PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
DEPARTAMENTO DE COMPUTAÇÃO
GRADUÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RECONHECIMENTO AUTOMÁTICO DE FALA POR COMPUTADOR

JAILTON ALKIMIN LOUZADA

JUNHO, 2010

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
DEPARTAMENTO DE COMPUTAÇÃO
GRADUÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RECONHECIMENTO AUTOMÁTICO DE FALA POR COMPUTADOR

Trabalho de Conclusão de Curso apresentado por Jailton Alkimin Louzada à Pontifícia Universidade Católica de Goiás, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação aprovado em 22/06/2010 pela Banca Examinadora:

Professor Alexandre Ribeiro, MsC. UCG – Orientador
Professora Solange
Professora Nágela

RECONHECIMENTO AUTOMÁTICO DE FALA POR COMPUTADOR

JAILTON ALKIMIN LOUZADA

Trabalho de Conclusão de Curso apresentado por Jailton Alkimin Louzada à

Pontifícia Universidade Católica de Goiás – Departamento de Computação, como parte dos requisitos para obtenção do título Bacharel em Ciência da Computação.

Professor Alexandre Ribeiro, MsC Orientador	Professor Jeová Martins Ribeiro, MsC Coordenador do TCC

DEDICATÓRIA

Dedico este trabalho, aos meus pais, Maria Helieni da Silva Alkimin e Jailton Neves Louzada pelo amor, carinho, dedicação, compreensão e principalmente por acreditarem em meu sucesso e ajudarem a realizar mais um dos meus sonhos.

Ao meu amigo e professor Alexandre Ribeiro pela amizade, companheirismo e consideração.

EPÍGRAFE

“Investir em conhecimentos rende sempre melhores juros.”

Benjamin Franklin

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me sustentado durante todo esse período.

Agradeço aos meus pais, pelo amor e carinho, paciência e apoio em todos os momentos, principalmente naqueles em que tive que dedicar parte do meu tempo a este projeto.

Agradeço aos meus professores, em especial ao meu orientador Alexandre Ribeiro pela paciência, atenção e dedicação oferecida e por sempre me orientar em decisões na minha vida profissional, acadêmica e pessoal.

À minha amiga Ana Carolina Rézio, agradeço pela amizade, confiança, apoio, incentivo, motivações e pelos bons momentos proporcionados.

RESUMO

Este trabalho tem como objetivo mostrar algumas técnicas de reconhecimento automático de fala por computador. Dentre as técnicas apresentadas foi demonstrada a utilização de Modelos Ocultos de Markov (do inglês *Hidden Markov Models*, HMM) [3,4] para sistemas de reconhecimento automático de fala baseado em análise estatística de padrões. Para a demonstração prática de uma das técnicas estudadas, foi criado um sistema do tipo reconhecedor de fala independente de locutor com vocabulário discreto utilizando o *engine* Julius [26]. Foi necessário além de todos os recursos utilizados, o uso de um modelo acústico [30] e uma gramática [22]. Ao final, a aplicação desenvolvida tinha por fim interagir de acordo com os comandos de fala ditos, sendo que somente reagiria a comandos de fala definidos na gramática e que tinham um grau de confiança (verossimilhança) acima de setenta por cento.

Palavras chave: *Modelos Ocultos de Markov, Reconhecimento Automático de Fala, Julius.*

ABSTRACT

This work aims to show the use of some techniques for automatic speech recognition by computer. Among these techniques was demonstrated using Hidden Markov Models (HMM) [3,4] for automatic speech recognition systems based on statistical analysis of patterns. For a practical demonstration of the techniques studied, was created a system such speech recognizer with vocabulary speaker-independent discrete using the Julius engine [26]. Was necessary in addition to all the resources used, the use of an acoustic model [30] and a grammar [22]. Finally, the developed application was intended to interact in accordance with speech commands said, and only react to speech commands defined in the grammar and had a degree of confidence (likelihood) over seventy percent.

Key words: *Hidden Markov Models, Automatic Speech Recognition, Julius.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABFN	*Augmented Backus–Naur Form*
API	*Application Programming Interface*
ASR	*Automatic Speech Recognition*
CLR	*Common Language Runtime*
COM	*Component Object Model*
CSRC	*Continuous Speech Recognition Consortium*
CUED	*Cambridge University Engineering Department*
dB	Decibel
DLL	*Dynamic-link library*
DNA	Deoxyribonucleic Acid
DP	*Dynamic Programming*
HMM	*Hidden Markov Models*
HTK	*The Hidden Markov Model Toolkit*
Hz	*Hertz*
IDE	*Integrated Development Environment*
IPA	*Information-technology Promotion Agency*
ISTC	*Interactive Speech Technology Consortium*
LAPS	Laboratório de Processamento de Sinais
LVCSR	*Large Vocabulary Continuos Speech Recognition*
MA	Modelo Acústico
MFCC	*Mel-frequency Cepstrum Coefficients*
ML	Modelo de Liguagem
RAF	Reconhecimento Automático de Fala
RCA	*Radio Corporation of America*
SAPI	*Speech Application Programming Interface*
SNR	*Signal Noise Ratio*
SRGS	*Speech Recognition Grammar Specification*
SRI	*Speech Technology and Research Laboratory*
SRILM	*The SRI Language Modeling Toolkit*

W3C	*World Wide Web Consortium*
XML	*Extensible Markup Language*
ZCPAC	*Zero Crossing with Peak Amplitude Cepstrum*

LISTA DE SÍMBOLOS

(*F0*)	Freqüência fundamental / freqüência de *pitch*
$\hat{W}$	Probabilidade Condicional
$\prod$	Produto
~	Valor Aproximado

CAPÍTULO I

INTRODUÇÃO

O problema de reconhecimento de fala vem sido ativamente estudado desde 1950 [1]. No início dos estudos os pesquisadores procuravam explorar idéias básicas da fonética acústica e não tiveram resultados muito satisfatórios. Entre 1950 até 1959, os estudos foram liderados por pesquisadores americanos e ingleses, pesquisadores como Olson e Belar, do *RCA Laboratories* [1] e trabalhos de Fry e Denes da *University College of England* [2].

A partir dos anos 80 o processo de pesquisa no reconhecimento da fala aumentou de forma crucial [1]. Ao contrário do que se via anteriormente, o processo de pesquisa passou a ser realizado a partir de frases fluentes, ao invés de uma palavra isolada. As pesquisas nos anos 80 foram caracterizadas pelas aproximações estatísticas, especialmente pelos Modelos Ocultos de Markov [3] (vide seção 3.1.2.1).

Os sinais da fala são compostos por uma sequência de sons, regulados pelas regras da língua e pelas características do orador. Para entender, sintetizar, reconhecer ou, de um modo geral, processar os sinais de fala, é necessário conhecer as etapas e características de um sistema de reconhecimento de fala.

O reconhecimento da fala por computador consiste em mapear um sinal acústico, capturado por um transdutor (usualmente um microfone ou telefone) em um conjunto de palavras. A finalidade básica de um sistema de reconhecimento de fala, comumente conhecido na literatura como sistemas RAF (Reconhecimento Automático de Fala), é produzir como saída a sequência de palavras ou sentenças correspondentes ao sinal de entrada (transcrição). Os sistemas de reconhecimento de fala podem ser caracterizados e avaliados por vários parâmetros como: vocabulário (*vocabulary*), dependência de locutor (*enrollment*), modo de fala (s*peaking mode*), perplexidade (*perplexity*), condições adversas (SNR) [4] (vide capítulo 2). Existem também três grandes classes de acordo com a técnica utilizada para reconhecimento como: reconhecedores por comparação de padrões, reconhecedores baseados na análise de acústico fonética e reconhecedores empregando inteligência artificial [4].

Dentre as áreas provedoras para esse projeto, destacam-se o processamento digital de sinais e modelagem estatística. Difundido dentre as teorias de processamento e reconhecimento de sinais da fala, foi proposto esse projeto que visa demonstrar através de técnicas de reconhecimento de fala a sua utilização e aplicabilidade no mundo real. Para tal será utilizado a abordagem de reconhecimento de fala por palavras isoladas, com vocabulário discreto e utilizando as técnicas de modelos estatísticos, representado por modelos ocultos de Markov (*Hidden Markov Models*, HMM) [3]. O objetivo é após a transdução da fala como sinal analógico para sinal digital, poder reconhecer o comando falado de tal forma que tenha o maior grau de confiança (verossimilhança) possível, para que seja então realizado com maior exatidão (maior que setenta por cento) os comandos armazenados na gramática do sistema, demonstrando assim a aplicabilidade do reconhecimento de fala.

Durante o processo de reconhecimento da fala há pelo menos dois fatores que interferem diretamente no processo de reconhecimento, como interferência no ambiente, necessitando de um removedor de ruídos para ambientes muito ruidosos, que nesse trabalho contou com a técnica MFCC [30] (vide capítulo 4) para extração de parâmetros da fala.

1.1. CARACTERIZAÇÃO DO PROBLEMA

Para demonstração prática do reconhecimento automático de fala foi construído uma aplicação que propõe demonstrar um ambiente interativo através dos comandos de fala do locutor que esteja usando. A aplicação conta com um ambiente, onde se tem um carrinho que interage em um ambiente, de acordo com os comandos de fala pré-definidos em sua gramática que são: Frente, Trás, Esquerda, Direita e Parar. Além da gramática foi utilizado a *engine* de reconhecimento de fala Julius [26] (vide capítulo 3 e 4) e um modelo acústico criado graças aos esforços do Laboratório de Processamento de Sinais da UFPA [30] (vide capítulo 4).

1.2. ESTRUTURA E CONTEÚDO DO TRABALHO

O presente trabalho está estruturado de forma a ilustrar a teoria envolvida no estudo do tema, reconhecimento automático de fala por computador.

O capítulo 2 traz uma abordagem no contexto natural do que é a fala e como ela é reconhecida, e faz ainda uma analogia do reconhecimento da fala no contexto digital, apresenta algumas representações através de imagens que ilustram de forma mais fácil as etapas do reconhecimento da fala, o que é imprescindível para o entendimento do processo como um todo. O capítulo 3 aborda os recursos normalmente necessários para a construção de sistemas de reconhecimento de fala e ainda cita algumas tecnologias utilizadas para o projeto.

Todas as técnicas utilizadas para a construção da aplicação e ainda alguns detalhes de implementação e resultados obtidos são especificados no capítulo 4. O Capítulo 5 apresenta a conclusão dos estudos e as sugestões para os trabalhos futuros.

CAPÍTULO II

RECONHECIMENTO DE FALA

As seções contidas neste capítulo têm como objetivo apresentar fundamentos sobre o processo de produção e reconhecimento da fala a partir de uma analogia entre a forma natural e a digital. Estes fundamentos servirão como suporte para compreensão dos próximos capítulos.

2.1. FALA

A fala, através de um determinado padrão de uma língua é a forma de comunicação mais utilizada entre os seres humanos [15]. Graças à capacidade do cérebro humano de interpretar informações extremamente complexas, podem-se captar facilmente em uma mensagem falada várias informações tais como: identificar a pessoa que está falando, sua posição no espaço físico, seu estado emocional e outros dados como a ironia, seriedade ou tristeza.

Em contrapartida as máquinas ainda não têm o potencial de reconhecer através da fala informações/dados como nós seres humanos fazemos. A interface Homem *versus* Máquina é restrita a periféricos nem sempre muito fáceis de utilizar e que às vezes demanda uma grande quantidade de tempo para aprender como se utiliza.

2.2. MOTIVAÇÕES PARA O ESTUDO DO RECONHECIMENTO DE FALA

A motivação para se estudar o reconhecimento da fala no meio computacional é grande e representa um grande ganho por vários motivos como [7]:

- Menor curva de aprendizado para utilização de um sistema ou equipamento, não necessitando de determinadas habilidade;

- Rapidez e eficiência na utilização de máquinas/computadores dotados dessa tecnologia;
- Aumento de desempenho individual, pois poderia ao mesmo tempo utilizar as mãos para fazer outras coisas em quanto falasse com o computador ou outro meio dotado da tecnologia;
- Segurança, uma determinada máquina que poderia além de prover o reconhecimento da fala poderia reconhecer a voz de seu usuário.

Apesar das grandes motivações, um sistema de reconhecimento de fala apresenta uma série de cenários que dificultam a sua utilização tais como [7]:

- Interferência do ambiente, necessitando de um removedor de ruídos para ambientes muito ruidosos;
- As características do sinal de fala podem variar de acordo com algumas características como: sexo, sotaque, dimensões dos órgãos do trato vocal, doenças, velocidade da fala e condições físicas.

Para que se possa entender o funcionamento do reconhecimento de fala no meio computacional, deve-se entender o que é a fala propriamente dita, e como ela é gerada pelos seres humanos.

2.3. PRODUÇÃO E RECONHECIMENTO DE FALA HUMANA

Os sinais de fala são compostos por uma sequência de sons, regulados pelas regras da linguagem e pelas características do orador. Para entender, sintetizar, reconhecer ou, de um modo geral, processar os sinais de fala, é necessário conhecer o mecanismo da sua produção [15].

O processo de produção e reconhecimento da fala pode ser descrito de forma mais detalhada através do fluxograma da figura 2.1.

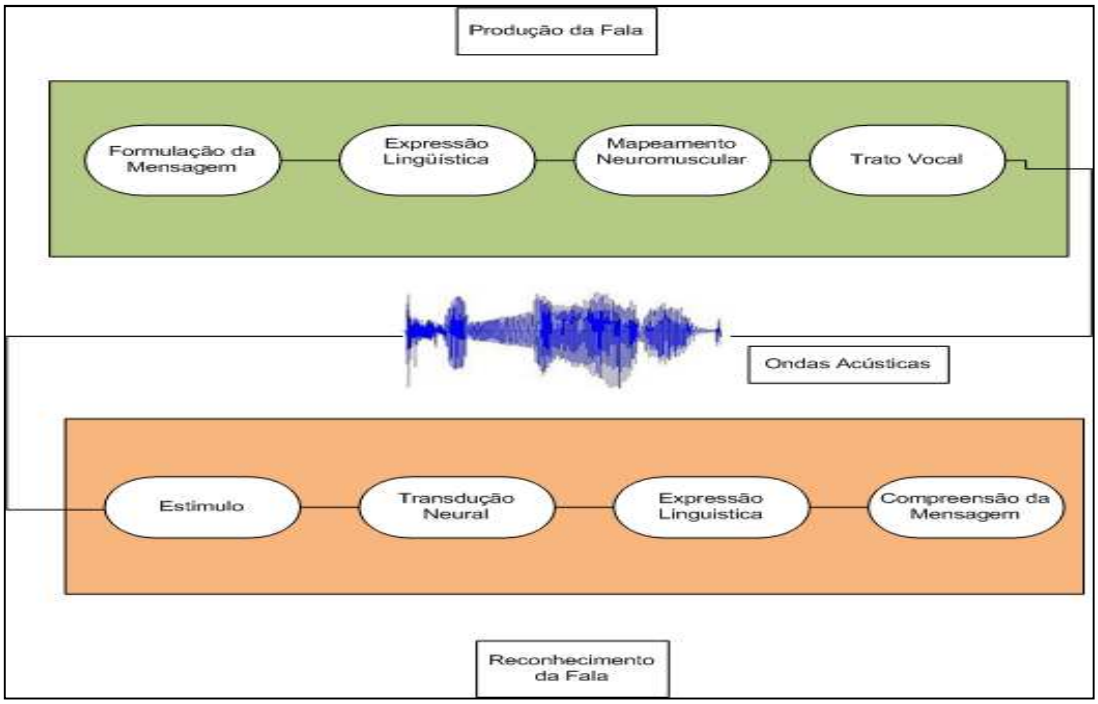


Figura 2.1 *– Processo de Produção e Reconhecimento da fala humana [11]*

O processo de produção inicia-se a partir da formulação da mensagem que consiste na construção da idéia a ser expressa. Em seguida o sistema de expressão linguística é acionado para converter a idéia construída em um conjunto de palavras que serão transmitidas [15]. Com as palavras e fonemas já definidos, o mapeamento neuromuscular inicia para que o trato vocal possa trabalhar corretamente e emitir os sons relativos à mensagem original.

Após a execução da fala pelo locutor, o som produzido propaga-se pelo ar e chega ao ouvinte. Neste momento inicia-se o processo de reconhecimento da fala. O ouvinte recebe um estímulo na membrana no interior do ouvido, a qual executa a análise espectral do sinal e em seguida, através da transdução neural, o sinal espectral que sai da membrana no interior do ouvido é transformado em um sinal elétrico no nervo auditivo. Ao longo do nervo auditivo os sinais são codificados em expressão linguística a partir de algumas informações como o vocabulário e a gramática. Depois de codificada, a mensagem é reconhecida e compreendida pelo ouvinte [15, 6].

2.3.1. TIPOS DE SOM

Existem dois tipos de sons: vozeados (gerados a partir da vibração das cordas vocais) e não vozeados (não faz as cordas vocais vibrarem) [15]. Os sons vozeados, à medida que as cordas vocais vibram, fazem variar o grau de abertura da glote e consequentemente o volume de ar proveniente dos pulmões que passa através dela. Devido às variações periódicas na velocidade de volume na glote que vai excitar o trato vocal, produzindo sons com harmônicas da freqüência de vibração das cordas vocais, ou seja, da freqüência fundamental (*F0*), habitualmente designada por freqüência de *pitch* [5].

A frequência fundamental depende da dimensão e espessura da glote. Para oradores do gênero masculino, a gama de vibração das cordas vocais situa-se entre 50-250 Hz, enquanto que para oradores do gênero feminino essa gama situa-se entre 120-300 Hz.

2.3.2. CARACTERÍSTICAS FISIOLÓGICAS DO SOM

Altura é a qualidade que permite que os sons possam ser classificados em graves (baixa freqüência) e agudos (alta freqüência). Os sons com frequência menor do que 16 Hz são chamados de infra-sons, enquanto os de freqüência maior do que 17.000Hz são denominados ultra-sons [5,15]. Alguns animais são capazes de emitir e de detectar ultra-sons. Entre eles estão muitos insetos. O morcego, por exemplo, ao voar, emite a cada segundo, 10 ou mais pulsos sonoros cuja freqüência localiza-se entre 20.000Hz a 100.000Hz.

A intensidade é a qualidade que permite um som ser percebido a uma maior ou menor distância da fonte sonora. A intensidade de um som é classificada entre forte ou fraca. Como o ouvido humano não tem a mesma sensibilidade para todas as frequências sonoras, mas é mais sensível aos sons cujas freqüências situam-se entre 2.000 e 4.000Hz, a intensidade dos sons ouvidos também varia com a frequência. Além disso, a intensidade dos sons [15]:

- É proporcional ao quadrado da amplitude da onda sonora;
- É mais intensa quanto maior for à superfície de vibração da fonte sonora;
- Aumenta com a densidade do meio em que ele se propaga;

- Diminui com o quadrado da distância entre o observador e fonte sonora, quando o som se propaga em meio homogêneo e infinito;
- Depende da proximidade de ressonadores, pois eles reforçam a intensidade do som;
- É alterada pelos ventos. Estes interferem na intensidade do som quando a distância entre a fonte e o observador é maior do que 6m.

Já o Timbre é a qualidade que diferencia dois sons de mesma altura e de mesma intensidade, mas que são produzidos por fontes sonoras diferentes. O timbre de um som depende do conjunto dos sons secundários (sons harmônicos) que acompanham o som principal.

2.4. RECONHECIMENTO DE FALA POR COMPUTADOR

O reconhecimento automático de fala ou RAF (do inglês, *Automatic Speech Recognition - ASR*) consiste em mapear um sinal acústico, capturado por um transdutor (usualmente um microfone ou telefone) em um conjunto de palavras, ou seja, dada uma entrada em forma de um sinal (onda acústica), produzir uma saída em forma de seqüência de fonemas, palavras ou sentenças correspondentes ao sinal de entrada [11].

A figura 2.2 ilustra um sinal de fala captado por um microfone, o qual passou por um sistema de reconhecimento de fala produzindo como saída o texto “dois”.

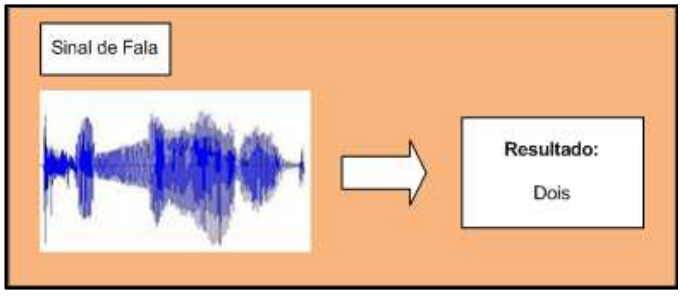


***Figura 2.2** – Processo de reconhecimento de fala utilizando RAF*

De forma simplificada, o RAF consiste no processo de extrair a informação linguística no sinal da fala. Esse processo normalmente acontece em três passos [8]:

1. Aquisição do sinal de fala

2. Extração de Parâmetros

3. Reconhecimento do Padrão

Os reconhecedores de fala podem ser divididos em três grandes classes de acordo com a técnica utilizada para reconhecimento [6]: Reconhecedores por comparação de padrões, reconhecedores baseados na análise acústica fonética e reconhecedores empregando inteligência artificial.

Os sistemas de reconhecimento de fala por comparação de padrões possuem duas etapas: treinamento e reconhecimento. Esse tipo de reconhecedor baseia-se que o sistema já foi calibrado/treinado com informações que poderá reconhecer. Na etapa de treinamento, exemplos dos padrões a serem reconhecidos são apresentados ao sistema, para que seja gerado a partir daí os padrões de referência. Na fase de reconhecimento, compara-se o padrão desconhecido com os padrões de referências no qual é calculada uma medida de similaridade. Finalmente o padrão que melhor corresponder ao padrão desconhecido é usado como padrão reconhecido. Nota-se que quanto maior o número de amostras fornecidas ao sistema na etapa de treinamento maior será o seu desempenho. Sistemas que implementam essa classe de reconhecedor utilizam geralmente a técnica de *Hidden Markov Models* (HMM) (vide capítulo 3) [4] .

Os sistemas que implementam a análise acústico fonética decodificam o sinal de fala baseados nas características acústicas do mesmo e nas relações entre essas características. A idéia é encontrar as unidades fonéticas (fonemas) que compõem a fala a ser reconhecida, e a partir da concatenação dessas unidades reconhecerem a fala [7].

Já a terceira classe, onde estão os sistemas que utilizam inteligência Artificial que exploram conceitos presentes nas suas classes representadas. Redes neurais podem ser enquadradas nesta classe, por exemplo, redes "*Multilayer Perceptrons"* [7].

2.5. CLASSIFICAÇÃO DOS RECONHECEDORES DE FALA

Os sistemas de reconhecimento de fala podem ser caracterizados e avaliados por vários parâmetros, como os exibidos na tabela 2.1 e melhor explicados nas próximas seções:

***Tabela 2.1** – Parâmetros típicos para se caracterizar um sistema RAF [7]*

Parâmetros	**Faixa de Valores**
Vocabulário	Pequeno (< 20 palavras) a Grande (>20000 palavras)
Dependência de Locutor	Independente de locutor a Dependente do locutor
Modo de Fala	Palavras isoladas e Fala continua
Perplexidade	Pequena (<10) a Grande (>100)
Relação sinal-ruído	Alto (>30dB) a Baixo (<10dB)

2.5.1. VOCABULÁRIO (*VOCABULARY*)

Não existe uma definição estabelecida em relação a seu tamanho, mas geralmente é seguida a faixa apresentada na tabela 2.1. Existe também uma definição em quatro tipos distintos de vocabulários que são [9]:

- Vocabulário Pequeno: 1 a 20 palavras;
- Vocabulário Médio: 20 a 100 palavras;
- Vocabulário Grande: 100 a 1000 palavras;
- Vocabulário Muito Grande: mais de 1000 palavras.

2.5.2. GRAU DE DEPENDÊNCIA DO LOCUTOR (*ENROLLMENT*)

O sistema é classificado como dependente do locutor quando o mesmo reconhece apenas a fala dos locutores para os quais foram treinados. Quando o sistema reconhece a fala de qualquer locutor é classificado como independente do locutor.

2.5.3. MODO DE FALA

O sistema é classificado como reconhecedor de palavras isoladas quando cada palavra é falada de forma isolada. Quando o padrão a ser reconhecido é uma sequência de palavras pertencentes a um vocabulário restrito e falado de forma continua o sistema é classificado como reconhecedor de palavras conectadas. O reconhecimento é feito usando padrões de referência para cada palavra. Caso os padrões sejam sentenças ou frases são classificados como reconhecedor de fala contínua.

2.5.4. PERPLEXIDADE (*PERPLEXITY*)

Esta é uma medida muito usada para medir a complexidade da tarefa, a qual constitui a média geométrica do número de palavras que podem seguir uma palavra em uma dada gramática a um vocabulário determinado. Sem a gramática o número de possibilidades pode ser extremamente grande.

2.5.5. RELAÇÃO SINAL-RUÍDO (*SIGNAL NOISE RATIO - SNR*)

Problemas que podem impactar no desempenho do sistema, como: ruídos, ambiente, distorção acústica, diferentes microfones e outros.

2.6. RECONHECEDOR DE FALA POR COMPARAÇÃO DE PADRÕES

Em sistemas de reconhecimento de fala que usa comparação de padrões, as características do sinal de fala desconhecido são comparadas com o padrão previamente armazenado e o padrão mais próximo do sinal de entrada é escolhido. Existem algumas restrições do pós-processador, por exemplo, a gramática que limita o reconhecimento somente para palavras armazenadas nela.

A figura 2.3 ilustra o funcionamento de um sistema de reconhecimento de fala por comparação de padrões. As descrições detalhadas de cada etapa serão apresentadas nas próximas seções.

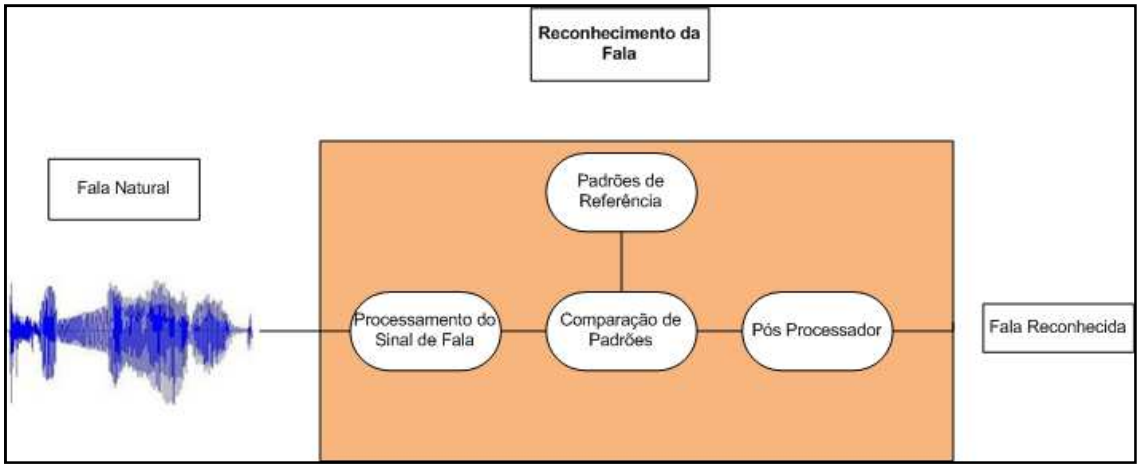


Figura 2.3 *– Sistema RAF baseado em comparação de padrões*

2.6.1. PROCESSAMENTO DO SINAL DA FALA

Nessa etapa as ondas acústicas são digitalizadas, e convertidas em um conjunto de parâmetros para que seja possível comparar os padrões referenciais. No caso de um reconhecedor palavras isoladas é necessário a detecção de pontos limitantes de cada palavra, ou seja, as pausas entre uma palavra e outra, normalmente para resolver esse problema são utilizados algoritmos de início/fim que separam sinal de fala do ruído e pausas entre elas.

2.6.2. PADRÕES DE REFERÊNCIA

Os padrões de referências são gerados na etapa de treinamento do sistema, no qual o usuário calibra o sistema com todas as possíveis palavras que serão reconhecidas, normalmente se usa corpora acústicos, um corpus linguístico é um conjunto de textos escritos ou falados numa língua que serve como base de análise, sendo que corpora é o plural de corpus [31].

É interessante nessa etapa que se tenha uma gama muito grande de padrões cadastrados, para garantir melhor funcionamento e tempo de resposta do reconhecedor. Um exemplo de padrão de referência seria o modelo acústico, no qual faz o uso de modelagem estatística das características dos exemplares cadastrados.

2.6.3. COMPARAÇÃO DE PADRÕES

Consiste em comparar os diversos parâmetros extraídos da fala não reconhecida com os parâmetros armazenados como padrões de referência. Quando se usa HMM (vide capítulo 3) [4], o resultado das comparações é probabilidade de que cada modelo de referência tenha gerado um conjunto de padrões de entrada.

2.6.4. PÓS-PROCESSADOR

Nessa fase são verificadas as distâncias das probabilidades obtidas, a fim de se encontrar os melhores padrões de referência candidatos a serem escolhidos, geralmente são utilizadas regras de sintática, semântica ou gramática. É interessante, pois descarta alternativa não muito boa para serem candidatos.

2.7. DIFICULDADES NA CONSTRUÇÃO DE SISTEMAS RAF

Apesar das tentativas de se chegar à perfeição quanto ao reconhecimento de fala por computador e dos grandes avanços da tecnologia de processamento de fala, o entendimento da fala por completo na sua forma natural ainda é uma tarefa difícil e complexa de se realizar. São vários os fatores que interferem na precisão do reconhecedor, ao passar dos anos alguns vem sendo eliminados e aumentando se cada vez mais os estudos e pesquisas para chegar à perfeição do reconhecimento da fala humana. Alguns dos problemas enfrentados, inclusive nesse trabalho são mostrados abaixo:

- Devido a uma determinada região, uma mesma palavra pode ser dita de diversas formas de acordo com o sotaque característico do locutor;
- Pronúncia das palavras, pois varia de locutor para locutor, podendo às vezes falar mais rápido ou mais devagar, nesse caso algumas silabas ou letras podem ser “engolidas”;
- Ambiguidade entre palavras, pois palavras homófonas, como “consertar/concertar” tem a mesma pronuncia, o que se torna quase impossível para o reconhecedor identificar somente com as informações acústicas qual é a palavra correspondente;

- Segmentação das palavras. Nesse caso o reconhecedor pode ter problemas em decodificar as palavras de forma isolada por não conseguir encontrar a fronteira de uma palavra com outra, pois na fala natural possa haver pequenas pausas entre as palavras;

- Baixa relação sinal-ruído (SNR). Durante o reconhecimento quanto mais limpo (sem vozes de outros locutores ou ruído de outros elementos) for o ambiente acusticamente falando, melhor para o reconhecedor decodificar e diferenciar as palavras.

Contudo, a proposta de um ambiente mais interativo entre o homem e a máquina ainda é algo fascinante, mesmo contando com alguns problemas como os que foram apresentados acima, com o passar dos tempos e avanços na tecnologia um dia haverá a comunicação natural entre homem e máquina de forma ótima, ultrapassando todos os empecilhos.

CAPÍTULO III

RECURSOS PARA UM SISTEMA DE RECONHECIMENTO DE FALA

Em um sistema de reconhecimento de fala são usados vários recursos, entender o funcionamento e aplicabilidade desses recursos será de suma importância para a construção de sistemas de reconhecimento de fala. Nesse capítulo serão mostrados alguns destes recursos, que inclusive servirão de suporte para a construção da aplicação proposta nesse trabalho.

3.1. DIVISÕES BÁSICAS DE UM SISTEMA RAF

A maioria dos sistemas de reconhecimento de fala atualmente são baseados em reconhecimento estatístico de padrões [10,11], como apresentado no capítulo 2. O sistema busca por sequências prováveis que melhor representam o sinal de entrada nos os modelos que foram treinados anteriormente na calibragem do sistema. Para realização da busca faz-se necessário um conhecimento prévio dos modelos acústicos e modelo de linguagem utilizado [4]. Faz-se o uso da regra de *Bayes*.

$$\widehat{W} = \arg\max_W \frac{P(O|W)P(W)}{P(O)} = \arg\max_W P(O|W)P(W) \quad 3.1$$

Com base na regra de *Bayes* visto na equação 3.1, pode-se analisar que a busca é realizada pela sequência de palavras *W* que maximiza a probabilidade condicional $\widehat{W}$. Sendo *O* a matriz de parâmetros que representa o sinal acústico, tem-se *P(O|W)* como sendo a probabilidade de se observar a matriz *O* dada a sequência de palavras *W*, valor fornecido pelo modelo acústico, e *P(W)* como sendo a probabilidade da sequência de palavras *W* adquirido pelo modelo de linguagem ou gramática. Como *P(O)* não depende de *W* o mesmo pode ser descartado.

Um sistema RAF é composto por vários módulos que possuem funções distintas conforme ilustrado na figura 3.1. Os módulos atuam de forma interligada, cada um com sua responsabilidade e provendo informações processadas para os outros. Estes módulos consistem de *front-end*, a parte do programa que é responsável pela interface do usuário e responsável pela extração de parâmetros (*features*) do sinal da fala, o Modelo Acústico (MA) que busca modelar a partir das *features* o sinal acústico e armazena toda a fala utilizada para treinamento do sistema.

O Modelo de Linguagem (ML), é algumas vezes conhecido como gramática dependendo do contexto onde é empregado, normalmente se usa o termo gramática quando se trata de um modelo de linguagem para reconhecedores de fala discreto. O ML tenta a partir de textos da língua, obter as possíveis sequências de palavras a serem reconhecidas e o reconhecedor que realiza efetivamente o reconhecimento do sinal de fala. O Reconhecedor é a parte final, onde através de todas as informações anteriormente processadas são então realizadas as comparações entre os padrões de referência e a palavra então é reconhecida.

Nas próximas seções será detalhado o funcionamento de cada um dos módulos de um sistema de reconhecimento de fala:

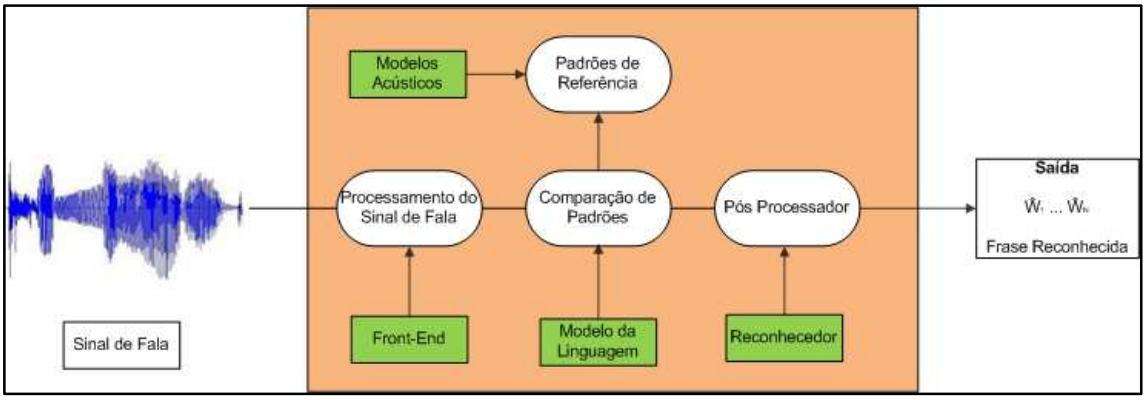


Figura 3.1 *– Módulos de um sistema RAF X Sistema baseado em comparação de Padrões*

3.1.1. FRONT-END

O início do processamento da fala é a conversão analógico-digital que decodifica a onda analógica em dados digitais. Durante a conversão o sistema faz o uso de filtros, para

filtrar o som digitalizado com a intenção de remover ruídos indesejáveis, na parametrização do sinal da fala (processo de filtragem).

Embora não seja foco desse trabalho demonstrar a utilização da melhor técnica para extração de características do sinal da fala em termos de desempenho do sistema em ambientes ruidosos, vale ressaltar a duas técnicas de extração, denominadas MFCC (*Mel-frequency Cepstral Coefficients*) [30] e ZCPAC (*Zero Crossing with Peak Amplitude Cepstrum)* [30].

A Técnica ZCPA, tem seu desempenho melhor do que a técnica MFCC em ambientes ruidosos. Porém, para ambientes não ruidosos, o MFCC proporciona um bom desempenho nos sistemas de reconhecimento de locutor alcançando índices de até 100% no reconhecimento para ambientes com uma Relação Sinal Ruído (SNR) de 0dB (sinal limpo) [17].

No presente trabalho foi adotada a técnica MFCC, proposta por Davis e Merlmestein, em 1980 [16]. Esta técnica foi adotada por apresentar resultados desejáveis no âmbito desse projeto e por já possuir integração nativa com a tecnologia proposta. Apesar da robustez do sistema de reconhecimento de fala ser obtida em diferentes estágios do processamento, a escolha da melhor técnica para a extração de características dos sinais da fala é de grande importância se tratando do desempenho do sistema. Uma análise detalhada, principalmente do ambiente é de suma importância para uma escolha adequada.

3.1.2. MODELO ACÚSTICO

O modelo acústico gera um modelo matemático, a partir das características (*features*) extraídas da fala, que represente um sinal original. Isso ocorre de tal forma que os seguimentos da fala que são desconhecidos possam ser mapeados de modo correto e que seja determinada a palavra correspondente. Em caso de sistemas de reconhecimento de fala contínua o modelo matemático é determinado por fonemas e não por palavras isoladas.

3.1.2.1. MODELOS OCULTOS DE MARKOV

Os estudos de processamento de voz e posteriormente no campo de reconhecimento estatístico da fala são apontados pelos trabalhos independentes de Baker juntamente com a *Carnegie Mellon University* [12], Jelinek juntamente com a IBM [4,13].

Os Modelos Ocultos de Markov (em inglês, *Hidden Markov Models*, HMM) podem ser classificados em modelos discretos e contínuos [20]. Em sistemas RAF discretos, o reconhecimento do sinal de fala é feito baseado nas palavras mantidas na gramática. Para sistemas RAF contínuos [11], geralmente as palavras do modelo de linguagem são representadas através de um conjunto de modelos probabilísticos de unidades linguísticas elementares (por exemplo, fonemas) [4]. Qualquer sequência de parâmetros acústicos extraídos de uma locução é definida pela concatenação de processos elementares descritos por HMM. Um HMM é composto de dois processos estocásticos, uma cadeia de Markov oculta, relacionada à variação temporal, e um processo observável, relacionado à variabilidade espectral [4].

Nos modelos discretos, a distribuições são definidas em espaços finitos. Neste caso, as observações são vetores de símbolos de um alfabeto finito de N elementos distintos [4]. Nos modelos contínuos são definidas distribuições como densidades de probabilidade em espaços de observação contínuo. Nesse caso, devem ser impostas fortes restrições à forma funcional das distribuições, de modo a se obter um numero razoável de parâmetros a serem estimados [4].

Um HMM é definido como um par de processos estocásticos (X, Y). O processo X é uma cadeia de Markov de primeira ordem e não é diretamente observável, enquanto que o processo Y é uma sequência de variáveis aleatórias que assumem valores no espaço de parâmetros acústicos (observações) [20]. Um HMM gera sequências de observações pulando de um estado para outro, emitindo uma observação a cada salto.

Em geral, para o reconhecimento de fala, é utilizado um modelo simplificado de HMM conhecido como modelo *left-right* ou modelo de Bakis [4, 20]. Neste modelo, a sequência de estados associada ao modelo tem a característica de, à medida que o tempo aumenta, o índice do estado aumenta ou permanece o mesmo. O sistema trabalha da esquerda para a direita no modelo, como pode ser visto na figura 3.2.

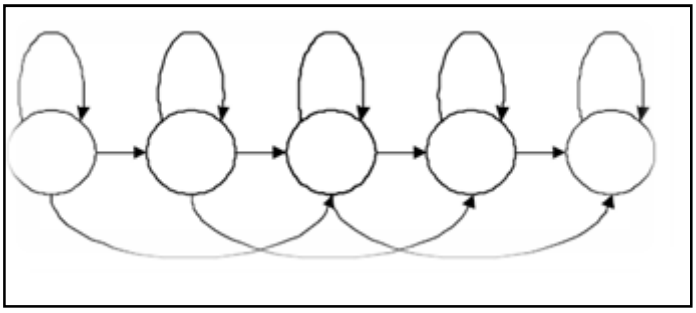


Figura 3.2 *– Estrutura de um HMM left-right de cinco estados [11]*

A observação de saída é a ocorrência do fenômeno sendo modelado. Em sistemas RAF com vocabulário discreto são as palavras contidas na gramática. A figura 3.3 ilustra o modelo básico de um HMM com 3 estados emissores (estados 2, 3 e 4) e 2 estados que não emitem observações.

Estados não emissores são utilizados para concatenação de modelos. A evolução da cadeia de estados, para o caso de reconhecimento da fala, é da esquerda para direita (*left-right*). Sendo *aij* a probabilidade de transição do estado *i* para estado *j* que ocorre a cada tempo *t*, onde nos casos em que *i=j* um estado sofre transição para ele mesmo, e *bi*(*xt*) é a probabilidade da observação *x* no tempo *t* dado que o estado é *i*. Durante o processo não se tem informação sobre a evolução da cadeia de estados (*hidden*).

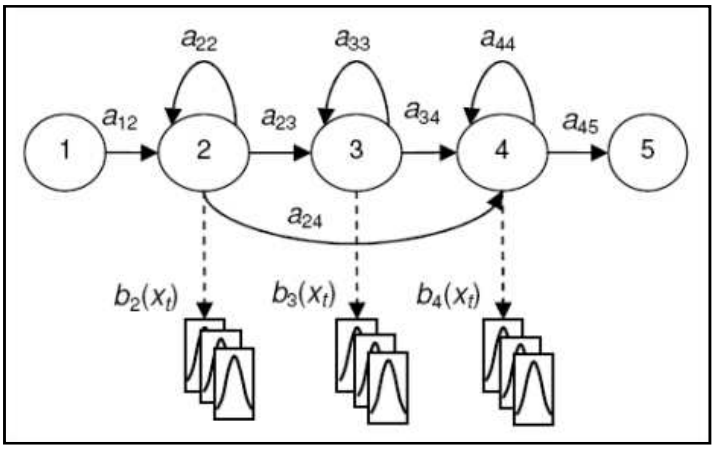


Figura 3.3 *– Modelo HMM [11]*

3.1.2.2. HTK- *THE HIDDEN MARKOV MODEL TOOLKIT*

The Hidden Markov Model Toolkit (HTK) é uma ferramenta portátil para a construção e manipulação de HMM, originalmente desenvolvido no *Speech Vision and Robotics Group* no Departamento de Engenharia da Universidade de Cambridge (CUED) [18]. Esta ferramenta é utilizada principalmente em pesquisas na área de reconhecimento de fala, embora tenha sido utilizado para várias outras aplicações, incluindo pesquisas no que diz respeito à síntese de voz, reconhecimento de caracteres e sequenciamento de DNA.

O HTK consiste de uma série de módulos de biblioteca e ferramentas disponíveis em linguagem C. As ferramentas provem sofisticadas facilidades para análise da fala, treinamento de HMM, teste e análise de resultados. O software suporta HMMs usando tanto modelos de fala discreta como contínua e pode ser usado para construir complexos sistemas HMM [18].

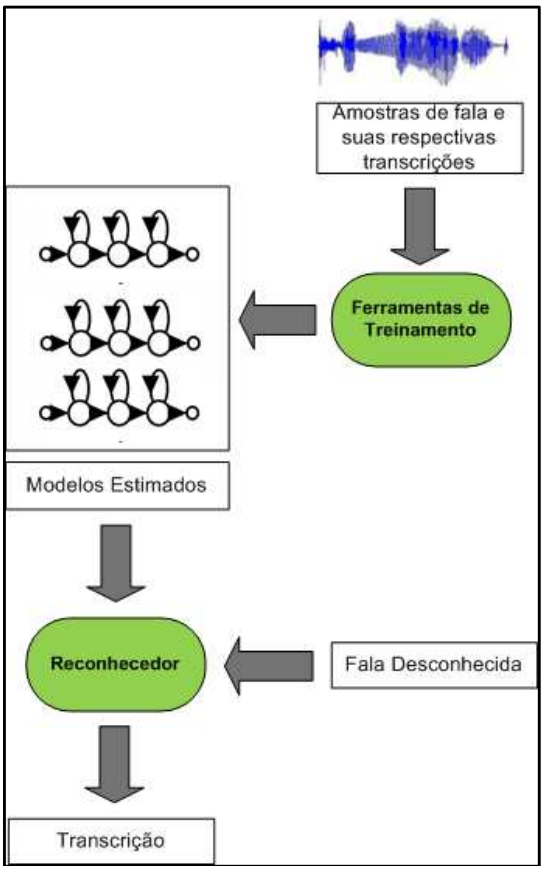


Figura 3.4 *– Principais estágios na criação de sistemas RAF com o HTK*

FONTE: HTK Book

Os principais estágios para a criação de sistemas RAF utilizando HTK são apresentados na figura 3.4 [19]. Na primeira fase as amostras da fala com suas respectivas transcrições são utilizadas pelo *toolkit* do HTK para estimar os parâmetros das HMMs. No estágio seguinte as declarações são transcritas através das ferramentas do reconhecedor do HTK.

3.1.2.3. ATK - *REAL-TIME API FOR HTK*

A *Real-Time API for HTK* (ATK) é uma Interface de Programação de Aplicativos (API, do inglês *Application Programming Interface*) desenvolvida para facilitar a construção de aplicações experimentais para HTK. O mesmo consiste de uma coleção de bibliotecas em C++ que fazem uso das bibliotecas do HTK. Isso permite que modernos reconhecedores sejam construídos usando versões customizadas do HTK para serem compiladas com ATK e testadas. ATK é multiplataforma sendo muito usada em Unix e Windows [18].

3.1.3. MODELO DE LINGUAGEM

Principalmente em sistemas de reconhecimento contínuo para grandes vocabulários, comumente conhecido na literatura como LVCSR (*Large Vocabulary Continuos Speech Recognition*), que possuem uma extensa gramática, utilizar informações acústicas não é a maneira mais fácil de obter um bom desempenho do sistema, pois as palavras não necessariamente são ditas de forma isolada. Uma palavra poderia ser seguida por qualquer outra o que dificultaria o trabalho do reconhecedor.

Estimando-se um vocabulário de tamanho *V*, no reconhecimento de *N* palavras existem V^N possibilidades, resolve-se parte desse problema utilizando Modelos de Linguagem (ML).

O ML tem como objetivo estimar de forma o mais confiável possível a probabilidade de ocorrência de uma determinada sequência de palavras $\mathrm{W} = \mathrm{w}_1, \mathrm{w}_1, \ldots \mathrm{w}_\mathrm{k}$, onde k é o numero de palavras na sentença [11]. A probabilidade $P(W_1^k)$ é definida na equação 3.2.

$$P\left(W_1^k\right) = P(w_1)\mathrm{P}(w_2|w_1)\mathrm{P}(w_3|w_1,w_2)\ldots P(w_k\,|w_1,\ \ldots,\ w_k\ \text{-}\ 1) \qquad (3.2)$$

3.1.3.1. MODELOS DE LINGUAGEM N-GRAMA

Um *n*-grama é uma sequência de *n* símbolos (por exemplo: palavras). Um modelo de linguagem *n*-grama é usado para predizer cada símbolo na sequência dado os *n*-1 predecessores [19]. Isto é feito na suposição de que a probabilidade de um especifico *n*-grama ocorre em algum texto teste desconhecido que pode ser estimado da frequência dessa ocorrência em algum texto dado para treino.

Desse modo, como ilustrado na figura 3.4, a construção de um *n*-grama é um processo de três estágios, primeiramente o texto treinado é escaneado e os *n*-gramas são contados e armazenados em uma base de dados de símbolos *n*-gramas (*gram files*) [19]. No segundo estágio algumas palavras podem ser mapeadas para fora de uma classe do vocabulário ou outra classe de mapeamento pode ser aplicada. No final do estágio o resultado da contagem dos símbolos *n*-gramas são usados para computar as probabilidades dos *n*-gramas que podem ser armazenadas em um arquivo definido como um modelo de linguagem.

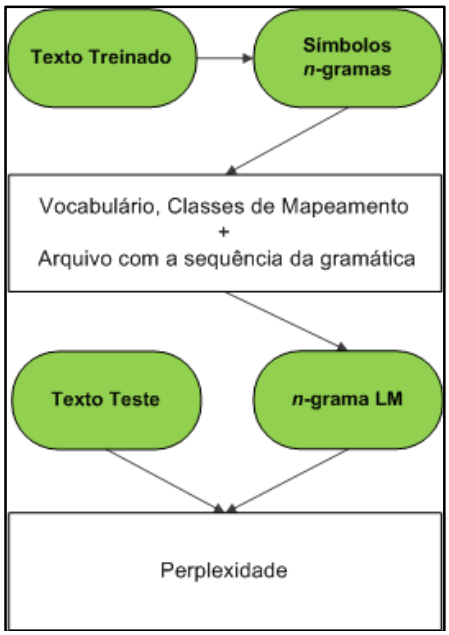


Figura 3.5 *– Principais estágios na Construção de n-gramas*

FONTE: HTK Book

Finalmente um bom modelo de linguagem pode ser estimado para uso no cálculo de uma medida chamada perplexidade (*perplexity*) no conjunto de texto não observado [19].

3.1.3.2. PERPLEXIDADE DE UM MODELO DE LINGUAGEM

Na criação de modelos de linguagem são várias as características que os diferenciam, tais como: vocabulário, definido pela quantidade de palavras existentes no modelo de linguagem e tipo de corpus utilizado para treino. Dependendo da aplicabilidade do sistema RAF as frases utilizadas na estimação podem pertencer a domínios específicos, tais como: conversação, jornalismo, jurídico, radiologia, etc. Como o modelo de linguagem busca facilitar o trabalho do decodificador, restringindo o número de possíveis palavras que podem seguir uma dada palavra, o mesmo influencia diretamente no seu desempenho. Nesse contexto é abordada a perplexidade em sistemas RAF, no qual a perplexidade do conjunto de frases não vistas pelo modelo na fase de treino (conjunto de teste) [11] avalia a capacidade de generalização do ML. Nesse cenário percebe-se que quanto menor a perplexidade melhor tende a ser o *performance* do reconhecedor.

3.1.3.3. CRIAÇÃO DE MODELOS DE LINGUAGEM USANDO O SRILM

Um modelo de linguagem (ML) representa o domínio de todas as palavras que o reconhecedor pode efetivamente “entender” no processo de reconhecimento representando um grande ganho como visto nas seções anteriores. Na construção de modelos de linguagem para o PT-Br vale ressaltar os esforços do grupo Fala Brasil [30] que construíram um modelo de linguagem *n*-grama com o toolkit SRILM (*The SRI Language Modeling Toolkit*) [32] para utilização principalmente em sistemas de reconhecimento do tipo LVCSR (reconhecimento contínuo para grandes vocabulários). Para treino foram utilizadas frases dos corpora CETENFolha, Spoltech, OGI-22, Westpoint, LapsStory e LapsNews, totalizando 1,6 milhões de frases [30].

O SRILM é um toolkit para construção e manipulação de modelos estatísticos de linguagens, criado para aplicações em reconhecimento de voz e processamento de linguagem natural [32]. Criado pelo SRI (*Speech Technology and Research Laboratory*) o SRILM conta

com um grande número de ferramentas e possui código aberto, além de ser multiplataforma e já possuir uma boa documentação [32]. O modelo de linguagem em PT-Br criado, foi construído através dos corpora citado e foram definidos no formato ARPA (ARPA – *text format*), no qual o *engine* Julius possui suporte para interpretação.

```
1
2  \data\
3  ngram 1=64966
4  ngram 2=3515726
5  ngram 3=2359696
6
7  \1-grams:
8  -1.216156   </s>
9  -99 <s> -0.9400495
10 -1.489458   a   -1.027403
11 -6.733438   aas -0.2643106
12 -6.07068    aba -0.2308172
```

Figura 3.6 *– Trecho do Modelo de Linguagem*

Na figura 3.6 é exibido um trecho do modelo de linguagem utilizado para esse projeto no formato ARPA - *text format*, basicamente o modelo ARPA- *text format*, possui um cabeçalho com algumas informações do modelo de linguagem, iniciado pela *tag* **data**\ que define o início das informações do ML, em seguida, são definidos a quantidade de *n*-gramas para cada tamanho de *n*-gramas contidos no ML, nesse caso definindo para ***ngram 1, 2** e **3***. Logo abaixo, são definidos os *n*-gramas propriamente ditos, sendo que são definidos em suas *tags* de início respectivamente, nesse exemplo *n*-gramas de tamanho 1 começando a partir da *tag* **1-grams:**, ao final o término do ML é definido pela *tag* **end**\.

3.1.3.4. TIPO DE GRAMÁTICA

Uma gramática segue os mesmos princípios de um modelo de linguagem, a diferença é que na literatura costuma-se chamar de gramática o modelo de linguagem que é usado para sistemas RAF de vocabulário discreto. O termo modelo de linguagem é mais utilizado para reconhecedores de fala contínua.

São várias as formas de se definir uma gramática, no entanto a mais comum é a baseada na Especificação de Gramática para Reconhecedores de Fala (SRGS, do inglês *Speech Recognition Grammar Specification*) [22] que é um padrão W3C (*World Wide Web Consortium*) de como definir gramáticas para sistemas reconhecedores de fala.

SRGS especifica duas formas alternativas, mas equivalentes de sintaxe, uma baseada em XML (*Extensible Markup Language*) e outra usando ABFN (*Augmented Backus–Naur Form*) [22,23]. Na prática XML é mais frequentemente usada, pela facilidade de utilização e suporte ao desenvolvimento da gramática.

Dentre as vantagens de se utilizar uma gramática definida em SRGS-XML destacam-se [24]:

- Melhora a precisão do reconhecimento, limitando e indicando a uma *engine* (vide seção 3.1.4.1) que palavras ela deve esperar;
- Facilita a manutenção das gramáticas textuais, através de construções de componentes reutilizáveis de texto, listas de frases e identificadores numéricos;
- Melhora a tradução da fala reconhecida em ações da aplicação. Isto é facilitado pela existência de *tags* semânticas que definem o nome da propriedade e as associações de valor para palavras/frases declaradas dentro da gramática.

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <GRAMMAR>
3    <RULE>
4      <LIST>
5        <P>UM</P>
6        <P>DOIS</P>
7        <P>TRES</P>
8        </LIST>
9    </RULE>
10 </GRAMMAR>
```

Figura 3.7 *– Exemplo de uma gramática no padrão SRGS-XML*

Todo elemento XML consiste de *tags* de início (<INÍCIO>) e *tags* de fim (</FIM>), no qual o nome da *tag* é *case-insensitive* (não faz a diferença entre letras maiúsculas e

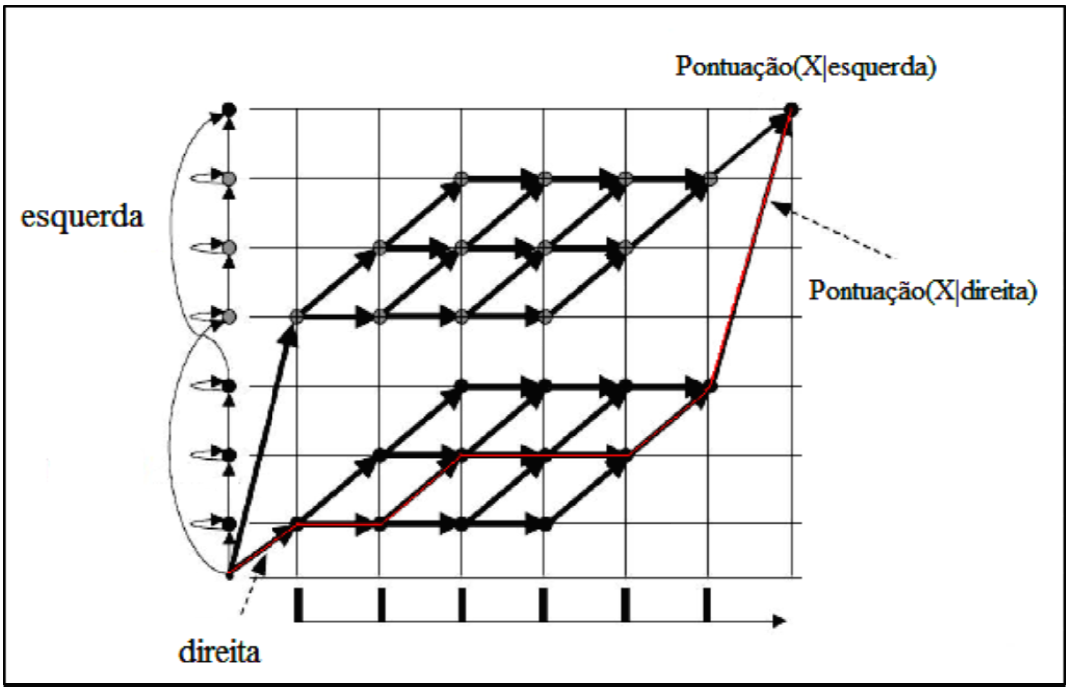
minúsculas) e o seu conteúdo é definido entre as *tags* respectivas [24]. Na figura 3.7 é mostrado um simples exemplo de uma gramática definida no padrão SRGS-XML.

3.1.4. RECONHECEDOR

O reconhecedor que é um dos módulos finais de um sistema RAF tem por finalidade a transcrição de amostras de fala desconhecidas para sua forma textual. Os itens anteriores mostram os modelos matemáticos para construção de modelos acústicos e de linguagem construídos pelos módulos anteriores, que são utilizados pelo reconhecedor nesse processo de transcrição.

O processo de reconhecimento é controlado por uma rede de palavras construída a partir do modelo de linguagem. A rede consiste em um conjunto de palavras (nós) conectadas por arcos, onde cada arco possui uma probabilidade de ocorrência (transição). Com base no dicionário fonético e um conjunto de modelos HMMs estimados, consegue-se determinar, dada uma fala desconhecida, a probabilidade de qualquer um dos caminhos que a rede pode percorrer. O reconhecedor tem como objetivo encontrar os caminhos que são mais prováveis. A busca pela procura do caminho mais provável é realizada através de programação dinâmica (*DP – Dynamic Programming* [21]).

A Figura 3.8 mostra um exemplo de reconhecimento de palavras isoladas, onde se tem um HMM com três estados para a palavra “direita” e outra para palavra “esquerda”. No eixo *X* têm-se os *frames* vindos do *front-end* após a realização da extração de parâmetros (*features*) da fala, cada um dos possíveis caminhos possui uma probabilidade de ocorrência. Após determinada a máxima probabilidade para cada HMM das palavras “esquerda” e “direita”, aquela que possuir maior pontuação será escolhida pelo reconhecedor.



ras isoladas

3.1.4.1. *ENGINE*

Um *engine* voltado para o reconhecimento da fala é um programa de computador e/ou conjunto de bibliotecas, para simplificar e abstrair o desenvolvimento de aplicações voltadas para essa área. Existem três tipos de *engines* [25]: dependentes de locutor, independentes de locutor e adaptativas.

Os *engines* desenvolvidos com dependência de locutor necessitam de um longo e cansativo treinamento adicional. Além do sistema se adaptar às características acústicas do usuário, o locutor também acaba ajustando seu estilo de voz para conversar com o computador. Esses softwares não demandam muita capacidade da máquina, proporcionam bom desempenho e são mais empregados em aplicações de uso individual, ou para pessoas com deficiência na fala [25].

Nos *engines* com independência de locutor, os treinamentos extras são dispensáveis, pois os mesmos são implementados para serem utilizados por qualquer pessoa, não interessando a sua origem, sendo ideal para ambientes públicos [25].

Por último, existem os *engines* adaptativos, que tampouco necessitam de treinamento, se aprimorando, contudo, com o uso. Com isso, podem ser falhos quando utilizados por um grupo muito variado. Tanto os *engines* independentes, como os adaptativos, precisam dispor de uma elevada capacidade computacional e apresentam resposta razoável [25].

Para o presente trabalho foi utilizado o *engine* Julius [26]. O Julius é um reconhecedor *open-source* de alta *performance* para grandes vocabulários e independente de locutor que permite uma grande gama de opções além de suportar modelos (acústicos e de linguagem) criados com a ferramenta HTK. Sua licença permite o seu uso comercial.

O Julius vem sido desenvolvido e aprimorado por pesquisadores japoneses desde 1997 [26], e o trabalho foi continuado pela IPA (*Information-technology Promotion Agency*, *Japonese dictation toolkit project* (1997-2000), *Continuous Speech Recognition Consortium*, *Japan* (CSRC) (2000-2003) e atualmente pelo *Interactive Speech Technology Consortium* (ISTC) [26].

Dentre as várias características que tornam o Julius um ótimo *engine* para a criação de sistemas RAF destacam-se [26]:

- Julius é um *software* de alta *performance* para sistemas LVCSR baseado em modelos de linguagem *n*-gramas. Ele é capaz de executar o reconhecimento de frases no nível de um vocabulário de dezenas de milhares palavras;

- Realiza reconhecimento de fala em alta velocidade em PC’s típicos, executa em tempo real e tem uma taxa de reconhecimento superior a 90% de um ditado em um vocabulário de aproximadamente 20 mil palavras;

- A melhor característica do Julius é o multiuso, pois recombinando o modelo de linguagem e os modelos acústicos é possível construir vários sistemas RAF tanto baseado em fala contínua como por palavras isoladas, com tarefas específicas. Além disso, seu código fonte é *open-source* o que permite a sua flexibilidade na utilização e inclusive o uso comercial;

- As plataformas atualmente suportadas incluem: Linux, Solaris e versões do Unix e Windows. Na versão Windows o Julius é compatível com a tecnologia Microsoft

SAPI 5.0 [24] e ainda dispõe da versão DLL (*Dynamic-link library*, biblioteca de ligação dinâmica).

Na figura 3.9 é mostrada uma visão geral da organização e código fonte do *engine* Julius, separando o núcleo do sistema com as bibliotecas utilizadas.

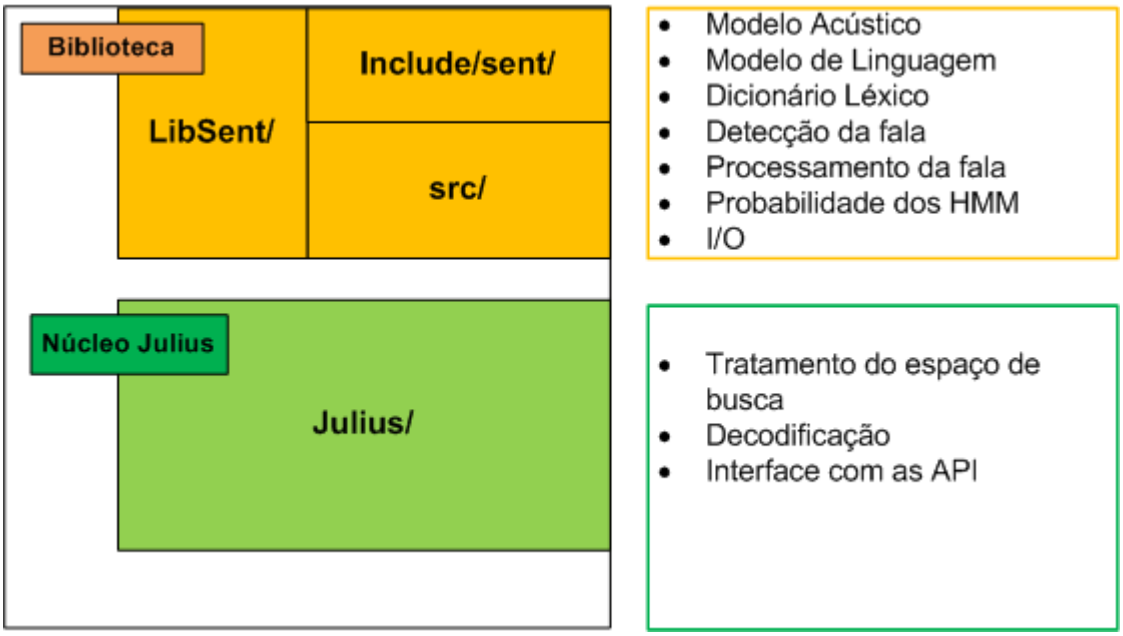


Figura 3.9 *– Visão geral do código fonte e organização do Julius*

CAPÍTULO IV

APLICAÇÃO DESENVOLVIDA

Para alcançar o objetivo desse trabalho foram utilizados vários recursos que facilitam o desenvolvimento de aplicações que utilizam as tecnologias empregadas no reconhecimento de fala. Muitos esforços voltados para essa área já foram realizados e vêm cada vez mais ganhando forças. Dentre esses esforços destacam-se os do grupo Fala Brasil, do Laboratório de Processamento de Sinais (LAPS) da Universidade Federal do Pará [30].

4.1. INTERAÇÃO DO USUÁRIO COM O SISTEMA

Para o proposto trabalho que consiste em demonstrar uma aplicabilidade prática de sistemas RAF utilizando modelos estatísticos, foi elaborado um único módulo para representar a interação do usuário com o sistema a fim de prover uma interface amigável (vide figura 4.1). A interação do usuário com o sistema ocorre via fala, ao iniciar o sistema, será habilitada a interface de voz do usuário entre o sistema que permitirá ao usuário interagir com os respectivos comandos pré-definidos na gramática (vide seção 4.5). Os outros módulos do sistema, que foram construídos a partir de outras ferramentas são mostrados em forma de pacote.

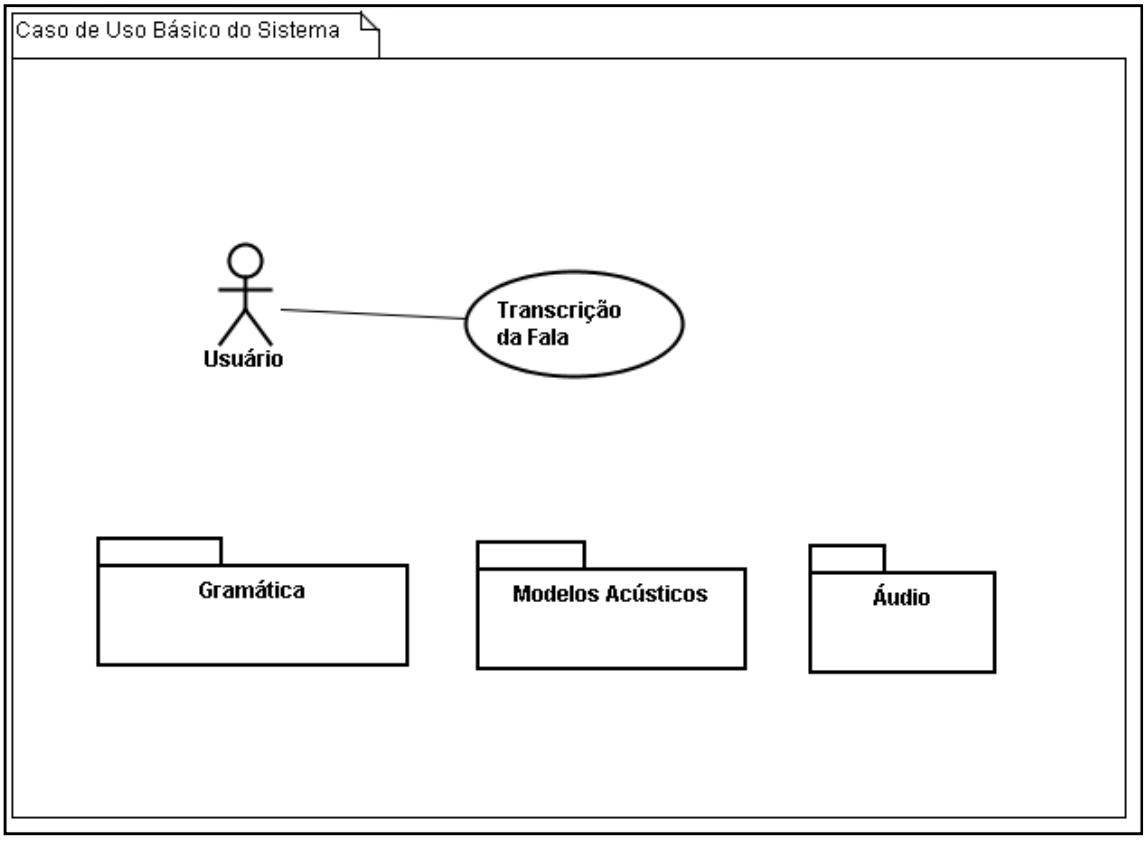


Figura 4.1 *– Diagrama de Caso de Uso do Sistema*

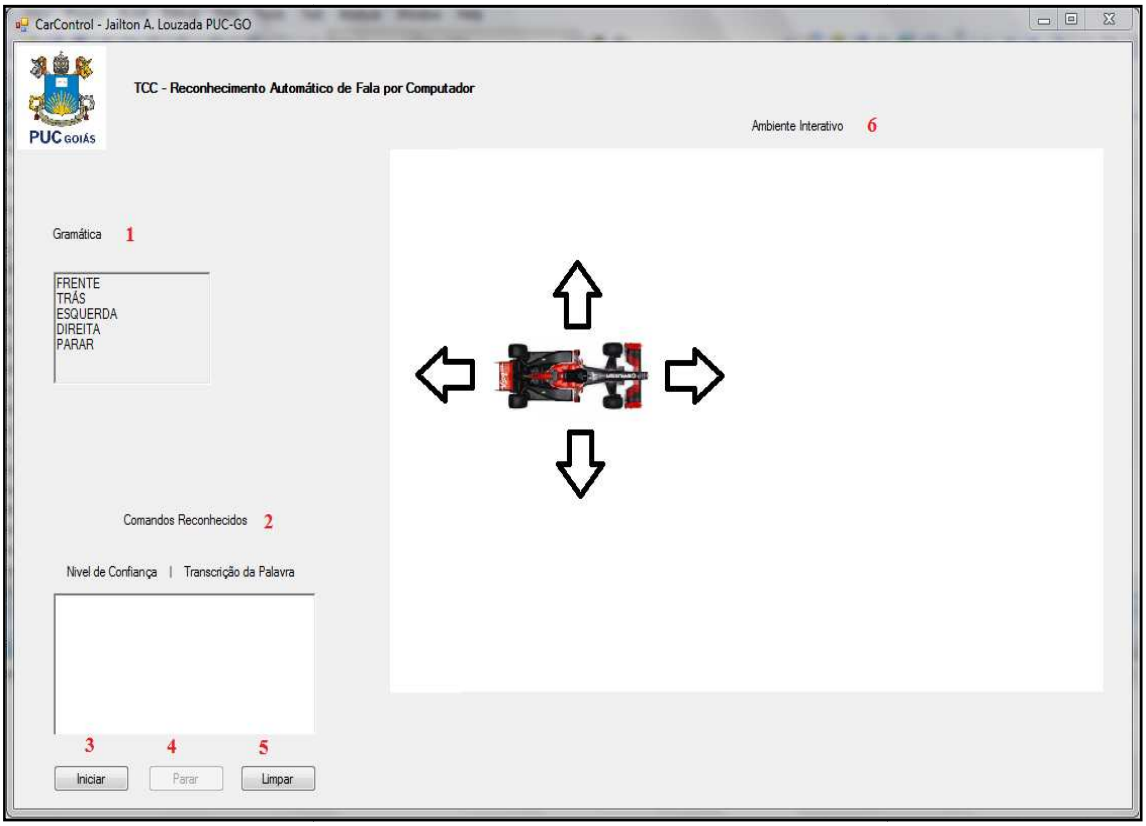
4.1.2. PROTÓTIPO DE TELA DO SISTEMA

A proposta de implementação para esse trabalho é uma pequena aplicação com uma interface amigável como pode ser vista na figura 4.2.

No item 1 da figura 4.2 é vista as palavras da gramática, no item 2 é onde serão exibidos todos os *logs* das ações realizadas pelo sistema, ou seja, para todo comando reconhecido será exibido o seu nível de confiança e a respectiva palavra que foi dita, por exemplo: “0.9422246 | PARAR”.

Quando uma palavra não for reconhecida o sistema exibe a frase: “Não reconhecido”. O item 3 é o botão “Iniciar” onde habilita o sistema para a captura da fala e realiza o reconhecimento efetivamente, em seguida no item 4 é o botão que faz com que a aplicação pare o reconhecimento e o carrinho, nesse caso o botão “Parar”só é habilitado quando o botão “Iniciar” estiver habilitado, para excluir os *logs* das ações do sistema basta clicar no botão “Limpar” conforme definido no item 5. Por final as ações faladas após reconhecidas e

transcritas são “convertidas” em ações em um carrinho que interage no ambiente conforme a ação dita respectivamente, como mostrado no item 6.



4.2. ETAPAS SEQUENCIAIS DO SISTEMA

Até efetivamente se realizar o reconhecimento e consequentemente a transcrição da fala, o sistema, passa por várias etapas, dentro dos módulos principais do sistema (vide capítulo 3). O processo inicia-se no *Front-End* onde começa pela captura e transdução do sinal analógico para o sinal digital que logo em seguida passa por uma técnica para extração de parâmetros da fala (MFCC), eliminando os ruídos. O próximo passo seria reunir todas as informações necessárias para realizar o reconhecimento da fala dita. Nessa etapa, o *engine* carrega as informações do Modelo Acústico e da Gramática. As etapas do processo de funcionamento do sistema podem ser esclarecidas no diagrama de atividades da figura 4.3

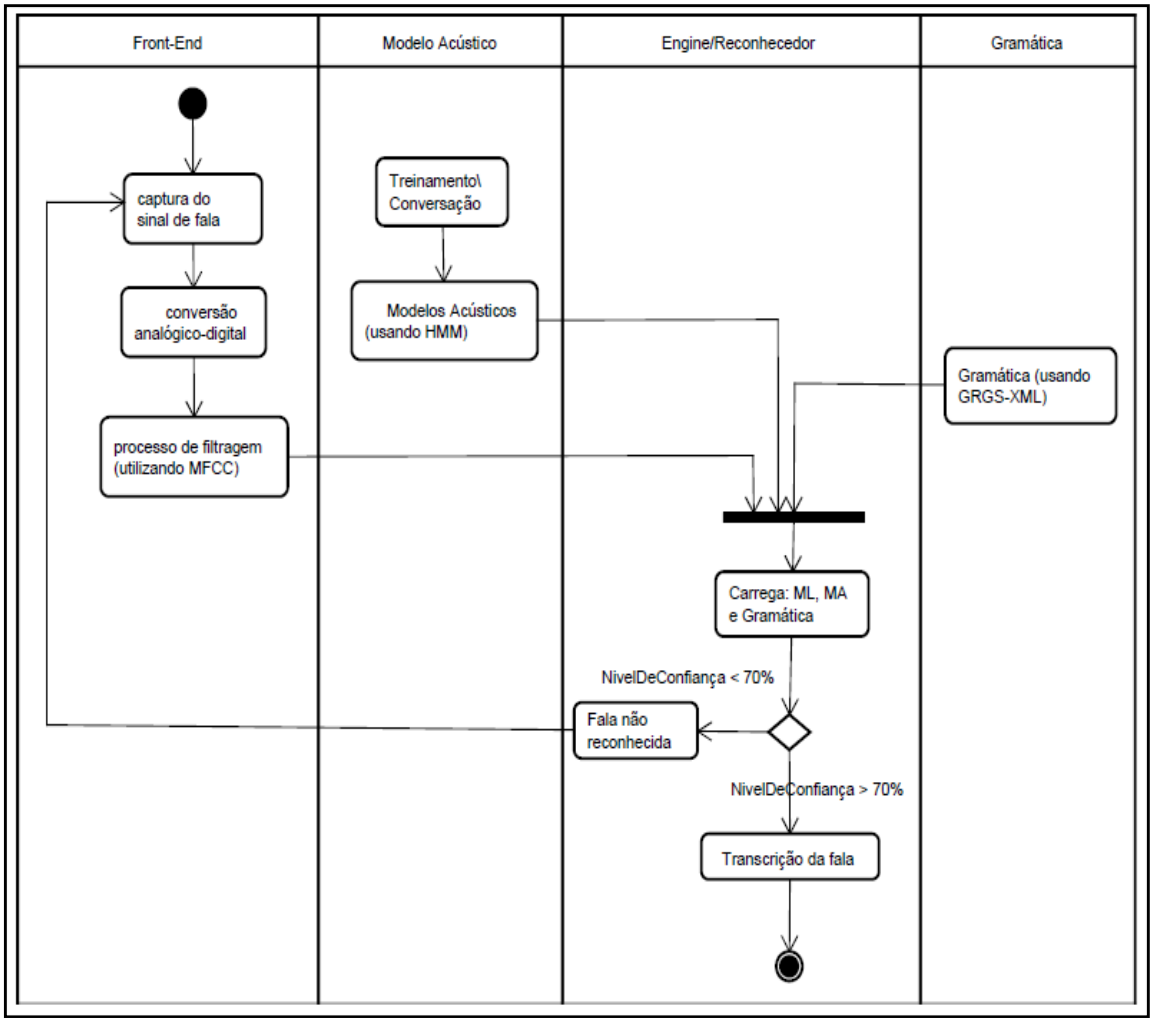


Figura 4.3 *– Diagrama de Atividades do Sistema*

4.3. INTERFACE DE PROGRAMAÇÃO DE APLICATIVOS UTILIZADA

Através da praticidade oriunda de uma Interface de Programação de Aplicativos (API) que foi implementada através do pacote de distribuição do Julius pelo LAPS, foi possível de forma prática desenvolver uma aplicação que por meio das várias tecnologias descritas no capítulos anteriores pudesse realizar o reconhecimento de fala independente de locutor para um vocabulário discreto.

A API foi desenvolvida em linguagem de programação C++ com a especificação *Common Language Runtime* (CLR), que permite a comunicação entre as linguagens suportadas pela plataforma .NET [27,28], como por exemplo, a linguagem C#. A API permite controle em tempo real do *engine* Julius e da interface de áudio do sistema. Resumidamente a

API mascara detalhes da implementação do *software*, e apenas provê recursos para usar seus serviços, como pode ser visto na figura 4.4.

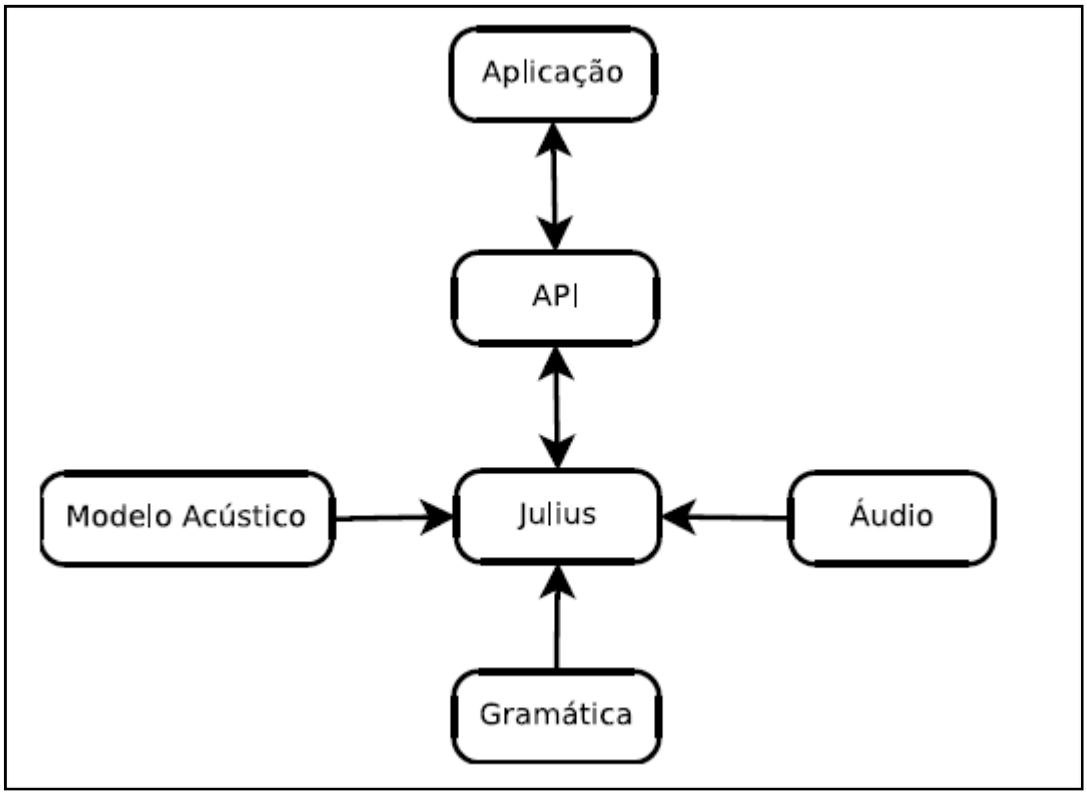


Figura 4.4 *– Modelo de Interatividade entre a API e demais Módulos do Sistema*

FONTE: LAPS

A API atualmente suporta somente ambiente Windows. Visto que a API suporta objetos compatíveis com o modelo de automação COM (*Component Object Model*) [29], é possível acessar, manipular, ajustar propriedades e invocar métodos de objetos de automação compartilhados que são exportados por outras aplicações. Em termos de codificação a API consiste de uma classe principal denominada *SREngine*. Essa classe provê à aplicação uma gama de métodos e eventos que são descritos na tabela 4.1.

Tabela 4.1 *– Principais Métodos e Eventos da API*

Método/Evento	**Descrição Básica**
SREngine	Método para carregar e inicializar o reconhecedor
startRecognition	Método para iniciar o reconhecimento
stopRecognition	Método para pausar/parar o reconhecimento
OnRecognition	Evento chamado quando alguma sentença é reconhecida
OnSpeechReady	Evento chamado quando o reconhecimento é ativado
loadGrammar	Carrega as informações da gramática

O método construtor *SREngine* permite que a aplicação controle aspectos do reconhecedor Julius. Esse método possibilita que a aplicação carregue o modelo acústico e de linguagem a serem utilizados, inicie e pare o reconhecimento e receba eventos e resultados provenientes do *engine* de reconhecimento.

Uma aplicação RAF precisa carregar e ativar uma gramática que essencialmente indica o método de reconhecimento empregado, nesse caso para carregar a gramática usa-se o método *loadGrammar*.

O método *startRecognition*, responsável por iniciar o processo de reconhecimento, ativa as regras gramaticais e abre o *stream* de áudio. Similarmente, o método *stopRecognition* desativa as regras e fecha o *stream* de áudio.

O evento *OnSpeechReady* sinaliza que o *engine* está ativado para reconhecimento. Em outras palavras, ele surge toda vez que o método *startRecognition* é invocado. Já o evento *OnSRecognition* ocorre sempre que o resultado do reconhecimento encontra-se disponível, juntamente com o seu nível de confiança (consiste na verossimilhança do que foi dito com a palavra existente na gramática). Para atingir os objetivos desse projeto o nível de confiança (ou verossimilhança) foi definido em 70%, por ser normalmente o valor usado em aplicações desse tipo.

O nível de confiança do que foi reconhecido pelo sistema é essencial para aplicações reais, considerando que sempre poderão ocorrer erros no reconhecimento e, nesse caso, então influencia diretamente na transcrição do que foi reconhecido. A sequência de palavras e seu nível de confiança são passadas da API para aplicação através da classe *RecoResult*.

A API conta com cinco DLL's: *julius.dll* e *sent.dll* que possuem código fonte do *engine* Julius, *XML2grammarConverter.dll* e *mkfa.dll* que são responsáveis para a conversão da gramática no formato SRGS-XML para o formato nativo do Julius (ABFN) e a *LapsAPI.dll* que é efetivamente a API de desenvolvimento construída pelo LAPS [30], como pode ser visto na figura 4.5.

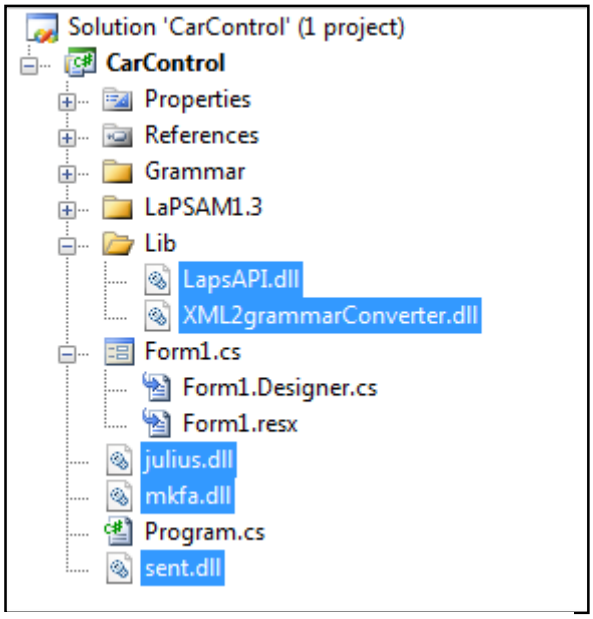


Figura 4.5 *– Principais DLL's do Projeto*

O construtor *SREngine* inicializa o Julius enviando como argumento o arquivo de configuração *jConf*. Esse arquivo contém as especificações de todos os recursos utilizados no Julius durante o processo de reconhecimento, como: modelos acústicos e a gramática livre de contexto.

4.5. CONSTRUÇÃO DA GRAMÁTICA

Na confecção da gramática para o projeto, foi escolhido o formato de gramática seguindo o padrão de especificação SRGS-XML (vide seção 3.1.3.3), conforme definido pelos padrões W3C, por se tratar de um padrão mais simples e de vasta documentação. A gramática especificada é muito trivial, e traz somente as palavras que o sistema RAF poderá

reconhecer, limitando assim a vasta quantidade de palavras que podem ser ditas pelo locutor e evitando erros de transcrição de palavras que não estão contidas na gramática da aplicação.

A gramática possui cinco palavras, que são: FRENTE, TRÁS, DIREITA, ESQUERDA e PARAR, que são respectivamente os comandos que irão fazer com que o carrinho interaja no ambiente logo após o reconhecimento e transcrição da fala.

Para a especificação da gramática devem-se seguir regras de sintaxe que são definidas por *tags* de início e fim que determinam cada elemento dentro da gramática, por se tratar de uma gramática trivial e utilizada para sistemas RAF do tipo vocabulário discreto, existe uma regra gramatical que é definida entre as *tags* <RULE> e </RULE> respectivamente, dentro dessa regra foi então definida uma lista de valores, que nesse caso são as cinco palavras propriamente ditas, que são definidas entre as *tags* <LIST> e </LIST> e finalmente para separar cada palavra uma da outra na lista, usa-se as *tags* <P> e <\P>. A gramática criada é mostrada na figura 4.6.

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <GRAMMAR>
3    <RULE>
4      <LIST>
5        <P>FRENTE</P>
6        <P>TRAS</P>
7        <P>DIREITA</P>
8        <P>ESQUERDA</P>
9        <P>PARAR</P>
10     </LIST>
11   </RULE>
12 </GRAMMAR>
```

Figura 4.6 *– Gramática utilizada pela Aplicação*

4.6. FUNCIONAMENTO DO MODELO ACÚSTICO

O modelo acústico é um arquivo que contém representações estatísticas de cada um dos sons distintos que compõem uma palavra. Cada uma destas representações estatística é chamada de fonema.

Um modelo acústico é criado e exige um grande banco de dados de fala (chamado de corpus de fala) e usando algoritmos de treinamento especial para criar representações estatísticas para cada fonema de uma língua. Estas representações são criadas usando as técnicas de Modelos Ocultos de Markov (HMM's, como visto no capítulo 3). Cada fonema tem seu próprio HMM, por exemplo, se o sistema está configurado com uma gramática simples para reconhecer a palavra "frente" que possui seis letras (f-r-e-n-t-e) e cinco fonemas ("f-r-en-t-e"), abaixo estão definidos os passos (simplificados) que o *engine* de reconhecimento de fala pode considerar:

- O reconhecedor de fala escuta os distintos sons falados por um locutor e em seguida, busca um HMM correspondente no modelo acústico. No exemplo, cada um dos fonemas ("f-r-en-t-e") da palavra "frente" tem seu próprio HMM;
- Quando se encontra um HMM correspondente no modelo acústico, o reconhecedor armazena o fonema. O reconhecedor rastreia os fonemas correspondentes até que ele atinja uma pausa na fala do locutor;
- O reconhecedor em seguida, busca na gramática uma palavra ou frase correspondente, como no exemplo da tabela 4.2 que mostra a lista das palavras da gramática e seus fonemas respectivos.

Tabela 4.2 *– Exemplo: Fonemas da palavra dita encontrado na gramática*

[DIREITA]	D-I-R-E-I-T-A
[ESQUERDA]	E-S-QU-E-R-D-A
[PARAR]	P-A-R-A-R
[FRENTE]	**F-R-EN-T-E**
[TRÁS]	T-R-A-S

O processo de reconhecimento fica um pouco mais complicado quando começa a se usar um Modelo de Linguagem (que contém as probabilidades de um grande número de

seqüências de palavras diferentes, ou seja, um vocabulário muito grande), mas segue a mesma abordagem. Um modelo de linguagem é normalmente usado para sistemas do tipo LVCSR.

O Modelo Acústico utilizado nesse projeto foi criado com o software HTK, graças aos esforços do grupo Fala Brasil do LAPS [30]. Para treino foi utilizado o corpus *LapsStory* combinado com as técnicas MLLR+MAP para adaptação de ambiente. Para adaptação foi utilizado todo corpus *Spoltech*. O tipo paramétrico utilizado foi o MFCC_E_D_A_Z [30].

4.7. RESULTADOS OBTIDOS

Inicialmente foram realizados três tipos de teste, o primeiro observando a diversidade de vozes, ou seja, fazendo o uso de locutores do sexo feminino e masculino de faixas etárias entre 18 a 40 anos de idade, para tal, o sistema foi testado por 4 mulheres e 4 homens.

O segundo teste consistiu em medir o quão o reconhecedor é capaz de reconhecer palavras de sua gramática em ambientes ruidosos, para isso foram usadas algumas taxas de SNR para simular ambientes ruidosos, sendo que a voz foi de um único locutor.

O terceiro teste media o tempo médio de resposta do sistema para o reconhecimento da fala nos outros dois cenários anteriores.

1. O primeiro teste consiste em adquirir os níveis de confiança obtidos pelo reconhecedor para cada palavra correta dita pelos locutores (homens e mulheres) e que estejam na gramática e, além disso, também obter os valores de palavras que não estejam na gramática. As palavras utilizadas além das que foram definidas na gramática foram: casa, engenho, cachorro.

Tabela 4.3 *– Teste de níveis de confiança para locutores femininos*

	FRENTE	TRÁS	DIREITA	ESQUERDA	PARAR	CASA	ENGENHO	CACHORRO
Loc. 1	0.996038	0.7010467	0.9567577	0.9968599	0.9224483	0.3972342	0.504701	0.4741387
Loc. 2	0.98602	0.8610463	0.9867456	0.9968700	0.9256481	0.4972300	0.404800	0.4332390
Loc. 3	0.81165	0.8934222	0.963863	0.9990853	0.8861187	0.4546807	0.518235	0.5855488
Loc. 4	0.90063	0.9344232	0.999861	0.9590100	0.9001485	0.3746920	0.423243	0.5659000

Tabela 4.4 *– Teste de níveis de confiança para locutores masculinos*

	FRENTE	TRÁS	DIREITA	ESQUERDA	PARAR	CASA	ENGENHO	CACHORRO
Loc. 1	0.942226	0.906761	0.7225596	0.9726312	0.8047706	0.5923843	0.676384	0.5192723
Loc. 2	0.886034	0.8970461	0.9767500	0.9968700	0.9156567	0.5972344	0.5446800	0.4392121
Loc. 3	0.913943	0.9540179	0.8496834	0.9976766	0.7835052	0.4636432	0.5564411	0.4157359
Loc. 4	0.887634	0.9944343	0.987986	0.9890701	0.8781567	0.3996932	0.4424377	0.4457201

Para os testes de diversidade de fala, pode-se observar que o reconhecedor funciona muito bem, principalmente no reconhecimento entre as palavras dentro da própria gramática (o que era de se esperar), não havendo muitos reconhecimentos e transcrições errôneas, como pode ser visto nas tabelas 4.3 e 4.4 respectivamente. Contudo nas palavras que não estão dentro da gramática principalmente a palavra “engenho” houve momentos em que se transcreveu a palavra “frente”, porém com uma taxa de acerto em não transcrever a palavra, girando em torno de 75%, o que ainda assim é aceitável.

2. O segundo teste teve como objetivo adquirir os valores de níveis de confiança para um só locutor, no entanto os testes foram realizados em vários “ambientes” diferentes simulados através de valores de SNR: 5db, 10db e 15db.

Tabela 4.5 *– Teste de níveis de confiança em ambientes ruidosos*

	5db	**10db**	**15db**
Loc. 1	0.8496834	0.7010467	0.676384

Os testes em ambientes ruidosos não foram muito satisfatórios principalmente para ruídos acima de 15 decibéis, é perceptível que o reconhecedor não se comporta muito bem quando se trata de reconhecer palavras em ambientes muito barulhentos, no entanto os testes para relações de sinal ruído (SNR) em ambientes não tão barulhentos se mostram bem satisfatórios como visto na tabela 4.5.

3. Nos testes de tempo de resposta, o objetivo foi obter os valores médios de tempo de resposta gastos para se reconhecer um comando de fala, sendo que os mesmos foram realizados baseando-se em diferentes locutores homens e mulheres (teste 1) e em ambientes ruidosos (teste 2).

***Tabela 4.6** – Teste de tempo médio gasto para o reconhecimento*

	Cenário 1 (somente palavras da gramática)	**Cenário 2**
Média de Tempo	~ 3segundos	~ 3segundos

A média de tempo gasto para o reconhecimento das palavras em ambos os ambientes não variaram muito, apesar de não ser um valor ideal, para o efeito desse trabalho os valores de tempo encontrados foram aceitáveis como pode serem vistos na tabela 4.6.

4.8. DETALHES DE *HARDWARE* E *SOFTWARE*

Para atender aos requisitos do projeto, foram selecionados e utilizados diversos equipamentos e ferramentas para o desenvolvimento. Os detalhes de *hardware* e *software* utilizados nas simulações, desenvolvimento e testes estão especificados na tabela 4.7.

***Tabela 4.7** – Especificações de Hardware e Software*

Hardware	
Processador (x64):	Intel(R) Core(TM)2 Duo CPU T6400 @ 2.00GHz, 1995 MHz
Memória:	4096 Mb DDR2 SDRAM
Placa de Áudio (Integrada):	Intel 82801IB ICH9 - High Definition Audio Controller [A-3] PCI
Microfone:	Phillips SHM3300

Software	
Sistema Operacional:	Windows Seven (x64)
Ambiente de Programação (IDE):	Microsoft Visual Studio 2008 v 9.0
Linguagem de programação:	Microsoft Visual C# 2008
Obs.: As dll's do projeto foram compiladas em arquitetura x86 (32 bits).	

CAPÍTULO V

CONCLUSÃO

As interfaces de sistemas via fala estão rapidamente se tornando uma necessidade. Em um futuro próximo, sistemas interativos irão fornecer fácil acesso a milhares de informações e serviços que irão afetar de forma profunda a vida cotidiana das pessoas, por isso a proposta de se elaborar um trabalho para o reconhecimento de fala, principalmente para o português brasileiro, já que a maioria das tecnologias criadas são normalmente para a língua inglesa.

Neste trabalho foram estudados alguns aspectos da teoria referente ao reconhecimento de fala, com ênfase especial ao problema de reconhecimento de fala discreta com vocabulário pequeno e independência do locutor. Todas as etapas da construção de um sistema RAF foram abordadas nesse projeto, desde o projeto e confecção de uma gramática, passando por todas as ferramentas necessárias ao tratamento dos dados, até o desenvolvimento final do sistema. Proporcionando uma boa compreensão das questões envolvidas em cada uma das etapas do desenvolvimento do sistema, terminando em uma pequena aplicação bastante amigável que demonstra a aplicabilidade prática do reconhecimento da fala por computador.

Nos testes para o reconhecimento dos comandos de fala, o sistema se mostrou bem funcional tanto para palavras ditas por mulheres como por homens, tendo em média taxas de 92% de confidência para as palavras da gramática, como pode ser visto no gráfico da figura 4.7 (cor azul). Nos testes algo interessante a ser salientado foi a dificuldade do reconhecedor em não transcrever a palavra “engenho”, que na maioria das vezes era transcrita erroneamente como “frente”, possivelmente a dificuldade é oriunda da própria pronuncia da palavra, já que além de ter um fonema vocálico nasal (/en/) possui um dígrafo (letras que, juntas, representam um único fonema: /nh/).

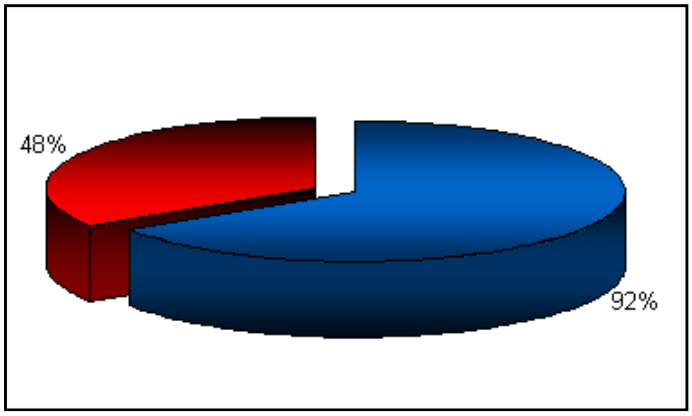


Figura 5.1 *– Palavras Reconhecidas X Palavras Não reconhecidas*

Para os testes envolvendo palavras fora do vocabulário, o sistema também se mostrou bem eficiente, trazendo em média taxas de 48% de confiança para as palavras tanto ditas por mulheres como por homens, como pode ser visto no gráfico da figura 4.7 (cor vermelha). que graças a limitação do grau mínimo de confiança para se transcrever uma palavra, que nesse projeto foi definido como acima de 70%, não permitia que palavras fora do vocabulário pudessem ser transcritas.

Para trabalhos futuros sugere-se: utilizar a técnica de ZCPAC para extração de parâmetros no *front-end*, já que estudos realizados comprovam que ela parece mais eficaz que a MFCC em ambientes mais ruidosos, utilização de modelo de linguagem para se criar um reconhecedor do tipo fala contínua. Como adaptação desse trabalho, sugere-se aplicar o reconhecimento da fala convertido em ações em uma cadeira de rodas motorizada para deficientes tetraplégicos que não têm condições de se locomover por meio da cadeira de rodas manuais.

REFERÊNCIAS BIBLIOGRÁFICAS

[1]. **H. F. Olson and H.Belar**, Phonetic Typewriter, J. Acoust. Soc. Am., 28(6): 1072-1081, 1956.

[2]. **D. B. Fry**, *Theoretical Aspects of Mechanical Speech Recognition; and P. Denes, The Design and Operation of the Mechanical Speech Recognizer at University College London*, J. British Inst. Radio Engr., 19: 4,211-229,1959.

[3]. **J. Ferguson**, Ed., *Hidden Markov Models for Speech*, IDA, Princeton, NJ, 1980.

[4]. **Ynoguti,Carlos Alberto**. Reconhecimento de fala Continua Usando Modelos Ocultos de Markov. Unicamp, 1999.

[5]. **Bresolin, Adriano de Andrade**. Estudo do reconhecimento de voz para o acionamento de equipamentos elétricos via comandos em português, UDESC, 2003.

[6]. **L.R. Rabiner and B. H. Juang**. *Fundamentals of Speech Recognition. Englewood Cliffs, Prentice-Hall*, 1993.

[7]. **Martins, José Antônio**. Avaliação de Diferentes Técnicas para Reconhecimento de Fala, Unicamp, 1997.

[8]. **ALENCAR, V. F. S.** Atributos e domínios de interpolação eficientes em reconhecimento de voz distribuído. Dissertação de mestrado. PUC - Rio, 2005.

[9]. **A. Nejat Ince, editor.** *Digital Speech Processing, Speech Coding, Synthesis and Recognition*. Kluwer Academic Plubishers, 1992.

[10]. **L.RabinerandR.Schafer**, *Digital Processingof Speech Signals Prentice-Hall*,1978.

[11]. **Patrick Silva**, Sistemas de Reconhecimento de Voz para o Português brasileiro utilizando os Corpora Spoltech e OGI-22, Trabalho de conclusão de curso, Universidade Federal do Pará, Instituto de Tecnologia, 2008.

[12**]. BAKER, J. K.** *Stochastic modeling for automatic speech understanding in Speech Recognition*. Reddy, D. R. ed. New York.

[13]. **JELINEK, F., BAHL, L. R.**, and **MERCER, R. L**. *Design of linguistic statical decoder for the continous speech. IEEE transactions on information Theory*. May, 1975.

[14]. **RAIBINER, L.** *Fundamentals of speech recognition. Prentice Hall Press*. 1993.

[15]. **Produção de Fala Humana.** *DEETC – Departamento de Engenharia e Eletrónica e Telecomunicações e de Computadores* Disponível em: <http://www.deetc.isel.ipl.pt/comunicacoesep/disciplinas/pdf/sebenta/pdf/producao_2.pdf>. Acessado em: Fevereiro de 2010.

[16]. **S. Davis and P. Merlmestein**, *Comparison of parametric representations for monosyllabic Word recognition in continously spoken sentences.* IEE Trans. On ASSP, Aug. 1980.

[17]. **B. Gajic**, e **K. Paliwal,Robust,** *Speech Recognition Using Features Based On Zero Crossing With Peak Amplitude*, ICASSP 2003, (2003), 64-67.

[18]. ***HTK.*** *The Hidden Markov Model Toolkit*. Disponível em: <http://htk.eng.cam.ac.uk/>

Acessado em: Abril de 2010.

[19]. ***Cambridge University Engineering Department***. *The HTK Book*. Disponível em: <http://www.spsc.tugraz.at/courses/scl/download/htkbook.pdf>, Acessado em: Maio de 2010.

[20]. **Lawrence R. Rabiner.** *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE, vol.77, no.2, pp.257–86, Feb.1989.

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

[22]. **W3C-SRGS v1. 0.** *Speech Recognition Grammar Specification Version 1.0*. Disponível em: <http://www.w3.org/TR/speech-grammar/>, Acessado em Maio de 2010.

[23]. **SRGS.** *Speech Recognition Grammar Specification*. Disponível em: <http://en.wikipedia.org/wiki/Speech_Recognition_Grammar_Specification>, Acessado em Maio de 2010.

[24]. **Microsoft Speech API 5.3.** *Text Grammar Format*. Disponível em: <http://msdn.microsoft.com/pt-pt/library/ms723632%28v=VS.85%29.aspx>, Acessado em Maio de 2010.

[25]. **Nelson Cruz Sampaio Neto.** Desenvolvimento de Aplicativos Usando Reconhecimento e Síntese de Voz, UFPA -2006.

[26]. **Julius.** *The Julius book rev.4.1.2*. Akinobu LEE, *part 4.1.2 - ENG*. Disponível em: <http://jaist.dl.sourceforge.jp/julius/37581/Juliusbook-part-4.1.2-en.pdf>, Acessado em Maio de 2010.

[27]. **MSDN - .NET Framework Developer Center.** .NET - *Microsoft Developer Network*. Disponível em: <http://msdn.microsoft.com/pt-pt/netframework/default(en-us).aspx>, Acessado em Maio de 2010.

[28]. **Monteiro Monique**. Introdução à Plataforma Microsoft .NET. UFPE. Disponível em: <http://www.cin.ufpe.br/~mlbm/download/DotNet.ppt>, Acessado em Março de 2010.

[29]. **COM Objects**. Component Object Mode. Disponível em: <http://www.microsoft.com/com/default.mspx>, Acessado em Março de 2010.

[30]. **Fala Brasil - LAPS**. Laboratório de Processamento de Sinais. UFPA. Disponível em: <http://www.laps.ufpa.br/falabrasil/index.php>, Acessado em Fevereiro de 2010.

[31]. **Wikipédia Pt-Br**. Corpus linguístico. Disponível em: <http://pt.wikipedia.org/wiki/Corpus>, Acessado em Maio de 2010.

[32]. **SRILM**. The SRI Language Modeling Toolkit. Disponível em: <http://www.speech.sri.com/projects/srilm/>, Acessado em Maio de 2010.

APÊNDICE

CÓDIGOS-FONTE DA APLICAÇÃO (RECONHECIMENTO DE FALA)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using LapsAPI;

namespace CarControl
{
    public partial class Form1 : Form
    {
       SREngine engine = null;
       private delegate void writeRecognizedDelegate(string text);
       private delegate void changeCheckBoxDelegate(CheckBox box, bool check);
       private writeRecognizedDelegate write;
       private changeCheckBoxDelegate changeCheck;

        public Form1()
        {
            InitializeComponent();
            try
            {
                SREngine.OnRecognition += HandleResults;
                engine = new SREngine(@"LaPSAM1.3\ppt_com.jconf");

                engine.loadGrammar(@"Grammar\grammar.xml");

                write = new
writeRecognizedDelegate(recognizedRichTextBox.AppendText);
                changeCheck = new changeCheckBoxDelegate(ChangeCheckBox);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void startButton_Click(object sender, EventArgs e)
        {
            try
            {
                engine.startRecognition();
                startButton.Enabled = false;
                stopButton.Enabled = true;
                recognizedRichTextBox.AppendText("--Reconhecendo--\n");
            }
            catch (Exception ex)
            {
                startButton.Enabled = true;
                stopButton.Enabled = false;
                MessageBox.Show(ex.Message);
            }
        }

        private void ChangeCheckBox(CheckBox box, bool check)
        {
```

```
            if (check)
            {
                box.Checked = true;
                box.CheckState = CheckState.Checked;
            }
            else
            {
                box.Checked = false;
                box.CheckState = CheckState.Unchecked;
            }
        }

        private void HandleResults(RecoResult result)
        {


            if (result.getConfidence() > 0.70)
            {
                this.Invoke(write, new Object[] { result.getConfidence() + " | " +
result.getUterrance().ToUpper().Replace("<S>", "").Replace("</S>", "").Trim() +
"\n" });

    this.Invoke(changeCheck, new Object[] { frontCheckBox, false });
    this.Invoke(changeCheck, new Object[] { backCheckBox, false });
    this.Invoke(changeCheck, new Object[] { leftCheckBox, false });
    this.Invoke(changeCheck, new Object[] { rightCheckBox, false });
    this.Invoke(changeCheck, new Object[] { stopedCheckBox, false });


switch (result.getUterrance().ToUpper().Replace("<S>", "").Replace("</S>",
"").Trim())
                {
                    case "FRENTE":
              this.Invoke(changeCheck, new Object[] { frontCheckBox, true });
                        break;
                    case "TRAS":
              this.Invoke(changeCheck, new Object[] { backCheckBox, true });
                        break;
                    case "ESQUERDA":
              this.Invoke(changeCheck, new Object[] { leftCheckBox, true });
                        break;
                    case "DIREITA":
              this.Invoke(changeCheck, new Object[] { rightCheckBox, true });
                        break;
                    case "PARAR":
              this.Invoke(changeCheck, new Object[] { stopedCheckBox, true });
                        break;
                }
            }

            else
            {
this.Invoke(write, new Object[] { result.getConfidence() + " | " + "Não
reconhecido" + "\n" });
            }
        }

        private void pararButton_Click(object sender, EventArgs e)
        {
            try
            {
                engine.stopRecognition();
                startButton.Enabled = true;
                stopButton.Enabled = false;
                recognizedRichTextBox.AppendText("--Reconhecimento parado--\n");
            }
            catch (Exception ex)
```

```
            {
                startButton.Enabled = false;
                stopButton.Enabled = true;
                MessageBox.Show(ex.Message);
            }
        }

        private void btnLimpar_Click(object sender, EventArgs e)
        {
            recognizedRichTextBox.Clear();
        }



    }
}
```

CÓDIGOS-FONTE DA APLICAÇÃO (CARRINHO)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;


namespace CarControl
{
    public partial class Form1 : Form
    {
        private const char ESQUERDA = 'A';
        private const char DIREITA = 'D';
        private const char ACIMA = 'W';
        private const char ABAIXO = 'S';
        private const char PARAR = 'P';

        private Thread thread = null;
        private volatile bool podeParar = false;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_KeyPress(object sender, KeyPressEventArgs e)
        {
            podeParar = true;

            if (!(thread == null)) { thread.Abort(); }

            if (Char.ToUpper(e.KeyChar).Equals(ESQUERDA)) {
```

```
                pictureBox1.Image =
Image.FromFile("D:/Imagens/Esquerdam.png");
                thread = new Thread(this.irParaEsquerda);
                podeParar = false;
                thread.Start();
            }
            if (Char.ToUpper(e.KeyChar).Equals(DIREITA ))
            {
                pictureBox1.Image =
Image.FromFile("D:/Imagens/DIREITAm.png");
                thread = new Thread(this.irParaDireita);
                podeParar = false;
                thread.Start();
            }
            if (Char.ToUpper(e.KeyChar).Equals(ACIMA ))
            {
                pictureBox1.Image =
Image.FromFile("D:/Imagens/Acimam.png");
                thread = new Thread(this.irParaAcima);
                podeParar = false;
                thread.Start();
            }
            if (Char.ToUpper(e.KeyChar).Equals(ABAIXO ))
            {
                pictureBox1.Image =
Image.FromFile("D:/Imagens/Abaixom.png");
                thread = new Thread(this.irParaAbaixo);
                podeParar = false;
                thread.Start();
            }
            if (Char.ToUpper(e.KeyChar).Equals(PARAR))
            {
                thread.Abort();
            }

        }

        private void irParaEsquerda() {
            while (!podeParar) {
                Thread.Sleep(500);
                this.pictureBox1.Left -= 10;
            }
        }

        private void irParaDireita() {
            while (!podeParar)
            {
                Thread.Sleep(500);
                this.pictureBox1.Left += 10;
            }
        }

        private void irParaAcima() {
            while (!podeParar)
            {
                Thread.Sleep(500);
                this.pictureBox1.Top -= 10;
            }
        }
```

```
        private void irParaAbaixo() {
            while (!podeParar)
            {
                Thread.Sleep(500);
                this.pictureBox1.Top += 10;
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            pictureBox1.Image = Image.FromFile("D:/Imagens/Direitam.png");
        }


        private void Form1_FormClosing(object sender, FormClosingEventArgs
e)
        {
            if (thread == null) { thread.Abort(); }
        }
    }
}
```

GRAMÁTICA

```
<?xml version="1.0" encoding="utf-8" ?>
<GRAMMAR>
  <RULE>
    <LIST>
      <P>FRENTE</P>
      <P>TRAS</P>
      <P>DIREITA</P>
      <P>ESQUERDA</P>
      <P>PARAR</P>
    </LIST>
  </RULE>
</GRAMMAR>
```