as duas situações (16 e 2 Gaussianas, respectivamente), observa-se uma redução de 26,4% no WER com respec-

tiva redução de 73,8% no RTF. Os resultados encontrados corroboraram as constatações empíricas mostradas na literatura, que diz que modelos acústicos construídos utilizando um número maior de componentes Gaussianas para representar a funções de densidade de probabilidade de emissão de símbolos nos estados, tendem a apresentar menor WER.

8 CONCLUSÕES

No presente trabalho foi utilizada a abordagem dos algoritmos genéticos multi-objetivo para a estimação dos parâmetros do decodificador AVite de forma a minimizar simultaneamente o RTF e o WER em sistemas de reconhecimento de fala contínua (ASR) para amplo vocabulário para o Português Brasileiro. Na busca desse objetivo, foi realizado um levantamento teórico apontando os principais pontos a serem considerados na construção desse sistema. Foram estudados os atributos acústicos mais eficientes para a representação do sinal de voz no ASR, tendo sido apresentados os atributos LPC e MFCC. Foi apresentada a modelagem acústica utilizando os modelos ocultos de Markov, bem como os detalhes práticos para a implementação de sistemas de reconhecimento de palavras isoladas e de discurso contínuo. Foi apresentada a teoria básica utilizada para a construção dos modelos de linguagem n-grama, bem como as principais variantes dos algoritmos utilizados para suavização dos modelos de linguagem. Foi realizada um incursão acerca da otimização mono e multi-objetivo, e foram apresentados os principais trabalhos no tocante à otimização dos parâmetros de um decodificador. A abordagem proposta para otimização conjunta e multiobjetiva dos parâmetros do decodificador, que caracteriza a principal contribuição científica desta pesquisa, foi apresentada e os resultados práticos obtidos foram mostrados. Com a abordagem utilizada foi possível encontrar um conjunto de soluções (cada solução corresponde a um conjunto de parâmetros do decodificador) que apresentam relações ótimas entre WER e RTF. Assim, pode ser escolhido o conjunto de parâmetros que leve ao resultado que seja mais adequado à aplicação em termos do RTF e do WER. Os resultados encontrados a partir da abordagem apresentada foram comparados com o resultado utilizando parâmetros encontrados na literatura. Foi mostrado que com a abordagem apresentada, quando utilizando modelos acústicos com 16 Gaussianas por estado e utilizando 10 Gerações no processo de otimização, foi possível conseguir uma solução com resultados médios que apresenta uma redução de até 68,7% no RTF, sem que haja degradação do WER em comparação com os resultados médios obtidos com parâmetros encontrados na literatura. Foi também conseguida uma solução que apresenta uma

redução de até 9,6% nos WER, com uma redução de aproximadamente 50% no RTF em comparação com resultados médios obtidos utilizando parâmetros encontrados na literatura. Foi mostrado que bons resultados podem ser conseguidos com apenas 10 gerações do processo de otimização. A influência do número de Gaussianas nas métricas WER e RTF, com decodificador otimizado, foi verificada. Os resultados encontrados corroboraram as constatações empíricas mostradas na literatura, que diz que modelos acústicos construídos utilizando um número maior de componentes Gaussianas para representar a funções de densidade de probabilidade de emissão de símbolos nos estados, tendem a apresentar menor WER. A maior vantagem do método de otimização dos parâmetros do decodificador apresentado em relação aos métodos apresentados na literatura é a possibilidade de realizar uma otimização conjunta dos parâmetros do decodificador em relação a múltiplos objetivos, sem que o usuário precise fornecer um ponto de partida para o algoritmo. Pelo exposto, considera-se que os objetivos do trabalho apresentado foram cumpridos de forma satisfatória.

REFERÊNCIAS

Alcaim, A., & Oliveira, C. A. D. (2011). *Fundamentos do processamento de sinais de voz e imagem.*

Atal, B. S., & Hanauer, S. L. (1971). Speech analysis and synthesis by linear prediction of the speech wave. *The journal of the acoustical society of America*, *50*(2B), 637–655.

The ATK real-time API for HTK. (2014). Retrieved from `http://mi.eng.cam.ac.uk/research/dialogue/atk_home`

Cirigliano, R. J. R., Monteiro, C., Barbosa, F., Resende Junior, F. G. V., Couto, L. R., & Moraes, J. A. (2005). Um conjunto de 1000 frases foneticamente balanceadas para o português brasileiro obtido utilizando a abordagem de algoritmos genéticos. In *Anais do xxii simposio brasileiro de telecomunicacoes (sbrt 2005)* (pp. 544–549).

Davenport, J., Schwartz, R., & Nguyen, L. (1999). Towards a robust real-time decoder. In *Acoustics, speech, and signal processing, 1999. proceedings., 1999 ieee international conference on* (Vol. 2, pp. 645–648).

Davis, S., & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, *28*(4), 357–366.

Deb, K., et al. (2001). *Multi-objective optimization using evolutionary algorithms* (Vol. 2012). John Wiley & Sons Chichester.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, *6*(2), 182–197.

El Hannani, A., & Hain, T. (2010). Automatic optimization of speech decoder parameters. *Signal Processing Letters, IEEE*, *17*(1), 95–98.

Fonseca, C. M., Fleming, P. J., et al. (1993). Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization. In *Icga* (Vol. 93, pp. 416–423).

Gaikwad, S. K., Gawali, B. W., & Yannawar, P. (2010). A review on speech recognition technique. *International Journal of Computer Applications*, *10*(3), 16–24.

Goldberg, D. E., et al. (1989). *Genetic algorithms in search, optimization, and machine learning* (Vol. 412). Addison-wesley Reading Menlo Park.

Hermansky, H. (1990). Perceptual linear predictive (plp) analysis of speech. *the Journal of the Acoustical Society of America*, *87*(4), 1738–1752.

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. In *Evolutionary computation, 1994. ieee world congress on computational intelligence., proceedings of the first ieee conference on* (pp. 82–87).

Huang, X., Acero, A., Hon, H.-W., et al. (2001). *Spoken language processing* (Vol. 18). Prentice Hall Englewood Cliffs.

Itakura, F. (1975). Minimum prediction residual principle applied to speech recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, *23*(1), 67–72.

Kacur, J., & Korosi, J. (2007). An accuracy optimization of a dialog asr system utilizing evolutional strategies. In *Image and signal processing and analysis, 2007. ispa 2007. 5th international symposium on* (pp. 180–184).

Kumar, R. (2010, February 9). *System and method for the use of an adaptive mutation operator in genetic algorithms.* Google Patents. (US Patent 7,660,773)

Neto, N., Patrick, C., Klautau, A., & Trancoso, I. (2011). Free tools and resources for brazilian portuguese speech recognition. *Journal of the Brazilian Computer Society*, *17*(1), 53–68.

Ney, H., & Ortmanns, S. (1999). Dynamic programming search for continuous speech recognition. *Signal Processing Magazine, IEEE*, *16*(5), 64–83.

Prabhakar, O. P., & Sahu, N. K. (2013). A survey on: Voice command recognition technique. *International Journal of Advanced Research in Computer Science and*

Software Engineering, *3*(5).

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Rabiner, L., Levinson, S. E., Rosenberg, A. E., & Wilpon, J. G. (1979). Speaker-independent recognition of isolated words using clustering techniques. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, *27*(4), 336–349.

Rusko, M., Trnka, M., & Darjaa, S. (2006). Mobildat-sk-a mobile telephone extension to the speechdat-e sk telephone speech database in slovak. In *Proc. of the 11 th international conference speech and computer, specom* (pp. 485–488).

Satoshi, K., Takaaki, H., Yoshikazu, Y., Taichi, A., Hirokazu, M., & Satoshi, T. (2012). Efficient prior and incremental beam width control to suppress excessive speech recognition time based on score range estimation. In *Spoken language technology workshop (slt), 2012 ieee* (pp. 125–130).

Schaffer, J. D. (1984). Some experiments in machine learning using vector evaluated genetic algorithms (artificial intelligence, optimization, adaptation, pattern recognition).

Silva, E., Pantoja, M., Celidônio, J., & Klautau, A. (2004). Modelos de linguagem n-grama para reconhecimento de voz com grande vocabulário. In *Iii workshop em tecnologia da informação e da linguagem humana.*

Silva, P., Neto, N., & Klautau, A. (2009). Novos recursos e utilizaçao de adaptaçao de locutor no desenvolvimento de um sistema de reconhecimento de voz para o português brasileiro. *XXVII simpósio Brasileiro de telecomuniçacões*.

Soares, G. L., Guimaraes, F., Maia, C. A., Vasconcelos, J., & Jaulin, L. (2009, May). Interval robust multi-objective evolutionary algorithm. In *Evolutionary computation, 2009. cec '09. ieee congress on* (p. 1637-1643). doi: 10.1109/CEC.2009.4983138

Srinivas, N., & Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, *2*(3), 221–248.

Teruszkin, R., & Vianna, F. (2006). Implementation of a large vocabulary continuous speech recognition system for brazilian portuguese. *Journal of Communication and*

Information Systems, *21*(3), 204–218.

Young, S., Evermann, G., Kershaw, D., Moore, G., Odell, J., Ollason, D., ... Woodland, P. (2009). *The htk book* (Vol. 1). Entropic Cambridge Research Laboratory Cambridge.

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications* (Vol. 63). Shaker Ithaca.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, *8*(2), 173–195.

Zitzler, E., Laumanns, M., & Bleuler, S. (2004). A tutorial on evolutionary multiobjective optimization. In *Metaheuristics for multiobjective optimisation* (pp. 3–37). Springer.