UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA

LUÍS GUSTAVO LUDESCHER
RAPHAEL MIELLE TRINTINALIA
VITOR ARAUJO ROMERA

ROLETA RUSSA

Jogo de Perguntas e Respostas com Reconhecimento de Fala

São Paulo
2009

LUÍS GUSTAVO LUDESCHER
RAPHAEL MIELLE TRINTINALIA
VITOR ARAUJO ROMERA

ROLETA RUSSA

Jogo de Perguntas e Respostas com Reconhecimento de Fala

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo, como parte dos requisitos necessários para a obtenção do título de Engenheiro de Computação.

São Paulo
2009

LUÍS GUSTAVO LUDESCHER
RAPHAEL MIELLE TRINTINALIA
VITOR ARAUJO ROMERA

ROLETA RUSSA

Jogo de Perguntas e Respostas com Reconhecimento de Fala

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo, como parte dos requisitos necessários para a obtenção do título de Engenheiro de Computação

Orientador: Prof.ª Dra. Selma Shin Shimizu Melnikoff

Co-orientador: Prof. Dr. Mario Minami

São Paulo
2009

Aos nossos pais, responsáveis pela nossa vida, nossos valores, nossa educação, e aos nossos mestres, que nos trouxeram mais que o simples conhecimento, nos trouxe inspiração para seguirmos nossos caminhos daqui em diante.

OK

AGRADECIMENTOS

Diante desta obra, agradecemos aos nossos pais, que nos educaram, ensinaram os bons valores da ética e da humildade, que guiaram nossos caminhos na direção da Engenharia. A eles agradecemos pelo apoio, pelo incentivo e pela paciência.

Agradecemos aos familiares e bons amigos, que formaram nossa torcida pelo sucesso nesta jornada desde o primeiro até o último dia na Escola Politécnica.

À nossa orientadora Profª Drª Selma Shin Shimizu Melnikoff e ao nosso co-orientador Prof. Dr. Mário Minami, por terem nos guiado, auxiliado e cobrado para que este trabalho pudesse ser realizado.

Ao LPS (Laboratório de Processamento de Sinais) que nos ofereceu a infra-estrutura necessária para a gravação e o processamento das amostras de fala.

Por todos os professores que nos ajudaram durante nossa caminhada pela Poli e pelos nossos colegas de faculdade, futuros colegas de profissão.

Cabe também um agradecimento especial aos colegas Mark Hodgkin Pimentel, que nos auxiliou com a edição de áudio, a Edmar Miyake pelo empréstimo do equipamento e Renan Lotto Sacilotto e Rafael Weffort Lopes que estiveram conosco no início deste trabalho e deram sua colaboração.

"O néctar da vitória se bebe num cálice de sacrifícios"

Fidel Castro

RESUMO

O projeto visa desenvolver um jogo para celular, no estilo *Quiz*, como prova de conceito para o desenvolvimento de aplicativos que utilizem como interface com o usuário um algoritmo de reconhecimento de fala, ou seja, todos os comandos do aplicativo são acionados pela voz do usuário. A escolha deste modelo visa explorar o período gasto em veículos automotores durante congestionamentos, criando uma forma de entretenimento alternativa à radiodifusão, sem tirar a atenção do motorista ao volante ou limitar seus movimentos. Acredita-se que a construção de uma ferramenta desta natureza pode ser aproveitada no mercado de entretenimento, assim como no desenvolvimento de ferramentas voltadas a deficientes visuais e utilidades cotidianas.

Palavras-chave: Reconhecimento de fala. Aplicativos *Mobile*. Modelo Oculto de Markov (HMM). Decodificador de Viterbi. Amostragem de voz.

ABSTRACT

The project aims to develop a mobile game, in a quiz style, as proof of concept for developing applications that use as User Interface an algorithm for speech recognition, i.e. all the commands of the application are driven by the user's voice. The choice of this model is to exploit the time spent in motor vehicles during heavy traffic, creating an alternative form of entertainment related to radio without taking the attention of the driver at the handle or limit their movements. It is believed that the construction of such a tool can be used in the entertainment market, as well as in the development of tools aimed at the visually impaired and in everyday uses.

Key words: Speech Recognition. Mobile applications. Hidden Markov Models (HMM). Viterbi's Algorithm. Sampling Voice.

LISTA DE TABELAS

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIAÇÕES E SIGLAS

BD	Banco de Dados
DV	Decodificador de Viterbi
HMM	Hidden Markov Model (Modelo Oculto de Markov)
QV	Quantização Vetorial
RF	Reconhecimento de Fala

SUMÁRIO

1. Introdução

1.1. Motivação

Com o crescente aumento do mercado de dispositivos móveis no mundo, estes aparelhos estão cada vez mais presentes no cotidiano das pessoas. Segundo a companhia Huawei, fabricante de *smartphones*, os dispositivos móveis vendidos no Brasil chegam a 50 milhões por ano e se acompanharem a tendência de crescimento apresentada no mundo, chegaram a cerca de 13 milhões de *smartphones* abertos vendidos anualmente no Brasil (http://www.huawei.com/pt/).

Esse cenário promissor leva a atenção, cada dia maior por parte das empresas distribuidoras de *smartphones*. A RIM, detentora da marca *BlackBerry,* a *Apple*, fabricante do *IPhone* e o *Google* desenvolvedor do sistema operacional *Android* são exemplos deste crescimento. É de interesse comum das empresas diversificar o modo como o usuário interage com suas aplicações; hoje se encontra, como expoente máximo na interface com o usuário, o próprio *IPhone*, que tem como diferencial a utilização de toques diretos no visor em seu dispositivo. É buscando uma nova forma de interagir com o usuário que este trabalho foi desenvolvido. Há uma grande parcela de aparelhos baseados na plataforma *Windows Mobile*, que não conta ainda com uma tecnologia de reconhecimento de fala e fica em desvantagem em relação aos concorrentes no quesito usabilidade. Essa limitação, em uma plataforma conhecida, serviu de motivação para esse trabalho.

1.2. Objetivo

O objetivo deste trabalho é desenvolver uma prova de conceito para aplicativos que utilizem como interface homem-computador apenas comandos de voz, utilizando reconhecimento de fala. Desta forma, foi desenvolvido um jogo para celular, do tipo perguntas e respostas, que possa entreter as pessoas sem que elas precisem utilizar as mãos nem a visão. O jogo interage com o usuário

fazendo-lhe perguntas e obtendo as respostas faladas, através de sistema de reconhecimento de fala que reconhece as palavras pré-definidas que compõem o vocabulário definido para o jogo.

Tal software deverá ser integrado a sistemas de som automotivos, através do protocolo *Bluetooth*, o que poderá aumentar a interação do aplicativo com o usuário, visto que facilitaria a sua utilização, dado que geralmente os carros já possuem um sistema de som sendo que o software integrado a tal sistema teria um grande número de potenciais usuários.

1.3. Justificativa

Acredita-se que a construção de uma ferramenta desta natureza pode ser largamente aproveitada no mercado de entretenimento, inclusive para desenvolvimento de outros tipos de jogos. Uma ferramenta com reconhecimento de fala pode ser aproveitada numa infinidade de aplicações em diversas áreas. Facilmente pode-se imaginar como tal tecnologia pode ser aplicada para tornar mais prática a vida das pessoas no dia-a-dia, tornando dispensável o contato manual em diversos casos, como acender a luz, por exemplo. É importante ressaltar que tal tecnologia pode ser aproveitada no desenvolvimento de ferramentas voltadas a deficientes visuais, facilitando a vida dos mesmos.

Atualmente existem sistemas que envolvem reconhecimento de fala para, por exemplo, serviços de atendimento automático fornecidos por bancos para consulta de saldos, transferências, entre outros. Este é o caso da tecnologia de reconhecimento de fala da Dígitro, a qual tem duas modalidades de operação: reconhecimento de palavras isoladas ou de fala contínua, dependente ou independente do locutor.

1.4. Estrutura da Monografia

Além deste capítulo, esta monografia está organizada na seguinte forma:

- Processo de Reconhecimento de Fala (Capítulo 2)

Descreve a teoria envolvendo os métodos de reconhecimento de fala, compreendendo todo o procedimento e o embasamento necessário para a construção de um reconhecedor desta natureza.

- Descrição do Roleta Russa (Capítulo 3)

  Apresenta a especificação do sistema desenvolvido, através das idéias iniciais para o projeto, da estratégia do jogo, dos requisitos e da estrutura do software sem a parte de reconhecimento de fala.

- Desenvolvimento do Reconhecedor (Capítulo 4)

  Define a metodologia utilizada no desenvolvimento do reconhecedor de fala, através da descrição das etapas de desenvolvimento do reconhecedor, das ferramentas e das plataformas utilizadas.

- Integração com o Aparelho Móvel (Capítulo 5)

  Compreende o processo de aplicação dos valores de referência obtidos de forma experimental como entradas do processo implementado no aparelho móvel.

- Considerações Finais (Capítulo 6)

  Apresentam as conclusões, as contribuições referentes ao projeto e trabalhos futuros.

2. Processo de Reconhecimento de Fala

2.1. Definições básicas de reconhecimento de fala

O reconhecimento de fala ou discurso consiste no processo pelo qual um sistema automatizado identifica palavras faladas por um usuário. Basicamente, consiste em o sistema reconhecer corretamente os comandos fornecidos vocalmente pelo usuário.

As seguintes definições são necessárias para compreender a tecnologia de reconhecimento de fala.

Expressão:

Vocalização (dicção) de uma palavra ou de mais palavras que representem um único significado ao computador. As expressões podem ser uma única palavra, algumas palavras, uma sentença, ou mesmo sentenças múltiplas.

Dependência do locutor:

Os sistemas dependentes de locutores são projetados em torno de um locutor específico. São geralmente mais exatos para o locutor correto, e muito menos exatos para outros locutores. Supõem que o locutor fala em voz e em tempo consistentes, ou seja, de forma clara e objetiva. Os sistemas independentes de locutor são projetados para uma variedade de locutores. Os sistemas adaptáveis geralmente começam como sistemas independentes de locutor e utilizam técnicas de aprendizado para adaptar-se a um locutor e aumentar sua precisão no reconhecimento.

Vocabulários:

Os vocabulários (ou dicionários) são listas de palavras ou de expressões que podem ser reconhecidos pelo sistema de RF (Reconhecimento de Fala). Geralmente, os vocabulários menores, ou seja, um conjunto menor de palavras reconhecidas, são mais fáceis para que um computador reconheça, enquanto os vocabulários maiores (com maior número de vocábulos) são mais difíceis. Ao contrário dos dicionários normais, cada entrada não tem que ser uma única

palavra. Podem ser longas como uma sentença ou duas. Os vocabulários menores podem ter somente uma ou duas expressões reconhecidas (por exemplo, "acorde"), enquanto os vocabulários muito grandes podem ter cem mil ou mais expressões.

Precisão:

A habilidade de um reconhecedor pode ser examinada com a medição da sua precisão. Isto consiste em não somente identificar corretamente uma expressão, mas também identificá-la se não estiver em seu vocabulário. Os sistemas eficientes de RF têm uma precisão de 98% ou mais. A real exatidão aceitável de um sistema depende da aplicação.

Aprendizado:

Alguns reconhecedores de discurso têm a habilidade de adaptar-se a um locutor. Quando o sistema tem esta habilidade, ele contém tarefas específicas de aprendizado, sendo adaptado a cada utilização por parte do usuário. Um sistema de RF aprende tendo as frases padrão ou comuns repetidas pelo locutor e ajustando seus algoritmos de comparação para corresponder ao um locutor particular. Ensinar um reconhecedor geralmente melhora sua precisão, devido ao aumento no número de referências utilizadas para comparação.

O aprendizado pode também ser usado para locutores que têm dificuldade ao falar ou pronunciar determinadas palavras. Contanto que o locutor possa consistentemente repetir uma expressão, os sistemas RF com aprendizado devem adaptam-se. Esse tipo de reconhecedor é amplamente usado na área de fonoaudiologia.

2.2. Tipos de Reconhecimento de Fala

Os sistemas de RF podem ser classificados em função dos tipos de expressões que têm a habilidade de reconhecer. Esta classificação é baseada no fato de que uma das dificuldades no reconhecimento de fala é a habilidade de determinar o início e o término uma expressão dita pelo locutor. A maioria dos sistemas pode

estar inserida em mais de uma classe, dependendo da modalidade que está sendo utilizada.

Apresentam-se, a seguir, as definições relevantes ao tipo de reconhecimento de fala:

Palavras ou expressões isoladas:

Os reconhecedores de palavras isoladas requerem, geralmente em cada expressão, silêncio (falta de um sinal de áudio) em AMBOS os lados da janela (sinal compreendido entre os pontos limítrofes da palavra falada) de amostra. Freqüentemente, estes sistemas têm estados de escutando ou não escutando, ou seja, requerem que o locutor espere um intervalo de tempo entre as expressões (com o sistema realizando processamento durante as pausas).

Palavras ou expressões conectadas:

Os sistemas de palavras conectadas (ou expressões conectadas) são similares às de palavras isoladas, mas permitem que as expressões funcionem juntas com uma pausa mínima entre elas.

Fala contínua:

O reconhecimento contínuo é um processo mais elaborado. Os reconhecedores com capacidade de reconhecimento contínuo de discurso possuem maior dificuldade em sua construção, pois devem utilizar métodos especiais para determinar limites de expressão. Os reconhecedores contínuos de discurso permitem que os usuários falem de forma quase natural, enquanto o computador determina o conteúdo dito. Basicamente, é preceito do computador indicar qual o ponto de início e o fim de cada expressão.

Fala espontânea:

Em um nível básico, pode ser considerado como um discurso que soa natural e não recitado existindo variedade sobre o que um discurso espontâneo pode apresentar. Um sistema de RF com habilidade de discurso espontâneo deve manter uma variedade de características naturais de discurso tais como as

palavras de ligação, os "hums" e os "ahs", e mesmo sotaques e pequenas prolongações de sons.

2.3. Usos e aplicações de RF

Embora toda a tarefa que envolva interação com um computador possa potencialmente usar RF, as seguintes aplicações são as mais utilizadas no cotidiano.

Ditado:

O ditado é o uso o mais comum para sistemas de RF atualmente. Isto inclui transcrições médicas, legais e de negócios, bem como o processamento de palavras em geral. Em alguns casos os vocabulários especiais, que incluem jargões e termos técnicos, são usados para aumentar a precisão do sistema.

Comando e controle:

Os sistemas de comando e de controle são sistemas de RF projetados para executar funções e ações através dos sistemas. As expressões como "Netscape aberto" e "inicia um terminal novo" fariam apenas isso.

Telefonia:

Alguns sistemas de PABX/Caixa postal permitem que os usuários recitem comandos de voz em vez de usar teclas para a execução de comandos específicos.

Equipamentos sem fio:

Como algumas entradas para dispositivos sem fio são limitadas, o uso da fala é uma possibilidade natural para garantia de segurança no que diz respeito ao controle de acesso, assim como outras tecnologias utilizadas como o reconhecimento de face ou da impressão digital do usuário.

Uso adaptativo / inabilidades:

Muitas pessoas têm dificuldade ao datilografar devido a limitações físicas tais como lesões por esforço repetitivo (LER), distrofia muscular, e outras síndromes; elas poderiam interagir com sistemas através de RF. Pessoas com dificuldades na audição poderiam usufruir de um sistema conectado ao telefone que pode converter o discurso audível do telefone em texto.

Aplicações embutidas:

Grande parte dos telefones celulares inclui funções de reconhecimento de fala que permitem comandos tais como o de "ligar pra casa". Este pode ser o uso principal no futuro de reconhecimento de fala.

2.4. Funcionamento dos Sistemas de Reconhecimento de Fala

Os sistemas de RF podem ser classificados em dois tipos principais:

- Sistemas de reconhecimento de teste padrão: comparam modelos conhecidos para determinar uma combinação.
- Sistemas fonéticos acústicos: usam o conhecimento do corpo humano (voz e audição) e compara as características do discurso (fonética tal como os sons de vogais).

A maioria dos sistemas modernos foca no reconhecimento de teste padrão, já que este tipo de RF pode ser implementado de forma mais prática com técnicas computacionais atuais e tende a ter uma elevada taxa de acertos (BASHIR, F 2005).

A maioria dos identificadores segue as seguintes etapas (Rabiner, 1993):

- Gravação do áudio e detecção do discurso.
- Pré-filtragem do sinal de áudio
- Enquadramento ou janelamento
- Filtragem

- Comparação e combinação dos padrões encontrados
- Execução da ação

Embora cada etapa pareça simples, cada uma delas pode envolver uma variedade de técnicas diferentes, em algumas vezes opostas, ou seja, enquanto algumas técnicas podem utilizar ênfase em tons graves para detecção de consoantes, outras enfatizam os tons agudos para realce dos sons referentes às vogais. Apresenta-se, a seguir, uma descrição sucinta destas etapas.

Gravação do áudio e detecção da fala:

A detecção da fala pode ser realizada de diversas maneiras. O início de um discurso pode ser encontrado por comparação dos níveis de som ambiente considerado como ruído, com as amostras gravadas. A detecção do *endpoint* (final) da expressão se torna mais difícil, pois os locutores tendem a deixar vestígios como respiração, suspiros, rangidos de dentes, ou ecos, dificultando a ação do algoritmo.

A escolha do ambiente de gravação das amostras de referência deve estar relacionada ao uso final do produto que utilizará os recursos de RF. Por exemplo, um software de RF voltado à fins de auxílio em fonoaudiologia deve conter amostras de referência gravadas em ambiente silencioso, de preferência em estúdio. Em sistemas que operam em ambientes ruidosos, devem ser consideradas também amostras em ambientes semelhantes, ou seja, utilizando um ambiente natural simulado ao fundo.

Pré-Filtragem:

É realizada de diversas maneiras, dependendo das características do sistema de reconhecimento. Os métodos utilizados mais comumente são: o método Banco de Filtros - que utiliza uma série de filtros de áudio para preparar a amostra - e o *Linear Predictive Coding method* - que usa uma função de predição para calcular diferenças (erros). Diferentes formas de análise espectral também são usadas.

Janelamento:

Consiste em separar os dados da amostra em tamanhos específicos, filtrados por um fragmento de sinal denominado quadro ou janela. Isto é realizado freqüentemente nas etapas de pré-filtragem ou filtragem. Esta etapa envolve também a preparação dos limites da amostra para a análise (remoção de pequenos ruídos oriundos do movimento da boca ou ranger de dentes nas bordas, entre outros).

Filtragem adicional:

Esta atividade não é sempre realizada. É a preparação final para cada janela antes das etapas de comparação e combinação. Freqüentemente consiste no alinhamento e na normalização do tempo, isto é, as amostras sofrem um processo de normalização de acordo com um período médio de duração de uma palavra, eliminando inconsistências decorrentes de prolongamentos ou abreviações nas palavras amostradas, e também na conversão de escalas para análise (como as escalas Mel e Bark) e técnicas de quantização vetorial, a serem definidas posteriormente.

Há um número grande de técnicas disponíveis para comparação e combinação de amostras. A maioria envolve a comparação da amostra obtida com as amostras conhecidas. Existem métodos que utilizam o modelo oculto de Markov (HMM), a análise de freqüência, a análise diferencial, técnicas de álgebra linear, a distorção espectral, e os métodos de distorção do tempo. Todos estes métodos são usados para gerar uma probabilidade ou uma combinação exata (BASHIR, F 2005).

Nas próximas seções serão detalhadas as etapas do processamento para RF utilizadas neste projeto.

2.4.1. Processo de Pré-Ênfase

O processo de pré-ênfase de um sinal compreende a atenuação dos sinais de baixa freqüência fazendo com que os sinais agudos sejam destacados, equalizando o espectro da voz, de forma a melhorar o desempenho da análise espectral (Gómez-Cipriano).

Sendo x(T) o sinal de áudio captado e y(T) a saída da pré-ênfase, tem-se:

$$y(k) = x(k) - a * y(k-1), 0 < k < Tf$$

2-1

Inicialmente pode-se observar que a equação 2-1 possui forma discreta e caráter iterativo. Desta forma, para cada valor amostrado do sinal captado é subtraído o valor anterior calculado na saída (sendo y(0) nulo) modificado pelo fator a, resultando no valor de saída. A variável a é chamada de coeficiente de pré-ênfase, cujo valor é compreendido no intervalo 0,8 < a < 1.

2.4.2. Análise Cepstral

Após o processo de pré-ênfase é iniciado o processo de análise Cepstral que consiste na conversão do sinal em cepstros. Um cepstro é o produto da aplicação da transformada rápida de Fourier sobre o sinal em escala logarítmica. Esta conversão é feita seguindo os passos descritos a seguir:

i. Passagem pela função da transformada rápida de Fourier (Fast Fourier Transform – FFT)

ii. Transferência da escala de freqüência em escala logarítmica;

iii. Novo cálculo da transformada de Fourier (FFT) sobre a saída do passo (ii).

A equação 2-2 a seguir apresenta a transformada discreta de Fourier (DFT) a partir do qual é criado o algoritmo do FFT:

$$X(k) = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}}, k = 0, \ldots, N-1$$

2-2

O cálculo de um cepstro é dado pela equação a seguir:

$$Cepstrum = |FFT(\log(|FFT(y)|^2))|^2$$

2-3

Para o reconhecimento de fala, as transformadas de Fourier são realizadas em blocos do sinal recebido denominados janelas. O processo de construção das janelas é denominado janelamento e o algoritmo geralmente utilizado para esta função é o algoritmo de Hamming, cuja equação (2-4) encontra-se a seguir:

$$w(n) = 0{,}54 - 0{,}46 * \cos\frac{2\pi * n}{N_s - 1}, 0 < n < N_s - 1$$

2-4

Na equação de Hamming, N_s representa o número de amostras de uma janela n a amostra calculada.

Após o processo de janelamento é necessário o cálculo dos parâmetros mel-cepstrais das amostras de palavras obtidas. A escala Mel é uma escala subjetiva relativa às freqüências perceptíveis ao ouvido humano, que não seguem um perfil linear. Desta forma, é feita uma função de transferência para converter o sinal da escala convencional à escala Mel. Os coeficientes cepstrais na escala Mel que formam os cepstros são conhecidos por MFCC (Mel-Frequency Cepstral Coeficients).

A partir do cálculo do FFT em cada janela, aplica-se um banco de filtros de janelas triangulares, de forma a calcular a energia em cada canal do banco. A figura 2.1 apresenta o banco de filtros triangulares calculado para 20 quadros.

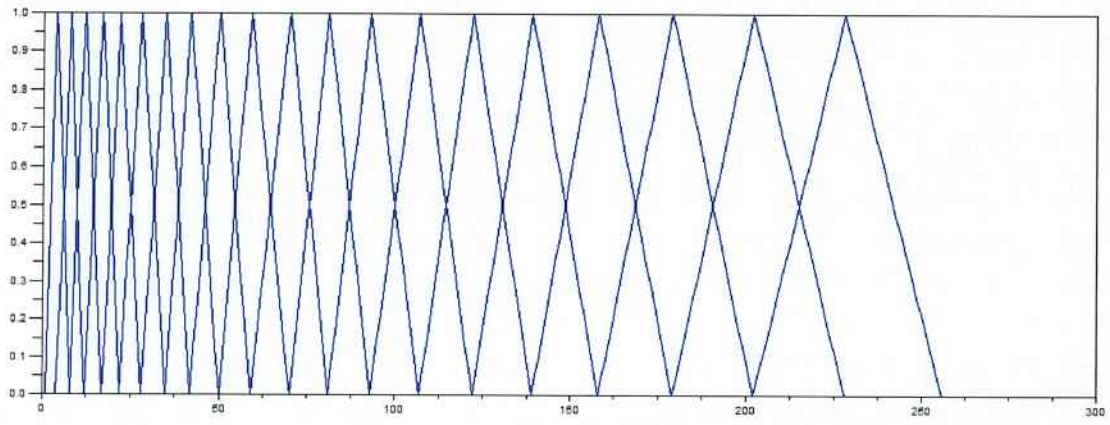


Figura 2-1 – Filtros triangulares para o cálculo dos coeficientes não-cepstrais feita no Scilab.

Após a filtragem, há a conversão dos valores à escala logarítmica, segundo a escala mel. Desta forma obtém a matriz referente ao MFCC.

2.4.3. Quantização Vetorial (VQ)

O processo de quantização vetorial consiste no mapeamento dos cepstros calculados na etapa anterior de forma a agrupar valores vetoriais de propriedades semelhantes, ou seja, um conjunto de sinais com o mesmo comportamento. O produto final da quantização vetorial é o *codebook*, representado como uma matriz determinada pela quantidade de bits utilizada na representação de cada valor, pelo número de *clusters*, ou perfis distintos determinados pelo construtor do reconhecedor.

O VQ reduz o volume de informações cepstrais decorrentes da análise, garantindo o mesmo nível de precisão; reduz também o tempo de processamento computacional para determinação da similaridade no processo de reconhecimento, além de representar sinais de áudio de maneira discreta. Por outro lado, o VQ pode distorcer os valores dos cepstros dependendo do número de bits utilizado para as representações. Quanto maior o número de bits utilizados, menor a distorção.

O processo de quantização pode ser dividido em quatro etapas, ilustradas de forma reduzida pela figura 2.2 (Rabiner, 1993):

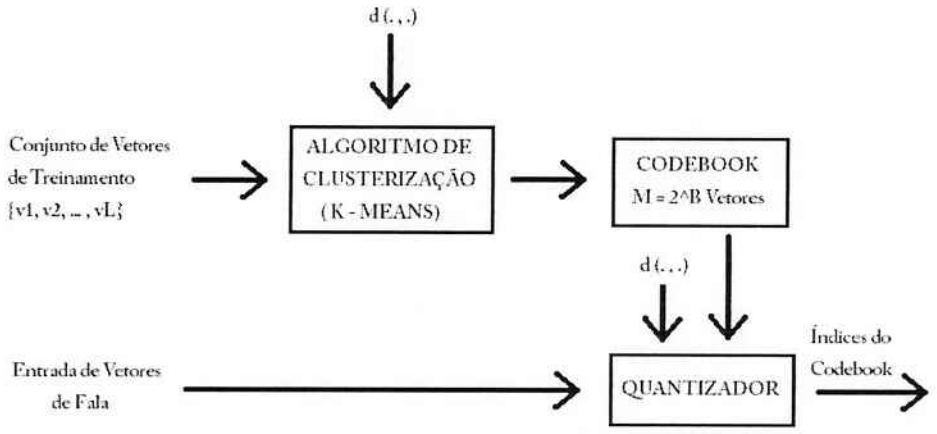


Figura 2-2 – Processo de Quantização Vetorial (Rabiner, 1993)

Treinamento

Os vetores cepstrais são analisados e processados de forma que seus valores sejam otimizados. O algoritmo de treinamento do VQ leva em consideração métricas relacionadas à idade, ao gênero, à intensidade da voz, níveis de ruído ao fundo, o efeito do transdutor relativo ao microfone usado para captação e unidades fonéticas. Assim, tem-se um conjunto de valores adequados para que sejam classificados e alocados futuramente em cada perfil.

Medição da similaridade

Os valores de análise espectral são analisados par a par com o intuito de que sejam calculadas as distâncias entre cada ponto para que posteriormente os cepstros sejam alocados de forma adequada.

Determinação dos centróides

De todos os valores obtidos com o vetor de análise cepstral, o VQ seleciona *M* pontos arbitrários não-pertencentes ao vetor de treinamento (sendo *M* o número determinado pelo projetista do reconhecedor para o número de *clusters)* que servirão como referência para a alocação dos valores cepstrais.

Classificação dos cepstros (clusterização)

Cada valor interno ao vetor de treinamento é analisado, assim como as distâncias calculadas anteriormente, acarretando na associação de um vetor a um centróide. Após a alocação de todos os valores, resulta-se o *codebook* das amostras obtidas. A figura 2.3 (Rabiner, 1993), a seguir, mostra a configuração dos centróides e um fluxograma, que exemplifica o processo de geração do *codebook.*

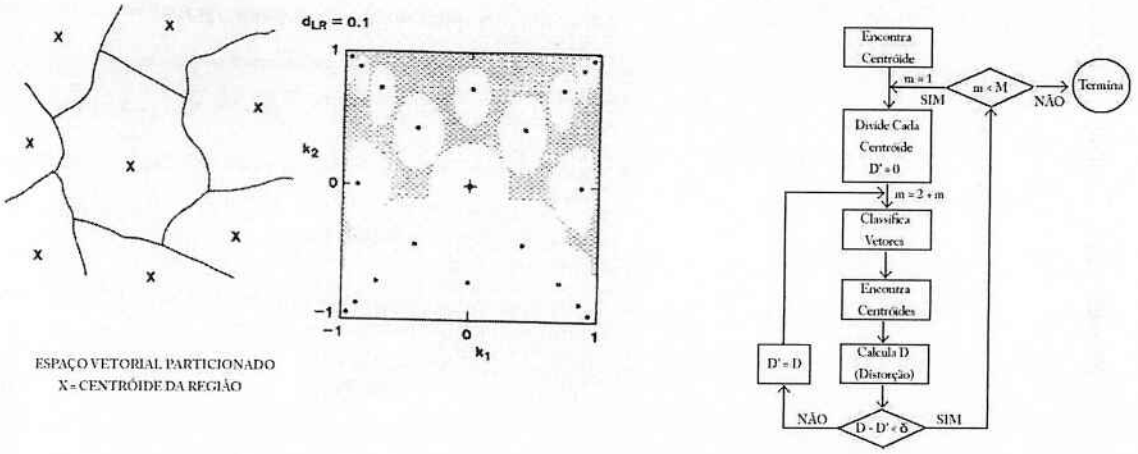


Figura 2-3 – (À dir.) Particionamento do espaço vetorial por clusters com determinação dos respectivos centróides. (À esq.) Processo de criação do codebook (Rabiner, 1993).

2.4.4. Cadeias de Markov (HMM)

O modelo oculto de Markov (Hidden Markov Models - HMM) pode ser descrito como um processo estocástico com pouca memória (FORSYTH; PONCE, 2002). Em um modelo de ordem n, o valor atual do processo é diretamente influenciado pelos n estados anteriores e não por toda a história de observações, caracterizando um ambiente não-determinístico. A principal função de um HMM, portanto, é modelar a probabilidade de uma seqüência de observações.

Cada estado do HMM é modelado como um evento e há possíveis transições que descrevem a probabilidade de mover-se para outro estado. Os Modelos de Markov são ditos ocultos porque os estados não são diretamente observáveis. Para cada estado existe a probabilidade de uma seqüência de observações (DUDA; HART; STORK, 2000). Os processos ocultos consistem de um conjunto de estados conectados por transições com probabilidades (autômato finito), enquanto que os processos observáveis (não escondidos) consistem de um conjunto de saídas ou observações, cada qual podendo ser emitido por um estado de acordo com alguma saída da função de densidade de probabilidade (fdp). Dependendo de sua fdp, várias classes de HMMs podem ser distinguidas, a seguir:

Discreta:

Observação discreta por natureza ou discretizada por vetor quantizado produzindo assim um alfabeto ou *codebook*.

Contínuo

Observação contínua, com sua fdp contínua usualmente aproximada para uma mistura de distribuição normal.

Semi-contínuo

Entre o discreto e o contínuo (híbrido).

O HMM tem se tornado, recentemente, a abordagem predominante para o reconhecimento da fala. Esses modelos estocásticos têm sido mostrados particularmente bem adaptados para caracterizar a variabilidade envolvida em sinais que variam no tempo. A maior vantagem do HMM situa-se na sua natureza probabilística, apropriada para sinais corrompidos por ruídos tal como a fala ou escrita, e na sua fundação teórica devido a existência de algoritmos poderosos para ajustar automaticamente os parâmetros do modelo através de procedimentos iterativos.

Um bom exemplo de como pode ser aplicado o HMM é descrito pelo caso das urnas e bolas (Rabiner, 1993). Imagine um sistema compreendido por N urnas, cada uma contendo certa quantidade de bolas coloridas, sendo possível encontrar M cores diferentes. Uma pessoa é então selecionada para subtrair sem ver as urnas uma bola qualquer. A pessoa irá observar que retirou uma bola de certa cor. Esta bola é devolvida à urna e esta é trocada de acordo com a bola retirada pela pessoa. O processo é então repetido por diversas vezes. Observe que a pessoa que retira as bolas tem como informação apenas uma seqüência de cores observadas, compreendidas pelo vetor $O = \{o_1, o_2, o_3, \ldots, o_k\}$, porém não tem conhecimento de qual das N urnas foi feita a subtração das bolas. Considerando cada urna como um estado e cada processo de subtração de uma bola uma observação seguida de uma transição de estados, tem-se que o sistema pode ser compreendido por um HMM.

Um modelo oculto de Markov geralmente é representado por uma tripla:

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

onde:

π é a distribuição inicial dos estados;

A é a matriz de probabilidade de transição entre os estados;

B é a matriz de probabilidades dos símbolos de observação.

Outras definições são necessárias para representar um HMM, como: N = número de estados, S = (S_1, S_2, ..., S_N) representando o conjunto de estados, M = número de observações possíveis, V = (V_1, V_2, ..., V_M) representando o conjunto de diferentes observações e q_t é um estado no instante t.

Cada elemento π_j é calculado através da probabilidade $P(q1 = Sj), 1 \leq j \leq N$.

Esses elementos representam a probabilidade do modelo iniciar no estado j. Os elementos da matriz A, ou seja, cada A_{ij} , são calculados pela probabilidade $P(qt = Sj \mid qt-1 = Si), 1 \leq i,j \leq N$. Esses elementos representam a probabilidade do próximo estado ser S_j sabendo que o estado anterior era S_i.

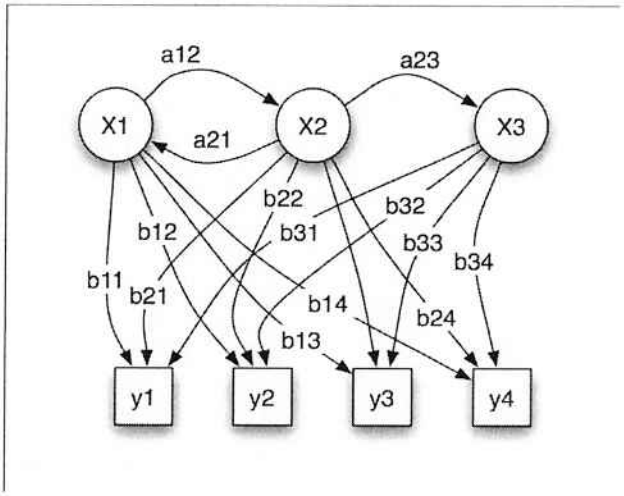


Figura 2-4 – Parâmetros probabilísticos do modelo HMM

x – estados

y – observações possíveis

a – possíveis transições de estado

b – probabilidades de saída

Cada elemento da matriz B, $B_{j(k)}$ é calculado pela probabilidade $P(O_t = V_k | q_t = S_j), 1 \leq j \leq N \leq k \leq M$ e O_t é a observação no tempo t. Esses elementos representam a probabilidade da observação ser igual a V_k e ser gerada pelo estado S_j.

2.4.5. Os três problemas básicos do HMM

Existem três problemas básicos que devem ser resolvidos para que modelo possa ser utilizado em aplicações do mundo real. Esses problemas são os seguintes:

i. Problema de avaliação:

Dada a seqüência de observação $O = (o_1, o_2, ..., o_T)$, e o modelo $\lambda = (A, B, \pi)$, como calcular eficientemente $P(O \mid \lambda)$, a probabilidade da seqüência de observações, dado o modelo?

ii. Problema da busca da melhor seqüência de estados:

Dado a seqüência de observações $O = (o_1 , o_2 , ..., o_T)$, e o modelo λ , como escolher uma seqüência de estados correspondente $Q = (q_1, q_2 , ..., q_T)$?

iii. Problema de treinamento:

Como ajustar os parâmetros do modelo $\lambda = (A, B, \pi)$ para maximizar $P(O \mid \lambda)$?

2.4.5.1. Solução do problema de avaliação

O primeiro problema é o problema de avaliação, isto é, dado um modelo e uma seqüência de observações, como calcular a probabilidade que a seqüência observada seja produzida pelo modelo? Este problema pode ser visto como um

dado modelo corresponde a uma dada seqüência de observações. Isso é extremamente útil.

Por exemplo, considerando o caso no qual se deve tentar escolher um modelo entre vários, a solução desse problema permite escolher o modelo que melhor corresponde às observações. A maneira mais direta de calcular a probabilidade de uma seqüência de observações $O = (o_1 , o_2 ,...,o_T)$, dado o modelo $\lambda = (A, B, \pi)$ é através da enumeração de todas as possíveis seqüências de estados de tamanho T (o número de observações), pela seguinte expressão:

$$\sum_{q_1 q_2 \cdots q_T} \pi_{q1} b_{q2}(o_1) a_{q1q2} b_{q2}(o_2) \ldots a_{q_{T-1}q_T} b_{q_T}(o_T)$$

2-5

A expressão acima envolve uma ordem de $2T \cdot N^T$ cálculos, para cada $t = 1, 2, ..., T$, existem N possíveis estados que podem ser alcançados, ou seja, existem N^T possíveis seqüências de estados, e para cada qual $2T$ cálculos são necessários (para ser preciso é necessário $(2T -1)N^T + N^T -1$). Este cálculo é computacionalmente inviável mesmo para pequenos valores de N e T, por exemplo, para $N = 5$ (estados), $T = 100$ (observações) existem na ordem de 10^{72} cálculos. Esse problema pode ser resolvido de maneira mais eficiente utilizando-se dos procedimentos *forward-backward.*

A variável *forward* $\alpha_t(i)$ é a probabilidade da seqüência de observações parciais $o_1,o_2 ,...,o_t$ (até o tempo t) e estado i no tempo t, dado o λ , onde:

$$\alpha_t(i) = P(o_1, o_1, \ldots, o_1, q_t = i|\lambda),$$

e $\alpha_t(i)$ pode ser resolvido recursivamente, utilizando as seguintes expressões:

Inicialização

$$\alpha_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N$$

2-6

Indução

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}), 1 \leq t \leq T-1 \; e \; 1 \leq j \leq N$$

2-7

Terminação

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

2-8

Para o mesmo exemplo citado acima, *N*=5 (estados), *T* = 100 (observações) são necessários 3.000 cálculos através do método *forward* , contra 1072 do cálculo direto. De maneira similar, a variável $\beta t\ (i)$ é definida como a probabilidade da seqüência de observações parciais de *t+1* para o final, dado estado *i* no tempo *t* e o modelo λ, onde:

$$\beta_T(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T | q_t = i, \lambda),$$

e $\beta t(i)$ pode ser resolvido recursivamente, utilizando as seguintes expressões:

Inicialização

$$\beta_T(i) = 1, 1 \leq i \leq N$$

2-9

Indução

$$\beta_t(j) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1}) \beta_{t+1}(j), t = T-1, T-2, \ldots, 1 e\ 1 \leq i \leq N$$

2-10

Terminação

$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1)\beta_1(i)$$

2-11

2.4.5.2. Problema da busca da melhor seqüência de estados

O problema 2 procura descobrir a parte escondida do modelo, ou seja, encontrar a seqüência de estados *correta*. Este problema geralmente é resolvido usando um procedimento próximo ao ótimo, o algoritmo de *Viterbi*, que procura a melhor seqüência de estados $Q = (q_1,q_2 ,...,q_T)$ para uma dada seqüência de observações $O = (o_1,o_2 ,...,o_T)$ e que será descrito posteriormente na seção 2.4.7.

2.4.5.3. Solução do problema de treinamento

O terceiro e mais difícil, é determinar um método para ajustar os parâmetros do modelo $\lambda = (A, B,\pi)$ para satisfazer um certo critério de otimização. A seqüência de observações utilizada para ajustar os parâmetros do modelo é chamada de seqüência de treinamento porque é utilizada para treinar o HMM. Não existe uma maneira conhecida de resolver analiticamente o conjunto de parâmetros do modelo que maximiza a probabilidade da seqüência de observações de uma maneira fechada. Entretanto, pode-se escolher $\varpi = (A, B,\pi)$ tal que sua probabilidade, $P(O \mid \lambda)$, é localmente maximizada usando um procedimento iterativo tal como o método de *Baum-Welch* (também conhecido como o método EM (*expectationmaximization*)) descrito na seção 2.4.6, ou técnicas de gradiente.

2.4.6. Algoritmo de Baum-Welch

Segundo Jelinek (Jelinek, 1997), o algoritmo de Baum-Welch estima os parâmetros do HMM ao permitir que os usuários recebam um texto preparado W, observando a saída do processador acústico e usando o HMM composto

correspondente a W como modelo do mecanismo de produção que resulta na saída observada A.

O algoritmo se inicia com o cálculo da variável de probabilidade $\xi(i,j)$, ou seja, a probabilidade de se estar em i no instante t e em j no instante t+1, dados os valores de O e λ. Utilizando os valores *backward* e *forward,* temos:

$$\xi(i,j) = \frac{P(q_t = i, q_{t+1} = j, O|\lambda)}{P(O|\lambda)} = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)}$$

2-12

Resultando em:

$$\xi(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}$$

2-13

Posteriormente, tem-se o cálculo da variável $\gamma_t(i)$ que representa a probabilidade de que se esteja no estado i no instante t. Utilizando as variáveis de *backward* e *forward* temos como expressão resultante:

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i) \beta_t(i)}$$

2-14

Portanto, há uma relação entre as probabilidades $\xi(i,j)$ e $\gamma_t(i)$, descrita por:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi(i,j)$$

2-15

Considerando as duas probabilidades em todos os valores discretos de t (1, 2,..., T), podemos encontrar um valor estimado para as transições do estado i para o

estado j, sendo $\sum_{t=1}^{T} \gamma_t(i)$ o número de transições que partem do estado i, e $\sum_{t=1}^{T} \xi(i,j)$ o número de transições de i para j.

Como conseqüência, podemos reestimar os parâmetros $\lambda = (\pi, A, B)$ do HMM, sendo:

$\pi_i =$ Frequência esperada (número de vezes) no estado (i)

$$a_{ij} = \frac{Número\ esperado\ de\ transições\ de\ i\ para\ j}{Número\ esperado\ de\ transições\ partindo\ de\ i}$$

$$b_j = \frac{Número\ esperado\ de\ vezes\ no\ estado\ j\ com\ observação\ v_k}{Número\ esperado\ de\ vezes\ no\ estado\ j}$$

Sendo v_k o k-ésimo símbolo no espaço amostral, ou seja, um elemento dentre as observações possíveis.

Observa-se que o método de Baum-Welch é iterativo e, portanto devem ser especificados critérios de parada.

Dentro do projeto do Roleta Russa, as cadeias ocultas de Markov são exigidas na fase de treinamento do reconhecedor, onde são definidos o número de estados de cada palavra utilizada como comando. As observações para cada estado são provenientes do processo de quantização vetorial e comparação com o *codebook*, sendo cada vetor referente a um cluster uma observação. Tendo valores iniciais estimados para as matrizes de probabilidade e observações, o algoritmo de Baum-Welch é rodado de forma a obter os melhores valores para estas mesmas matrizes. Na etapa de reconhecimento em tempo real, a amostra coletada e já quantizada passa pelo HMM de cada palavra, resultando em valores de probabilidade a serem analisadas pelo decodificador de Viterbi, descrito a seguir.

2.4.7. Algoritmo de Viterbi

Dada uma seqüência de observações $O = (o_1, o_2, ..., o_T)$ extraída de uma palavra desconhecida, o problema consiste em encontrar a classe (palavra) que

corresponde a melhor probabilidade de produzir a seqüência de observações $O = (o_1, o_2, ..., o_T)$. Para resolver esse problema, existem duas soluções: Modelo Discriminante e Modelo de Caminho Discriminante.

O primeiro, conhecido como Modelo Discriminante, consiste na construção de um modelo para cada classe ou palavra a fim de escolher a classe do modelo que leva a melhor probabilidade de produzir a seqüência de observações $O = (o_1, o_2, ..., o_T)$. A probabilidade pode ser calculada através dos algoritmos de *Viterbi* ou *Forward.* O segundo, conhecido como Modelo de Caminho Discriminante, consiste na construção de um único modelo no qual cada caminho corresponde a uma seqüência de letras. Então o algoritmo de *Viterbi* permite encontrar a seqüência de letras (palavra) que leva a melhor probabilidade de produzir a seqüência de observações $O = (o_1, o_2, ..., o_T)$.

Para encontrar a melhor seqüência de estados, Q = (q1 ,q2 ,...,qT) , para uma dada seqüência de observações O = (o1,o2 ,...,oT) define-se a quantidade:

$$\delta_{t+1}(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1 q_2 \ldots q_{t-1}, q_t = i, o_1, o_2 \ldots o_t \mid \lambda]$$

2-16

na qual, $\delta t\ (i)$ é o melhor resultado (probabilidade mais alta) ao longo de um caminho simples no tempo *t*, o qual leva em consideração as *t* primeiras observações e termina no estado *i*. Por indução tem-se:

$$\delta_{t+1}(i) = \left[\max_{t} \delta_t(i) a_{ij}\right] b_j(o_{t+1})$$

2-17

Para recuperar a seqüência de estados, é necessário manter os argumentos que maximizam a expressão anterior, para cada *t* e *i* através do *array* $\psi t\ (\ j)$. O procedimento completo para encontrar a melhor seqüência de estados é a seguinte:

Inicialização:

$$\delta_t(i) = \pi_i b_i(o_1), 1 \leq i \leq N \; \psi_1(\mathrm{i}) = 0$$

2-18

Recursão:

$$\delta_t(j) = \max_{1 \leq t \leq N} \left[\delta_{t-1}(i) a_{ij}\right] b_j(o_t), 2 \leq t \leq T \; e \; 1 \leq j \leq N$$

2-19

$$\psi_t(\mathrm{j}) = \arg \max_{1 \leq t \leq N} \left[\delta_{t-1}(\mathrm{i}) \mathrm{a}_{\mathrm{ij}}\right], 2 \leq t \leq T \; e \; 1 \leq j \leq N$$

2-20

Terminação:

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)]$$

2-21

$$q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)]$$

2-22

Caminho (*backtracking*):

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

2-23

Com exceção da etapa de *backtracking*, o algoritmo de *Viterbi* e o procedimento de *forward* têm basicamente a mesma implementação. A única diferença entre eles é que o somatório do procedimento *forward* é trocado pela maximização no algoritmo de Viterbi (Jelinek, 1997).

Como fase final do processo de reconhecimento, o algoritmo de Viterbi irá analisar os valores obtidos nos HMM e atribuir um valor de probabilidade relacionado ao caminho feito pelas amostras, ou seja, às transições ocorridas em cada modelo de Markov para a amostra. Desta forma, cabe ao reconhecedor apenas verificar qual valor de probabilidade foi o maior dentre os obtidos para cada palavra. Este valor é referente á palavra reconhecida pelo sistema.

2.5. Considerações Finais

Os conceitos teóricos apresentados neste capítulo serviram de base para a construção do reconhecedor de fala. A partir destas informações colhidas foram feitas pesquisas em busca de bibliotecas e funções próprias para os cálculos matemáticos e algoritmos para cada etapa da construção do sistema.

Para o grupo, esta etapa de estudos representou a aquisição de uma série de conceitos e definições não vistos durante o curso e ainda, no caso do processamento inicial dos sinais de áudio, da análise cepstral e da quantização vetorial, relacionados à parte de sistemas eletrônicos. O estudo sobre as cadeias ocultas de Markov e sobre o Decodificador de Viterbi serviu para aumentar o embasamento na área de processos estocásticos.

3. Descrição do Roleta Russa

O objetivo deste capítulo é apresentar a descrição do funcionamento do sistema, o jogo nomeado Roleta Russa, bem como identificar de forma completa e clara todos os requisitos referentes tanto à parte de software quanto à configuração de hardware.

Os itens citados foram levantados no início do projeto, acarretando a geração de um documento de especificação de requisitos e de um relatório de estudos referentes à parte teórica compreendida pelo reconhecimento de fala.

3.1. Especificação de Requisitos

Perspectivas do Produto

O sistema está instalado no aparelho celular e utiliza o equipamento de som automotivo; suas interfaces são descritas a seguir:

- Interface Homem-Computador: Correspondente à comunicação do aparelho celular com o usuário, sendo que todos os comandos são acionados por voz.
- Interface entre software e aparelho celular: Permite o uso pelo software dos dados vocais captados por intermédio do microfone do aparelho.
- Interface entre o aparelho celular e o som automotivo: Permite a transmissão das mensagens de voz do sistema para o som automotivo via protocolo *Bluetooth*.

O diagrama de blocos da figura 3.1 mostra as interações do sistema com o ambiente, identificando as interfaces descritas anteriormente. As setas brancas correspondem à interação entre aparelho celular e som automotivo, a qual não é essencial para o funcionamento do sistema, ou seja, trata-se de uma funcionalidade opcional para o usuário:

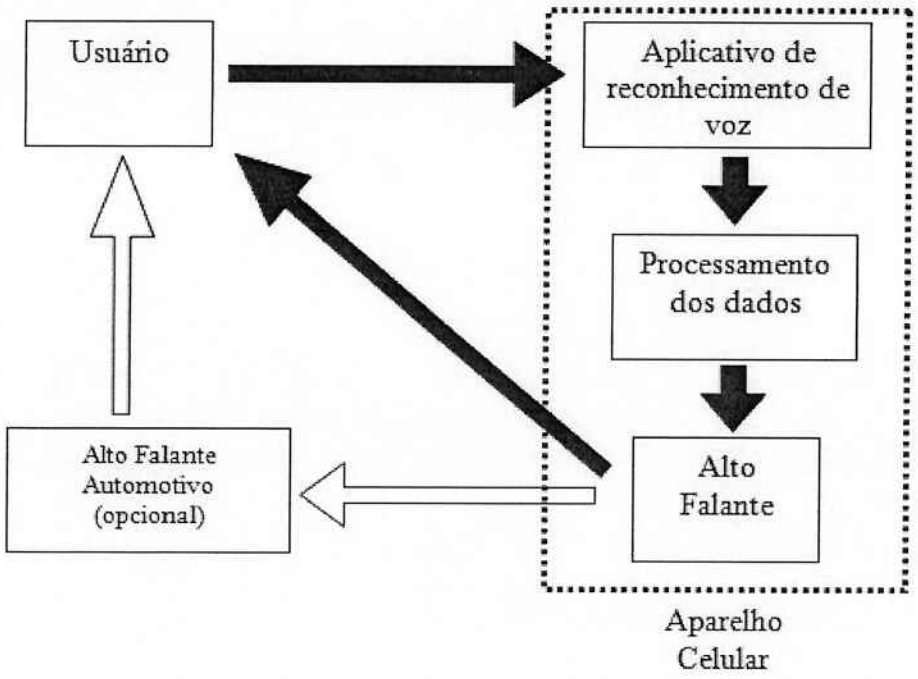


Figura 3-1 – Interações relacionadas ao sistema

3.2. Interface de Usuário

A interface homem-computador é realizada através da audição e da fala, não sendo exigida uma interface visual para o usuário. Este se comunica com o sistema exclusivamente por comandos de voz, definidos e descritos a seguir:

- "UM" → Escolha pela alternativa número 1;
- "DOIS" → Escolha pela alternativa número 2;
- "TRÊS" → Escolha pela alternativa número 3;
- "PULAR" → Usuário pula a questão, não sendo computados pontos;
- "SAIR" → Usuário termina a partida, ou seja, sai do sistema;
- "SIM" → Confirmação de comando;
- "NÃO" → Cancelamento de comando;

Para que o usuário possa ser guiado pelo sistema, haverá uma base de dados com mensagens de voz em ordem definida (com exceção das mensagens relacionadas às questões da partida). Inicialmente o sistema apresenta uma mensagem de boas vindas ao usuário e a partir deste ponto interage através de perguntas, exigindo um comando do usuário para o prosseguimento do jogo.

3.3. Interfaces de Hardware

A comunicação entre hardware e software é realizada no momento em que é solicitada uma resposta do usuário.

O sistema deve solicitar ao aparelho a abertura do microfone para captação de voz durante um determinado período de tempo. Depois de captado o comando, é realizada a digitalização do sinal vocal e fechada a comunicação com o aparelho pelo software, iniciando o processamento do comando. O sistema decodifica o comando e compara com as amostras de referência na base de dados.

3.4. Interface de Comunicação

Para realizar a comunicação com o som automotivo, é utilizado o *Bluetooth*. O Protocolo *Bluetooth* é um padrão de comunicação sem fio e de baixo consumo de energia, o qual realiza a transmissão de dados entre dispositivos. Para isso, uma combinação de hardware e software é utilizada para permitir que essa comunicação ocorra entre os mais diferentes tipos de aparelhos. A transmissão de dados é feita através de radiofreqüência, permitindo que um dispositivo detecte o outro independente de suas posições, desde que estejam dentro do limite de proximidade.

Um Perfil Bluetooth é uma especificação de interface para comunicação entre dispositivos baseadas em Bluetooth.

Os perfis definem padrões que os fabricantes devem seguir para permitir que seus dispositivos utilizem *Bluetooth* de uma maneira compatível. Para realizar essa tarefa, cada perfil usufrui de opções particulares e parâmetros em cada nível do protocolo, ou seja, utilizam configurações específicas dependendo da finalidade do perfil, como troca de dados, tratamento de áudio ou vídeo.

O som automotivo Sony Xplod MEX-BT2507X foi escolhido, pois, além de ser compatível com *Bluetooth*, possui também interface para A2DP. O A2DP (*Advanced Audio Distribution Profile*) é um perfil de controle de mídia do protocolo Bluetooth, o qual define os padrões de transmissão entre dois dispositivos distintos, como por exemplo, a troca de informações entre um telefone celular e o rádio automotivo (A2DP Specification).

Segundo sua especificação retirada do manual do produto (http://www.docs.sony.com/release/MEXBT2507X_PT.pdf), o equipamento permite:

"Conectividade com telefone celular Bluetooth, recursos hands free com microfone embutido, transferência de agenda telefônica e áudio streaming. A tecnologia Bluetooth permite uma distância de até 10 metros (dependendo do ambiente) na comunicação de dados entre dois aparelhos digitais compatíveis."

Serve, assim, como um reprodutor de áudio e transmissor de áudio por um microfone.

3.5. Funções do Software

O sistema desempenha uma única função relacionada ao jogo em si. Dentre os procedimentos utilizados no sistema, tem-se:

- Criação de uma nova partida, seja com regras padronizadas ou previamente configurada pelo usuário;
- Durante a configuração de uma partida, o sistema orienta o usuário, perguntando pelas opções desejadas para o jogo. Depois de terminada a configuração, o sistema inicia a partida;
- Durante uma partida, o sistema enuncia a pergunta e dá três alternativas ao usuário. Além das alternativas, o usuário tem a opção de pular a pergunta ou até mesmo de cancelar a partida;
- Após o término das perguntas, o sistema computa a pontuação final do usuário, informando-o. Haverá as opções de reinício de partida ou saída, retornando o sistema ao estado inicial.

3.6. Características dos Usuários

Há restrições de uso do sistema por portadores de deficiência auditiva e de fala, já que o software funciona por meio de comandos de voz. Fora destas situações,

espera-se o desenvolvimento de um software voltado a pessoas de todas as idades e perfis.

3.7. Versões Futuras

Alguns melhoramentos podem ser realizados, acarretando em novas versões do software, de modo a otimizar o desempenho e as funcionalidades do software.

- Aumento na base de dados de amostras de referência

O aumento da quantidade de amostras na base de dados de referências de voz, levando em conta outros ambientes diversos e outros tons de voz, aumenta a probabilidade de acerto, pelo sistema, na identificação de respostas dadas pelo usuário, melhorando seu desempenho.

- Criação de novas opções e funcionalidades

A implementação de novas opções e funcionalidades no sistema traz a necessidade da criação de novas amostras para identificação de voz, além de um aperfeiçoamento nas rotinas de reconhecimento de fala. Com a adição destes incrementos, o usuário pode ter maior flexibilidade com relação às suas opções, o que pode acarretar em maior aprovação do aplicativo pelo mercado.

- Aumento na base de perguntas

A inserção de novas questões faz com que o jogo se torne menos repetitivo, permitindo inclusive a criação de novos níveis de dificuldade, melhorando os níveis de jogabilidade e diversão do jogo.

- Adaptação a outras plataformas e sistemas operacionais

Uma expansão possível na implementação é a criação de versões voltadas a outros modelos de aparelhos celulares e outros sistemas operacionais. Assim, alcança-se uma fatia maior do mercado, possibilitando, inclusive, a criação de outros aplicativos desta natureza para estes sistemas.

3.8. Requisitos de Banco de Dados

A base de dados para este projeto consiste de uma base convencional - contendo informações relacionadas à configuração de partidas, às alternativas, às perguntas – e de uma base de arquivos de áudio para referência de vozes e para mensagens de voz relacionadas aos enunciados, armazenada em um diretório de conhecimento do sistema. O caminho para estes diretórios é apontado no próprio banco do sistema e acionado em uma ordem definida na implementação do software.

Como a base de dados permanece fixa, não há a ocorrência de conflitos procedentes da utilização por vários usuários, assim como não há também a existência de métodos de criação, edição e remoção de dados no banco, o que garante a integridade dos dados e, conseqüentemente, a confiabilidade do sistema, já que as falhas relacionadas à persistência no banco são diminutas.

O acesso do sistema ao banco também é restrito já que as perguntas e configurações de partida são requeridas uma única vez no início de cada partida.

3.9. Descrição dos Requisitos de Hardware

Os requisitos de hardware, necessário para este sistema estão descritos a seguir.

- Placas de Som

  Como a voz requer uma largura de banda relativamente baixa, praticamente todas as placas de 16 bits de qualidade média podem ser usadas para o reconhecimento de fala.

- Microfones

  Um microfone de qualidade é a chave ao utilizar RF. Na maioria dos casos, um microfone regular não atenderá as necessidades. Eles tendem a captar muito

ruído do ambiente o que aumenta o tempo de processamento dos programas de RF.

- Processadores

As aplicações de RF podem depender muito da velocidade de processamento, pois pode ocorrer uma grande quantidade de filtragens digitais e de processamento de sinal pode ocorrer durante o reconhecimento.

Assim como qualquer software de uso intensivo de CPU, quanto mais rápido for o processador e maior a quantidade de memória melhor serão seus resultados. É possível fazer reconhecimento vocal com memória RAM de 100MHZ e 16MB como requisitos mínimos.

3.10. Restrições de Projeto

- Aspectos Legais e Normativos

É importante salientar que problemas poderiam ser enumerados levando em conta aspectos normativos. A utilização de um jogo em pleno trânsito poderia trazer problemas com relação à distração do motorista.

O argumento utilizado para a utilização legal do produto é o fato de este interagir com o usuário dentro do automóvel da mesma forma que um rádio automotivo normal. Em nenhum momento o motorista utiliza os sentidos do tato e da visão para manipular o sistema. Uma recomendação a ser feita na especificação do projeto é a não utilização do sistema em auto-estradas e vias de grande velocidade de rolamento. O produto é voltado para utilização em situações de tráfego intenso, porém pode ser utilizado em ambientes variados, onde a atenção do usuário não é requerida.

- Restrições Relativas à Capacidade de Hardware

A instalação do sistema deve ser feita de acordo com os requisitos de hardware especificados neste documento, o que acaba por restringir o número de usuários, inclusive por fatores aquisitivos.

Deve ser considerado o grande volume de dados armazenados (mesmo em versões futuras do sistema) somado aos índices de desempenho e usabilidade que requerem um maior poder de processamento do hardware. Assim, consolida-se esta restrição a partir dos requisitos levantados para a aceitação final do sistema.

- Operações Paralelas

É imprescindível que o software possa ser interrompido durante sua execução para que seja priorizada outra tarefa do aparelho celular. O recebimento de ligações durante uma partida deve ser considerado, levando o sistema a um estado de interrupção (pausa), assim como o retorno desta interrupção num estado específico, como, por exemplo, a repetição do enunciado da pergunta.

Outro ponto importante está relacionado com o fato de o usuário poder finalizar a execução do software através de botão no teclado do aparelho celular. Comandos em interface convencional, i.e., pela tela do celular, devem também ser criados, porém não podem ser essenciais para a execução de uma funcionalidade do software.

- Tempo de Resposta do Usuário

O sistema não deve permanecer em aguardo de um sinal mais forte de voz do usuário para continuar em execução. Haverá uma temporização para que o usuário possa responder ou pular a pergunta. Se o microfone permanecesse aberto, podem ser geradas falhas no caso de recebimento de uma ligação, já que o aparelho pode ser considerado em estado ocupado. Além disso, o reconhecimento de fala pode ser prejudicado, podendo interferir nos níveis de confiabilidade e usabilidade do sistema.

- Inflexibilidade das Opções de Jogo

Os comandos de jogo são limitados, já que a criação de diversos comandos de voz acarreta em maior complexidade nos algoritmos de reconhecimento, interferindo no desempenho do software. A ocorrência do não-reconhecimento de

comandos do usuário devido à intromissão de ruídos externos e concorrentes seria maior, o que exigiria um remodelamento das gravações em cada ambiente.

3.11. Requisitos Não Funcionais

Os requisitos não funcionais considerados neste sistema são os seguintes:

- Usabilidade: O sistema deve ter alto nível de usabilidade, o usuário deve compreender os enunciados e os comandos que deve fazer com clareza, e o sistema deve ter poucos erros na identificação do comando dado pelo usuário; quanto mais erros no entendimento e reconhecimento, mais repetições de comando são exigidos, piorando o nível de usabilidade.
- Confiabilidade: Neste contexto, a confiabilidade do sistema está associada à freqüência de ocorrência de falhas decorrentes principalmente na identificação dos comandos dados pelo usuário. O sistema deve possuir uma taxa de acerto na identificação de no mínimo 75% dos casos.
- Desempenho: O tempo de resposta para a interação entre aparelho e usuário deve ser adequado de forma a não atrapalhar o decorrer do jogo, espera-se até no máximo três segundos para obtenção de uma resposta. Tempos de processamento muito longos fazem com que o usuário entedie, diminuindo a aprovação do produto.
- Jogabilidade: Este requisito, relacionado ao jargão técnico utilizado no desenvolvimento de jogos e cuja inclusão foi definida pelo grupo, leva em conta a aprovação dos usuários quanto à dinâmica e à inteligência das perguntas, ou seja, o nível de diversão do usuário ao utilizar o sistema.
- Especificação das Amostras de Voz: Para que o reconhecimento de fala seja eficiente, as amostras de voz que servirão como referências para comparação com o comando do usuário devem ser levantadas seguindo o procedimento desta seção. São especificados os possíveis ambientes de utilização do sistema, assim como os tons de voz possíveis considerando a diversidade de usuários, criando um leque maior para reconhecimento do comando de voz, aumentando usabilidade e confiabilidade.

- Especificação dos Ambientes: A tabela a seguir especifica os ambientes para gravação das amostras de referência, levando em consideração o escopo deste projeto, assim como outros possíveis locais de utilização do sistema:

Tabela 3-1 – Especificação de Ambientes

Ambiente externo com baixo nível de ruído	Locais abertos, com pouca circulação de pessoas e veículos, caracterizando um ambiente silencioso. Ex: parques, praças.
Ambiente externo com alto nível de ruído	Locais abertos, com alta incidência de ruídos oriundos de grande movimentação de pessoas e veículos, além de outros fatores de poluição sonora. Ex: ruas, avenidas.
Ambiente interno com baixo nível de ruído	Local fechado com baixo ou nenhum ruído, caracterizando um ambiente silencioso.
Ambiente interno com alto nível de ruído	Local fechado, porém com incidência de ruídos associados principalmente a aparelhos eletro-eletrônicos e interferências vocais.
Interior de veículo com vidros fechados e somente com o motorista.	Situação em que se pretende simular a condução de um veículo com vidros fechados e sem a interferência de outros sinais vocais, existindo apenas o som do motor do veículo.
Interior de veículo com vidros fechados e com a existência de passageiros.	Situação em que se pretende simular a condução de um veículo com vidros fechados, porém com a incidência de outros possíveis sinais vocais.
Interior de veículo com vidros abertos e somente com o motorista.	Situação em que se pretende simular a condução de um veículo com interferência de ruídos externos e sem a interferência de outros sinais vocais internos ao veículo.
Interior de veículo com vidros abertos e com a existência de passageiros.	Situação onde se pretende simular a condução de um veículo com interferência de ruídos externos além da interferência de outros sinais vocais internos ao veículo, caracterizando um ambiente de alto nível de ruídos.

- Especificação das Amostras: Para cada ambiente especificado deverá ser obtida uma bateria de amostras considerando tons de vozes diversos. A tabela a seguir indica os tons de voz e a quantidade de amostras consideradas para cada tom de voz:

Tabela 3-2 – Especificação das amostras

Voz Masculina	10 amostras	Amostras de tons de voz masculina, considerando a diversidade dos tons de voz nesta categoria.
Voz Feminina	10 amostras	Amostras de tons de voz feminina, considerando a diversidade dos tons de voz nesta categoria.
Voz de Crianças	10 amostras	Amostras de tons de voz relativas a crianças, considerando a diversidade dos tons de voz e idade nesta categoria.

3.12. Reconhecimento de Fala

- Método para Reconhecimento de fala

O método utilizado para o processamento e comparação de voz é o Decodificador de Viterbi, utilizando o modelo oculto de Markov. Uma descrição deste método encontra-se no Capítulo 2.

- Ferramentas Utilizadas

Dentre as ferramentas utilizadas para estudo, simulação e implementação do módulo de reconhecimento de fala, a mais importante é o Scilab, software voltado para levantamento de curvas e funções matemáticas. É utilizada a biblioteca HMM, voltada especificamente para a simulação de cadeias ocultas de Markov.

Uma ferramenta importante para o desenvolvimento e testes é o próprio Visual Studio, que, além de possuir uma ferramenta para desenvolvimento em aparelhos móveis, possibilita o uso de uma série de bibliotecas específicas para processamento de sinais vocais.

3.13. Critérios de Aceitação

A aceitação do projeto é dada pelo produto como um todo, ou seja, é adotado um critério de aceitação total do sistema.

O sistema para ser aceito deve realizar as seguintes funções:

- Reconhecimento de um comando a partir da comparação com amostras na base de dados.
- Reconhecimento dos comandos de navegação pelo sistema, i.e., a seqüência das funcionalidades e casos de uso do sistema.
- Confiabilidade, usabilidade e desempenho do sistema.

3.14. Definição do Funcionamento do Jogo

O modelo do software de entretenimento é de perguntas e respostas, com escolha entre alternativas, totalmente controlado por comandos de voz, assim como descrito no exemplo a seguir:

Pergunta: "Qual a capital da Ucrânia?".

Alternativas:

1) Minsk.

2) Kiev.

3) Moscou.

Resposta do usuário: "Um".

Correção pelo aplicativo: "Errado. A resposta certa é dois, Kiev".

3.15. Modelagem

O processo de modelagem do software consiste na confecção de artefatos relacionados à análise de casos de uso, modelo de classes, sequências, estados, além da descrição da interface por áudio e voz. Os produtos decorrentes destas análises encontram-se na seção de apêndices (Apêndices A, B, C, D, E) desta obra.

3.16. Considerações Finais

Neste capítulo foi apresentado um resumo da especificação de requisitos do sistema, feita na fase inicial do projeto, o que inclui interfaces, funções, requisitos funcionais e não-funcionais, casos de uso para descrever e definir a estratégia do jogo e também modelos de classes, interação e estados para especificar o software sem entrar na questão de como é feita a parte de reconhecimento de fala, visto que a teoria por trás do reconhecimento de fala tem um capítulo próprio (o Capítulo 2) bem como o desenvolvimento do reconhecedor (Capítulo 4).

O grupo optou por resumir a especificação feita inicialmente em vez de anexar o documento criado à monografia com o intuito de destacar os pontos mais importantes desta fase.

O termo jogabilidade não representa um requisito não-funcional padrão, porém é importante na avaliação do jogo. Este requisito representa a atratividade do jogo, além dos níveis de entretenimento e interatividade por parte do usuário.

Esta fase foi extremamente importante, pois resultou em um artefato para uso durante todo o projeto, descrevendo não só o escopo de projeto como o funcionamento detalhado do sistema, de forma que não houvesse desvios com relação aos requisitos do sistema durante a fase de implementação.

4. Desenvolvimento do Reconhecedor de Fala

O reconhecimento de fala é o processo pelo qual um computador (ou o outro tipo de máquina) identifica palavras faladas. Basicamente, significa falar para seu computador e este reconhecer corretamente o que está sendo dito.

As seguintes seções mostram os princípios utilizados para construção de um reconhecedor de fala.

4.1. Método de Reconhecimento de Fala

O método de reconhecimento utilizado foi o de expressões isoladas. Este método convencional de reconhecimento de palavras é constituído dos processos de pré-ênfase do sinal, conversão à escala MEL das transformadas de Fourier, quantização vetorial, aplicação do modelo oculto de Markov (HMM) e decodificação de Viterbi. Estes itens estão descritos no capítulo 2.

4.2. Reconhecedor de fala

Nesta seção é apresentada como foi realizado o desenvolvimento do reconhecedor e premissas teóricas adotadas. São abordados os passos do processo de reconhecimento de fala, desde a captação do sinal no microfone até a saída da palavra reconhecida.

4.2.1. Obtenção das Amostras

Como primeiro passo, foram levantadas as amostras de voz que serviram como referências para comparação com o comando do usuário em um ambientes livre de ruído e interferências. Foram utilizados diversos tons de voz, considerando a diversidade de usuários, criando assim um leque maior de amostras para aumentar a probabilidade de sucesso no reconhecimento

Para o ambiente especificado foi colhida uma bateria de amostras, cuja característica está descrita na Tabela 4-1.

Tabela 4-1 – Amostras Colhidas

Tom de Voz	*Amostras Necessárias*	*Amostras Colhidas*
Voz Masculina	10 amostras	Foram colhidas quatorze amostras de tons de voz masculina, considerando a diversidade dos tons de voz nesta categoria.
Voz Feminina	10 amostras	Foram colhidas treze amostras de tons de voz feminina, considerando a diversidade dos tons de voz nesta categoria.
Voz de Crianças	10 amostras	Não foram colhidas amostras de crianças.

4.2.2. Armazenamento de Amostras

As amostras foram armazenadas em arquivos WAV (*WAVEform audio format*). O WAV é o formato padrão da Microsoft e IBM para armazenamento de áudio em PC's. É uma variação do método de formatação de fluxo de bits RIFF para armazenar dados em blocos (*chunks*) e também parecido com os formatos IFF e AIFF usados em computadores Macintosh. Ambos, WAV e AIFF, são compatíveis com os sistemas operacionais Windows e Macintosh.

Apesar de um arquivo WAV poder conter áudio compactado, o formato mais comum de WAV assim como o utilizado pelo sistema, contém áudio em formato de modulação de pulsos PCM (pulse-code modulation), que usa um método de armazenamento de áudio não-comprimido (sem perda). O formato WAV foi utilizado para garantir a qualidade máxima de áudio, além de poder ser editado e manipulado com relativa facilidade usando software.

Por ser um formato sem compressão, o WAV ocupa um espaço muito grande de armazenamento, mas por outro lado poupa ciclos de processamento do dispositivo móvel por não necessitar de decodificação.

4.2.3. Processamento de Fala

Nesta seção é descrito o processo utilizado para reconhecimento de fala neste trabalho. De acordo com o processo escolhido, pôde-se fragmentar o reconhecedor em duas fases distintas: treinamento e reconhecimento.

Treinamento

A fase de treinamento do reconhecedor consiste na criação de um cenário onde estão inseridas as amostras colhidas, para que sirvam como referência para o reconhecimento das palavras. Ao fim desta etapa, tem-se como produto uma matriz de referência para a captação de voz, chamada *codebook* e o modelo oculto de Markov (*Hidden Markov Model* – HMM), cujas definições e propriedades são apresentadas no capítulo 2.

Reconhecimento

O reconhecimento propriamente dito consiste na captação do sinal de voz do usuário em tempo real, na criação de um vetor quantizado para a palavra e na comparação com os modelos de Markov obtidos na fase de treinamento por intermédio do Decodificador de Viterbi. Deste modo, o sistema obtém a palavra reconhecida.

Um esquema representando este processo é ilustrado na Figura 4.1 feita pelo grupo. Pode-se observar que, em ambas as etapas citadas anteriormente, há um pré-processamento do sinal. Este pré-processamento modifica o sinal obtido tanto das amostras quanto do sinal obtido em tempo real, de forma a ampliar as diferenças no timbre e nas entonações do sinal, facilitando o processo de reconhecimento.

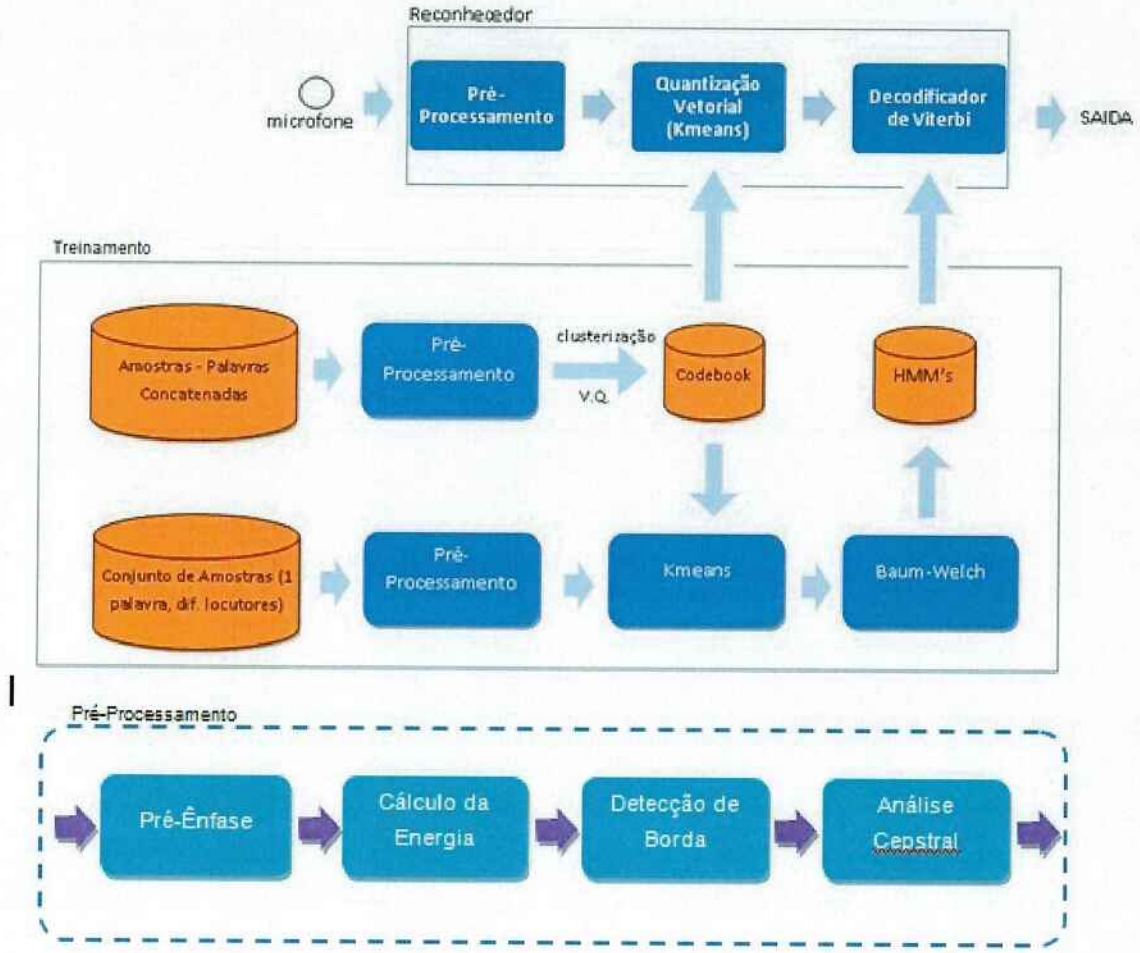


Figura 4-1 – Processo de Reconhecimento de Fala

A principal ferramenta utilizada para o processamento das amostras e geração das matrizes referentes ao *codebook* e ao HMM foi o software Scilab que, devido à sua voltada à operações e funções matemáticas, permitiu a visualização e a análise das saídas de cada etapa do processo, além de fornecer suporte a todo o processo de transformação dos sinais de áudio em valores discretos para criação dos perfis espectrais.

As atividades do processo de reconhecimento de fala são descritas com maior detalhamento nas seções seguintes.

4.2.3.1. Treinamento

Utilizando as amostras de voz recolhidas, foi iniciada a criação do processo de treinamento que pode ser dividida em duas partes:

- Obtenção do *codebook* através da concatenação de todas as amostras colhidas
- Obtenção das matrizes referentes ao HMM, através do algoritmo de Baum-Welch.

Para o a obtenção do codebook foram realizadas as seguintes etapas: pré processamento e clusterização e quantização vetorial.

Pré-processamento

O primeiro passo após ter as gravações das amostras foi concatená-las, de forma a gerar um vetor representando um sinal de áudio único. Para isso, cada amostra colhida em arquivos .wav (WAVE) foi aberta no Scilab e foi submetida a uma bateria de processamentos, definida como pré-processamento, a qual consistiu da pré-ênfase (descrita no capítulo 2), do cálculo da energia do sinal em decibéis (dB) e da detecção das bordas que definem o início e o fim de cada palavra falada, eliminando fragmentos no sinal constituídos apenas de ruído.

Após o pré-processamento, os valores de amostragem dos sinais processados foram então armazenados em um vetor, de forma a concatenar todas as amostras como visto na Figura 4.2.

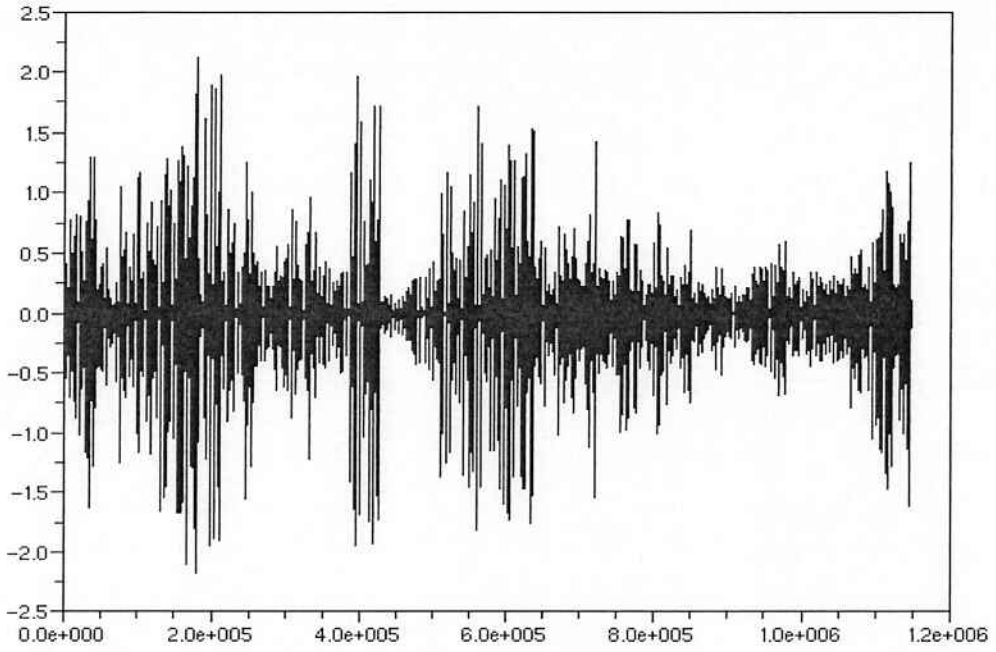


Figura 4-2 – Gráfico ilustrando o sinal resultante da concatenação das amostras (Scilab).

Clusterização e Quantização Vetorial

Para a etapa de clusterização das amostras foram necessárias funções não existentes no Scilab. Porém estas funções foram encontradas em um site relacionado a reconhecimento de fala (speech-recognition.de), implementadas nas funções de enquadramento e cálculo dos cepstros para a plataforma Matlab. Foi, então, necessário fazer uma conversão do código implementado em Matlab para a linguagem aceita pelo Scilab, atividade que não consumiu grande parcela de tempo devido à similaridade na sintaxe das linguagens apontadas.

Tais funções foram melFilterMatrix e computeMelSpectrum, cuja descrição é apresentada a seguir:

- melFilterMatrix:

Função responsável pela criação das matrizes relacionadas ao filtro de quadros triangulares para conversão dos sinais em cepstros, ilustrada na Figura 4.3. Esta função possui três parâmetros de entrada: a freqüência de

amostragem dos sinais de áudio, o tamanho de cada quadro, ou seja, quantas amostras de sinal serão compreendidas pelo filtro e o número de quadros criados para a filtragem. Usualmente este valor corresponde a 20 quadros. Esta função utiliza outras duas funções incluídas no pacote convertido para a linguagem da plataforma Scilab: a mel2freq e a freq2mel, ambas responsáveis pela conversão das freqüências entre a escala convencional e a escala mel.

- computeMelSpectrum:

Função que realiza a criação dos cepstros e a alocação de valores em duas matrizes representando os valores de freqüência e energia de cada cepstro. Inicialmente é utilizada a função computeSpectrum, responsável pelo cálculo de espectros do sinal de áudio por intermédio da transformada de Fourier (FFT). Posteriormente, os valores são convertidos para a escala mel. Como parâmetros de entrada, a função computeMelSpectrum aceita: o filtro de janelas triangulares criada com a função melFilterMatrix, e o valor de *winShift*, ou seja, a distância entre o início de uma janela e outra, relacionada inclusive com o tamanho da janela definida pelo programador. No caso deste projeto, o valor de *winShift* é a metade do tamanho da janela; o vetor de amostras concatenadas. Este valor foi obtido experimentalmente como o que melhor otimizava a produção de cepstros.

Como produto desta análise cepstral, obtém as matrizes dos cepstros na escala Mel e dos valores de energia, estes valores são clusterizados e repassados a etapa de quantização vetorial.

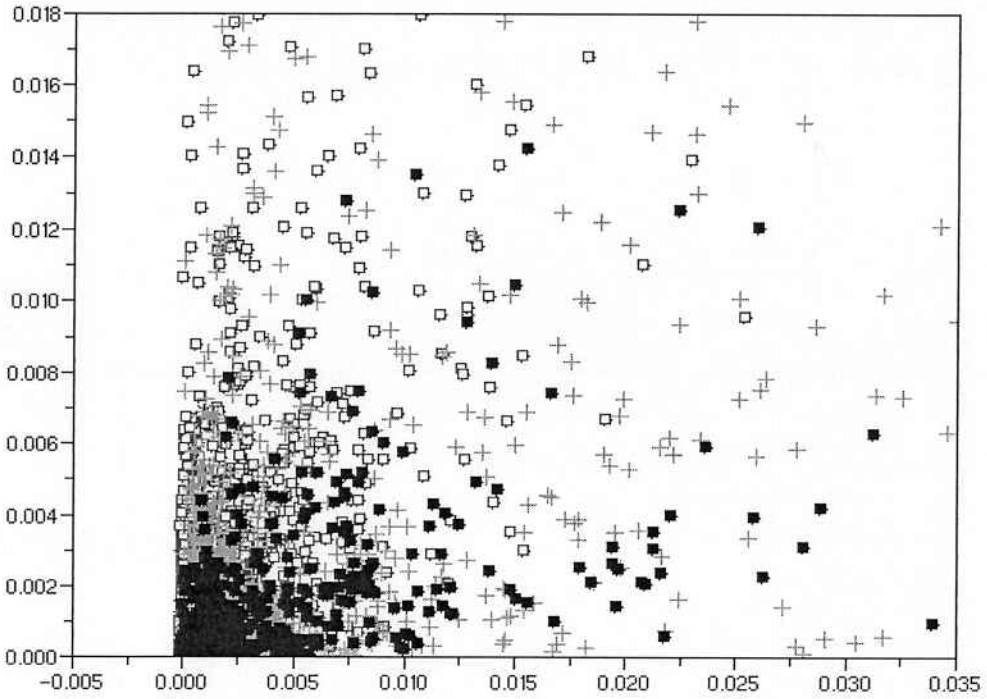


Figura 4-3 – Disposição dos valores calculados dos cepstros em comparação com cada quadro do filtro triangular (Scilab).

Os cálculos para clusterização e quantização vetorial foram feitos por outro pacote disponível no próprio site do Scilab (scilab.org), que contém uma biblioteca de funções relacionadas ao cálculo das matrizes do HMM. Entre as funções incluídas no pacote, encontra-se a função k-means, responsável pela clusterização dos cepstros em MEL e criação do *codebook* com os perfis calculados considerando os valores de cada cepstro. A função utiliza a matriz relacionada aos cepstros e analisa os valores separando-os em clusters, assim como visto no capítulo 2. O número de clusters é definido pelo programador e é um parâmetro de entrada de k-means. O valor utilizado no projeto foi de 64 clusters. Este valor foi definido devido ao número de amostras utilizadas, um valor menor afetaria a precisão do reconhecedor.

Como resultado tem-se o *codebook*, com as médias calculadas para cada cluster criado.

Partindo então para a obtenção das matrizes referentes ao HMM, foram realizadas as seguintes etapas: Pré-Processamento, Clusterização e Quantização Vetorial e Algoritmos de Baum-Welch e HMM.

Pré-Processamento

Realizada da mesma forma como na obtenção do codebook, porém, a concatenação feita é das locuções, ou seja, são agrupadas todas as amostras de uma única palavra determinada para ser reconhecida. Inclui também a etapa de pré-ênfase e detecção de borda.

Clusterização e Quantização Vetorial.

As etapas de análise cepstral também são idênticas as realizadas para confecção do *codebook*, a diferença está na quantização vetorial (k-means) que deve ser feita utilizando como referência o *codebook* calculado na etapa anterior. O pacote de HMM para o Scilab contém uma função denominada *nearest,* responsável pela análise dos cepstros calculados para a palavra correspondente e identificando, no *codebook*, qual cluster se adequa melhor ao resultado da análise feita. A função aceita como entrada o *codebook* e uma linha da matriz dos cepstros calculados na análise cepstral Como saída, *nearest* retorna um vetor com os valores do cluster mais próximo da linha de entrada, o valor de índice deste entre todos os clusters e a distância mínima entre os centróides do cluster calculado e do cluster selecionado no codebook.

Algoritmos de Baum-Welch e HMM

Para cada palavra reconhecida pelo sistema foi feito um modelo oculto de Markov (HMM), considerando um número de estados para cada palavra. Desta forma, foram escolhidos 5 estados para cada uma das palavras menores ("UM", "SIM", "NÃO") e 8 estados para cada uma das palavras com mais fonemas ("DOIS", "TRÊS", "SAIR", "PULAR").

O projeto do reconhecedor utiliza um modelo de Markov (HMM) denominado *left-to-right*, cujo processamento não possui retorno de estado, ou seja, partindo do estado inicial q_0, sendo N estados e k = 0,1,...,N, ao ser feita a transição de um estado q_k para o estado q_{k+1}, não há a possibilidade de retorno ao estado q_k.

Desta forma, tendo em mãos o pacote para o Scilab para a construção do HMM, foi utilizada a função *dlrrest,* voltada para a criação de um modelo left-to-right, que estima, através de um processo iterativo, os valores das matrizes A e B, a partir de valores iniciais correspondentes às próprias matrizes A e B definidas pelo programador. Há também a denominação de um vetor de estados no momento inicial π_0, que no modelo *left-to-right* sempre é compreendido pelo valor '1' no estado inicial e '0' nos demais estados.

A estimativa dos valores de A e B iniciais para cada palavra foi feita através de um algoritmo que calcula valores distintos relacionados ao número de colunas, já que a soma dos valores de cada coluna em uma linha deve ser igual a 1, pois representa todas as probabilidades de troca de estados, somando 100%.

A função *dlrrest* aceita como entrada os valores estimados iniciais de π_0, A e B, um vetor de Observações externas O, compreendido pelos valores da função *nearest* para a correspondente palavra, e dois valores de tolerância para término das iterações: o número máximo de iterações e a diferença mínima entre dois valores obtidos de passos subseqüentes.

Como produto de *dlrrest* obtém o modelo HMM para uma palavra, com os valores calculados de π_0, A, B, além de um vetor de probabilidades calculadas, próprias do HMM, que são utilizados posteriormente no decodificador de Viterbi.

4.2.3.2. Reconhecimento

Implementado no executável a ser rodado no aparelho móvel, a etapa de reconhecimento utiliza como entrada o sinal de áudio oriundo do microfone do aparelho, realiza os processos de sinal descritos abaixo e obtém o resultado final através do decodificador de Viterbi.

Pré-processamento

Utiliza o mesmo processo realizado na parte de treinamento do reconhecedor, com as etapas de pré-ênfase e detecção de borda.

Clusterização e Quantização Vetorial

De forma semelhante à realizada no processo de construção do HMM, o processo de clusterização utiliza a função *nearest* com o codebook obtido no treinamento como referência, de forma a identificar o cluster mais próximo do perfil montado da amostra de voz.

Decodificador de Viterbi

Tendo os modelos ocultos de Markov para todas as palavras reconhecidas pelo sistema, a última etapa do processo de reconhecimento consiste na chamada do decodificador de Viterbi para cada HMM, passando como entrada o modelo de HMM consistido do vetor π_0 e das matrizes A e B, além do vetor de observação da palavra captada. Como saída, o decodificador lança um valor de probabilidade resultante da passagem dos valores de observação pelo HMM.

Após passar pelos decodificadores de cada palavra, o sistema analisa as probabilidades resultantes de cada HMM, selecionando o maior valor como sendo correspondente à palavra captada. Este processo pode ser observado na Figura 4.4.

```
exec("energiaDO_02.sce");
exec("det_Borda_variavel.sce");
yt = y(Ein*Nover:(Efn+1)*Nover);

exec("mel2freq.sci");
exec("freq2mel.sci");
exec("melfiltermatrix.sce");
exec("computeSpectrum.sci");
exec("computeMelSpectrum.sci");

W = melFilterMatrix(fs, 512, 20);
MELT = computeMelSpectrum(W, 256, yt);

wwt = zeros(size(MELT.M));
[g,v] = size(MELT.M);
for ii=1:v
    [ZJt, Jt, d] = nearest(z, MELT.M(:,ii));
    wwt(ii) = Jt;
end
wwt

[LPum,Xum]=dviterbi(PinitUM,AUM,BUM,wwt);
[LPdo,Xdo]=dviterbi(PinitDO,ADO,BDO,wwt);
[LPtr,Xtr]=dviterbi(PinitTR,ATR,BTR,wwt);
[LPsi,Xsi]=dviterbi(PinitSI,ASI,BSI,wwt);
[LPno,Xno]=dviterbi(PinitNO,ANO,BNO,wwt);
[LPsa,Xsa]=dviterbi(PinitSA,ASA,BSA,wwt);
[LPpu,Xpu]=dviterbi(PinitPU,APU,BPU,wwt);

[LPum LPdo LPtr LPsi LPno LPsa LPpu]
max(LPum, LPdo, LPtr, LPsi, LPno, LPsa, LPpu)
```

Figura 4-4 – Algoritmo de teste realizado na plataforma Scilab para o decodificador de Viterbi.

Por mais eficiente que o processo de reconhecimento utilizado pareça ser, este sofre interferência de fatores como a qualidade das amostras obtidas para treinamento, a qualidade da amostra captada pelo microfone, os valores obtidos na construção do *codebook*, as estimativas feitas para os modelos de HMM. Desta forma, a precisão do decodificador tende a diminuir.

4.3. Considerações Finais

Para analisar o funcionamento do reconhecedor de fala, foi proposta a construção de um protótipo para o sistema, consistido apenas da funcionalidade básica de

reconhecimento de fala, tendo em vista que o *Quiz* serve apenas como prova de conceito para o processo de reconhecimento de fala. Desta maneira, tem-se como especificação do protótipo:

- Início do jogo com a vinheta de introdução e pedido de confirmação para o usuário;
- Após confirmação, começa uma bateria composta de 7 (sete) questões pré-determinadas cuja resposta deve ser UM, DOIS ou TRÊS, além do reconhecimento dos comandos PULAR para passar para a próxima pergunta e SAIR para fechamento do aplicativo;
- Para cada resposta dada pelo usuário, o sistema pede uma confirmação da resposta, esperando pelos comandos SIM ou NÃO.

A construção e validação deste protótipo juntamente à avaliação do funcionamento do sistema reconhecedor de discurso são consideradas como eventos suficientes para determinar os objetivos iniciais do projeto como alcançados, já que as demais funcionalidades definidas na especificação do projeto, assim como o plano de obtenção de amostras também contida no documento de especificação são apenas complementações objetivadas a aumentar os níveis de interatividade, funcionalidade e precisão do sistema.

5. Integração com o Aparelho Móvel

Neste capítulo são apresentadas as ferramentas usadas para implementação do reconhecedor de fala no aparelho móvel (PDA). As tecnologias e plataformas utilizadas como o *Windows Mobile 6.0*, *.NET Compact Framework* e *Visual Studio 2008,* assim como as descrições de hardware.

5.1. Aparelho móvel

Atualmente existem dispositivos móveis (PDA's) de diversas capacidades, funcionalidades e configurações. Cada modelo busca atender uma necessidade específica e pode ser apenas computadores portáteis ou possuir também a função de telefone móvel (*Smartphones*). Programas como calendário, lista de tarefas e agenda eletrônica são os encontrados com maior freqüência nos modelos comerciais e a maioria dispõe de diversos modos de conectividade, tais como, Internet sem fio (*wireless*), infravermelho (irDA) ou bluetooth.

Nos dispositivos modernos, a interface com o usuário pode ser feita através de *touch screen*, tecnologia onde o usuário utiliza o visor do aparelho para inserir informações e acessar atalhos de programas, ou através de botões convencionais.

Assim como os computadores pessoais comuns os PDAs estão se tornando cada vez mais compactos, rápidos e com um poder de armazenamento maior.

A escolha do aparelho móvel obedeceu quatro critérios de seleção: capacidade de processamento, capacidade de memória volátil (RAM), presença de tecnologia Bluetooth e sistema operacional Microsoft® Windows Mobile® 5 ou superior. O aparelho selecionado foi o HP iPaq 116, cujas características estão apresentadas na Tabela 5-1.

Tabela 5-1 – Especificação do Aparelho

Recursos do sistema	***Descrição***
Processador	Marvell® PXA 310
Sistema Operacional	Microsoft® Windows Mobile® 6.0
Memória	256 MB Flash e 64 MB SDRAM
Alimentação externa	Adaptador de alimentação: entrada 100-240 Vca, Carregador USB: 5 Vcc, 100/500 mA
Tela	3,5 polegadas QVGA TFT com tela de toque
Slot SD	Permite memória SD
Conector de fone de ouvido	3,5 mm com suporte de 3 e 4 pinos para fones de ouvido estéreos ou VoIP
Antena	Antenas WLAN e Bluetooth internas
Áudio	Alto-falante e um conector de fones de ouvido estéreos de 3,5 mm
Bateria	Íon de lítio removível/recarregável de 1.200 mAh
Bluetooth	Perfis de dispositivo Bluetooth 2.0: Viva-voz/ OBEX/ PAN/FTP/ Porta Serial/ A2DP, alcance de 10 metros – Alta velocidade, baixo consumo, comunicação sem fio de curto alcance com outros dispositivos Bluetooth
WLAN	IEEE 802.11b/g

5.2. Integração

A integração do software do aparelho móvel e os algoritmos desenvolvidos no Scilab é realizada através da comparação do codebook e das HMM's. As matrizes numéricas geradas pelas amostras coletadas são exportadas no formato csv e armazenadas dentro do aparelho móvel O csv, comma-separated values, é um formato de arquivo que armazena dados tabelados separados por um delimitador e usa a vírgula e a quebra de linha para separar os valores. O formato csv é bastante simples e utilizado por quase todas as planilhas eletrônicas e sistemas de bancos de dados e, por este motivo, foi escolhido como modelo de armazenamento de dados do sistema.

O sistema para o aparelho móvel foi desenvolvido utilizando o *.NET Compact* Framework versão 3.5 e a IDE Visual Studio 2008. O .NET Compact Framework 3.5 é uma atualização da versão 2.0 e além das melhorias na funcionalidade existente na versão anterior, com novas funcionalidades para o desenvolvimento em dispositivos móveis. O Visual Studio (VS) é um conjunto de ferramentas de desenvolvimento para aplicações ASP.NET, Web Services, Aplicações Desktop e Aplicações Mobile. A linguagem utilizada, Visual C# utiliza o VS como IDE de desenvolvimento. Para aplicações móveis, o VS 2008 oferece ferramentas que permitem o desenvolvimento e a depuração dos aplicativos em dispositivos móveis através do activesync. O *Visual Studio* também possui emuladores que possibilitam depurar o sistema sem a necessidade do dispositivo físico.

5.3. Resultados

A fim de testar a usabilidade do sistema (Figura 5.1), foram realizados testes do protótipo com cerca de 20 alunos da Escola Politécnica. Os resultados obtidos foram, de uma forma geral, satisfatórios e, apesar de uma taxa menor de reconhecimento que o esperado, o protótipo se mostrou atrativo aos usuários.

A análise dos testes não pode ser conclusiva devido a baixa quantidade de usuários testados e a semelhança em seus perfis de fala. Em um cenário ideal seria necessário o teste com usuários de diversas localidades e assim poderia ser avaliada a eficiência do software na análise de diversos tipos de sotaques e formas de expressão de uma mesma palavra.

Figura 5-1 – Design das telas do aplicativo.

5.4. Considerações Finais

O software apresentou um desempenho aquém do esperado devido as limitações do dispositivo e excesso de ruído no ambiente; no entanto, os algoritmos de reconhecimento desenvolvidos mostraram-se muito eficientes quando testados em seu ambiente de desenvolvimento (Scilab) e utilizando microfone em estúdio como mostrado na Tabela 5.2.

Tabela 5-2 – Tabela de Precisão (Estúdio)

Palavra	*Taxa de acerto*
Um	89%
Dois	96%
Três	87%
Sim	91%
Não	96%
Sair	100%
Pular	92%
Média	93%

Considerando os valores obtidos no mesmo Scilab com as gravações realizadas com o microfone Bluetooth, levando em consideração a distância do aparelho à boca do usuário e ruídos remanescentes da transmissão pelo protocolo, obtemos os seguintes valores:

Tabela 5-3 - Tabela de Precisão (Microfone Bluetooth)

Palavra	*Taxa de acerto*
Um	72%
Dois	78%
Três	76%
Sim	77%
Não	74%
Sair	85%
Pular	81%
Média	78%

6. Considerações Finais

Nesta seção é exposta a visão pós-projeto, as conclusões e finalizações referentes às atividades de projeto, as contribuições para a vida acadêmica e profissional dos membros do grupo, além da previsão de possíveis complementações deste trabalho e dos leques abertos para o surgimento de soluções inovadoras.

6.1. Conclusões

O grupo entende que houve sucesso nos objetivos propostos por este projeto, que foram a construção do reconhecedor de fala por intermédio da plataforma Scilab e a integração das métricas de referência (Codebook) ao aplicativo desenvolvido na linguagem C#, para o aparelho móvel.

A utilização de técnicas e teorias avançadas, como a transformada rápida de Fourier, os algoritmos de Markov e Viterbi, complementaram, com experiência prática na resolução de problemas complexos, a formação teórica recebida. No desenvolvimento do protótipo, foi necessária também a preocupação com as restrições dos dispositivos móveis, saindo assim do campo teórico das situações ideais.

6.2. Contribuições

Foram muitas as contribuições deste projeto em termos acadêmicos e também profissionais, sendo citados os seguintes pontos importantes:

I. Contribuição no que diz respeito à organização de projetos, levando em conta riscos, imprevistos e outras intempéries que foram inclusive vividas pelo grupo no decorrer do projeto;

II. Contribuição nos conceitos de engenharia de software devido à inclusão de processos e algoritmos relacionados à parte de processamento de sinais de áudio, à especificação de armazenamento de dados de áudio e construção de software para aparelhagem móvel, com recolhimento de

dados em tempo real e processamento destes de forma a obter uma resposta rápida;

III. Contribuição em termos extracurriculares com a inserção do tema reconhecimento de fala e nos estudos relacionados dos processos utilizados, inclusive usufruindo de métodos estocásticos como o modelo de Markov.

6.3. Trabalhos Futuros

Como complementos ao trabalho realizado neste projeto, podem ser citados:

I. A conclusão das funcionalidades e a melhoria dos níveis de precisão do protótipo entregue;

II. A utilização de métodos de aprendizagem do sistema, que consiste no armazenamento das amostras de voz do próprio usuário do programa em tempo real, com o intuito de aumentar a precisão do reconhecimento pelo aumento do número de amostras para treinamento;

III. Quebras de paradigma e utilização do reconhecimento de fala para o desenvolvimento de novos sistemas, com escopos variados e funcionalidades diferentes, viabilizando, desta maneira, a exploração de nichos de mercado.

7. Referências Bibliográficas

AAS, K.; EIKVIL, L.; ANDERSEN, T. Text recognition from grey level images using hidden Markov models. *Lecture Notes in Computer Science*, v. 970, p. 503–508, 1995.

ALMEIDA, G M. – Detecção de Situações Anormais em Caldeiras de Recuperação Química – Escola Politécnica, 2006.

BASHIR, F.; KHOKHAR, A.; SCHONFELD, D. Hmm-based motion recognition system using segmented pca. *Proc. IEEE International Conference on Image Processing*, IEEE Computer Society, Genova, Italy, 2005.

FORNEY JR., G. D. – The Viterbi Algorithm – Invited Paper – IEEE, 1973.

GÓMEZ-CIPRIANO, J., NUNES, R. P., BAMPI, S., BARONE, D. – Arquitetura Específica de Pré-processamento com Extração de parâmetros Mel-cepstrais para um Sistema de Reconhecimento Automático de Voz, UFRGS – Instituto de Informática.

HU, J.; BROWN, M. K.; TURIN, W. Hmm based on-line handwriting recognition. *IEEETrans. Pattern Anal. Mach. Intell.*, IEEE Computer Society, Washington, DC, USA, v. 18, n. 10, p. 1039–1045, 1996. ISSN 0162-8828.

JELINEK, F. – Statistical Methods for Speech Recognition – The MIT Press, London, 1997.

MONTERO, J. A.; SUCAR, L. E. Feature selection for visual gesture recognition using hidden markov models. In: *ENC*. [S.l.: s.n.], 2004. p. 196–203.

RABINER, L. – A Tutorial on Hidden Markov Models and Selected Application in Speech Recognition – Proceedings of the IEEE, Vol. 77, nº 2. February 1989.

RABINER, L., JUANG, B. – Fundamentals of Speech Recognition. PTR Prentice-Hall, New Jersey, 1993.

Certificação Digital nº. 0310467/CA, Cap. 4 – PUC-RJ.

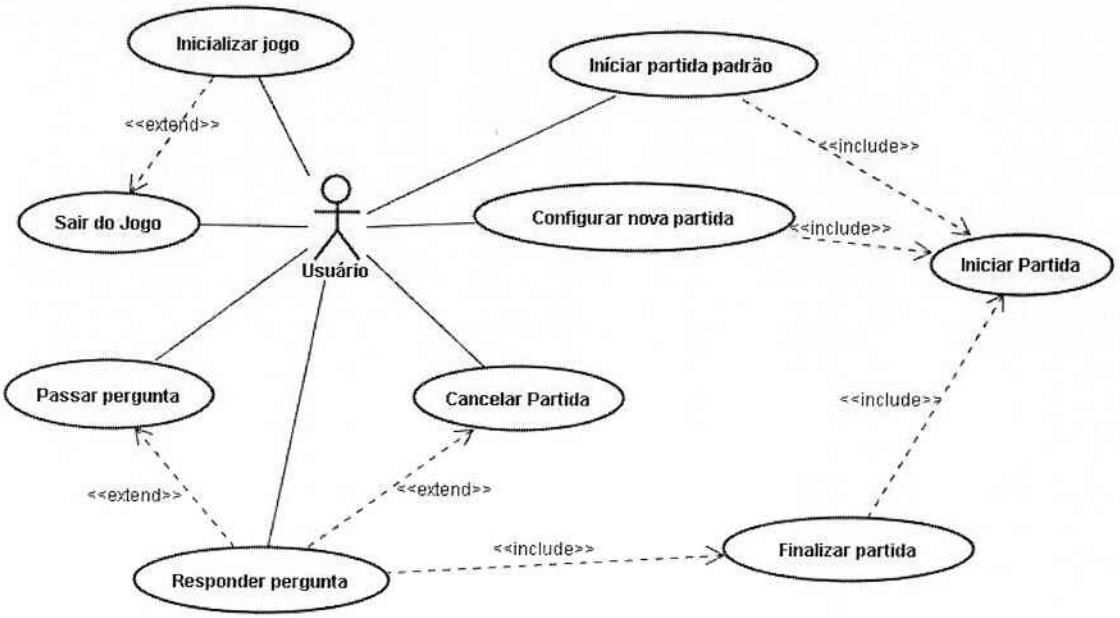
Advanced Audio Distribution Profile Specification, Bluetooth Audio Video Working Group. Version 1.0. (http://www.bluetooth.com).

APÊNDICE A - Modelo de Casos de Uso

- Atores

O sistema consiste em um jogo de perguntas e respostas direcionado a uma única pessoa. Portanto este usuário será o ator do sistema.

- Diagrama de Casos de Uso



- Descrição dos Casos de Uso

Caso de uso 1:

Inicializar Jogo.

Descrição:

Este caso de uso descreve o processo de inicialização do jogo pelo usuário.

Evento iniciador:

Usuário solicita execução do aplicativo referente ao jogo.

Atores:

Usuário

Pré-condição:

Nenhuma.

Seqüência de eventos:

1. Usuário abre aplicativo referente ao jogo
2. Sistema abre tela inicial e pergunta ao usuário se deseja iniciar uma partida padrão.

Pós-condição: aplicativo iniciado.

Extensões:

1. Usuário opta por sair do jogo, sistema fecha o aplicativo.

Inclusões:

Nenhuma.

Caso de uso 2:

Iniciar partida padrão

Descrição:

Este caso de uso descreve o processo de início de uma partida nas configurações padrão.

Evento iniciador:

Usuário aceita iniciar partida padrão.

Atores:

Usuário

Pré-condição:

Nenhuma.

Seqüência de eventos:

1. Usuário aceita iniciar uma partida padrão
2. Sistema inicia uma nova partida zerando a pontuação e selecionando as perguntas referentes ao nível de dificuldade padrão.

Pós-condição: regras configuradas, pontuação zerada e sistema pronto para iniciar com a primeira pergunta.

Extensões:

Nenhuma.

Inclusões:

1. Caso de uso 'Iniciar Partida' (passo 2).

Caso de uso 3:

Configurar nova partida

Descrição:

Este caso de uso descreve o processo de configuração das regras de uma nova partida pelo usuário.

Evento iniciador:

Usuário não aceita iniciar partida padrão.

Atores:

Usuário

Pré-condição:

Nenhuma.

Seqüência de eventos:

1. Usuário não aceita iniciar uma partida padrão
2. Sistema pergunta qual o nível de dificuldade das perguntas oferecendo três alternativas.
3. Usuário escolhe um dos níveis de dificuldade.
4. Sistema pergunta qual o número de perguntas de uma partida oferecendo três alternativas.
5. Usuário escolhe uma das quantidades de perguntas de uma partida.
6. Sistema inicia uma nova partida zerando a pontuação e selecionando as perguntas referentes ao nível de dificuldade escolhida pelo usuário.

Pós-condição: regras configuradas, pontuação zerada e sistema pronto para iniciar com a primeira pergunta.

Extensões:

Nenhuma.

Inclusões:

1. Caso de uso 'Iniciar Partida' (passo 6).

Caso de uso 4:

Responder Pergunta.

Descrição:

Este caso de uso descreve o processo de enunciação das perguntas pelo sistema, respostas do usuário e correção destas.

Evento iniciador:

Partida Inicializada.

Atores:

Usuário

Pré-condição:

Nenhuma.

Seqüência de eventos:

1. Sistema enuncia a pergunta ao usuário e lhe oferece três alternativas.
2. Usuário responde uma das alternativas propostas.
3. Sistema pede confirmação da resposta.
4. Usuário confirma (sim / não).
5. Sistema confere resposta e atribui os pontos ao usuário.
6. Sistema prepara próxima pergunta e torna ao passo 1.
7. Se as perguntas acabaram, sistema soma a pontuação informando o usuário e pergunta se o usuário deseja iniciar uma nova partida.

Pós-condição: fim de partida.

Extensões:

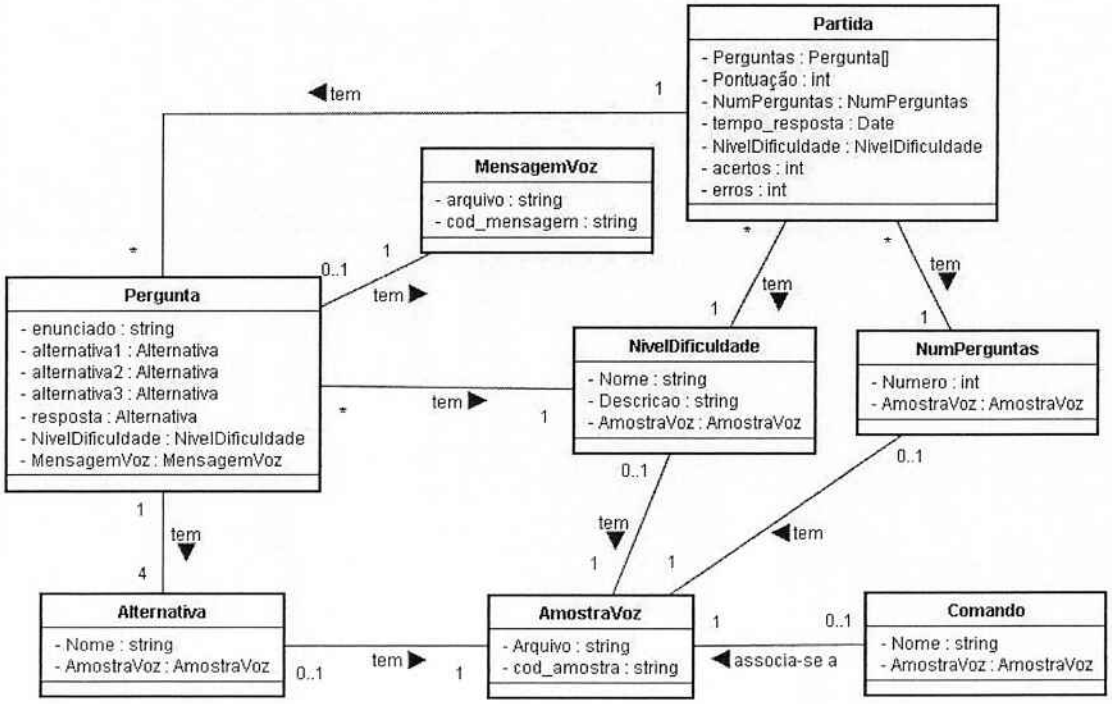
1. Usuário passa a pergunta, sistema não computa nenhum ponto ao usuário e parte para a próxima pergunta tornando ao passo 1 (passo 2).
2. Usuário cancela a partida, sistema volta à tela inicial (passos 2, 4).

Inclusões:

1. Caso de Uso 'Finalizar Partida (passos 7).

APÊNDICE B - Modelo de Classes

- Diagrama de Classes



- Descrição das Classes

Partida	Corresponde a uma nova partida.	Perguntas: Perguntas selecionadas para uma partida.
		Pontuação: pontos feitos pelo usuário na partida.
		NumPerguntas: Número de perguntas para aquela partida.
		Tempo_resposta: tempo para resposta do usuário a cada pergunta.
		NivelDificuldade: nível de dificuldade da partida.

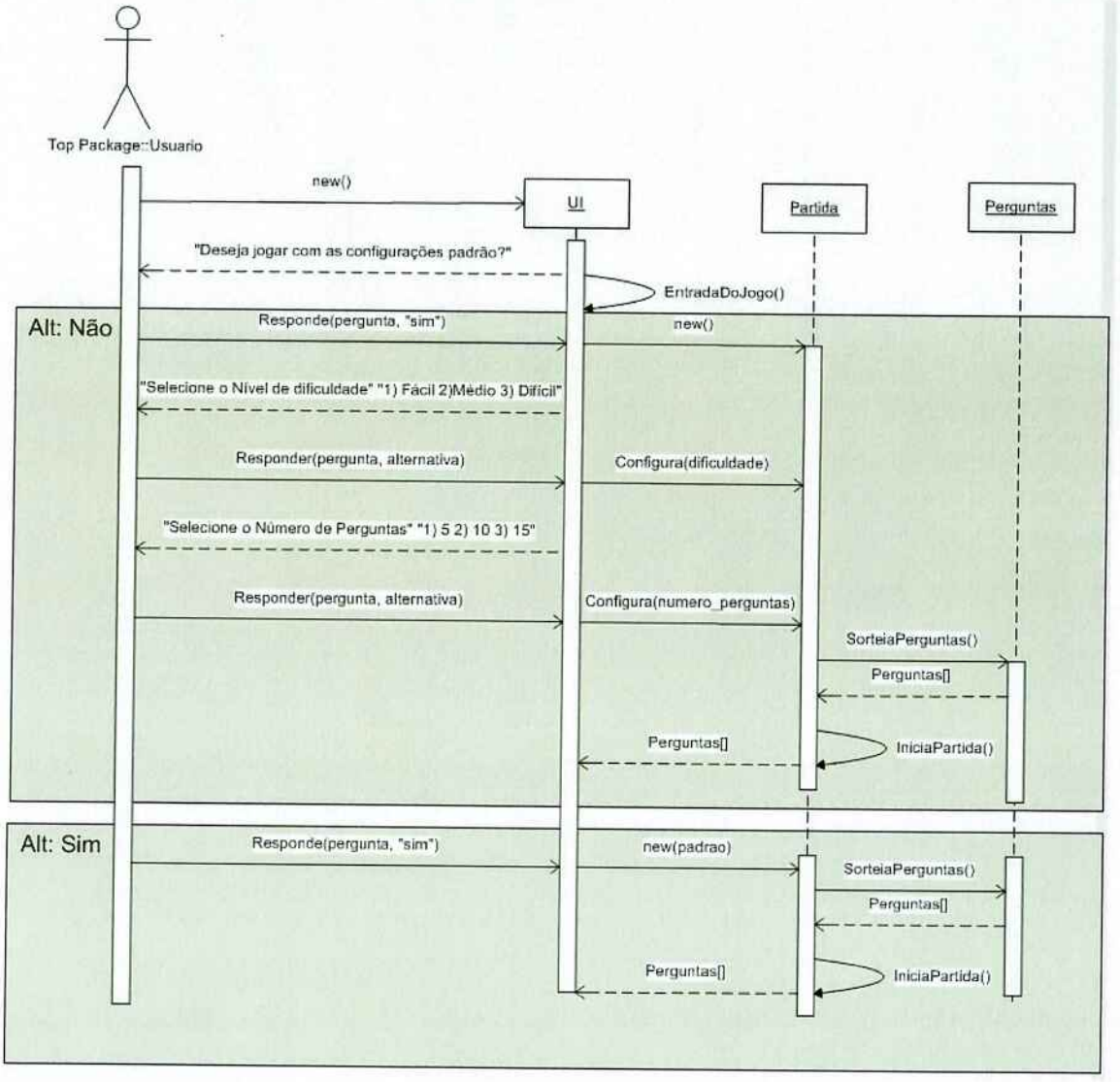
		Acertos: número de acertos do usuário na partida.
		Erros: número de erros do usuário na partida.
Pergunta	Corresponde a uma pergunta dentre as várias de uma partida	Enunciado: enunciado escrito da pergunta.
		Alternativa1: primeira alternativa da pergunta
		Alternativa1: segunda alternativa da pergunta
		Alternativa1: terceira alternativa da pergunta
		Resposta: Dentre as alternativas, aquela que é a correta.
		NivelDificuldade: nível de dificuldade da pergunta.
		MensagemVoz: mensagem de voz relacionada ao enunciado da pergunta.
Alternativa	Uma alternativa relacionada a certa pergunta	Nome: Nome associado à alternativa
		AmostraVoz: Amostra de voz na base de dados relacionada à alternativa.
NivelDificuldade	Representa o nível de dificuldade seja de uma pergunta ou de uma partida	Nome: Nome associado ao nível de dificuldade.
		Descrição: Descrição do nível de dificuldade.

		AmostraVoz: Amostra de voz na base de dados relacionada ao nível de dificuldade.
NumPerguntas	Número de perguntas para uma partida	Número: representa a quantidade de questões para a partida.
		AmostraVoz: Amostra de voz na base de dados relacionada ao número de perguntas.
Comando	Representa o comando vocal dado pelo usuário	Nome: nome associado ao comando.
		AmostraVoz: amostra de voz mais semelhante ao comando dado pelo usuário.
MensagemVoz	Representa uma mensagem de voz do sistema, seja na orientação do usuário pelo sistema como também enunciados e alternativas para as perguntas	Arquivo: caminho do arquivo de áudio correspondente à mensagem de voz.
		Cod_mensagem: código associado à mensagem de voz.
AmostraVoz	Amostra de voz na base de dados para referência	Arquivo: caminho do arquivo de áudio correspondente à amostra de voz.
		Cod_amostra: código associado à amostra de voz.

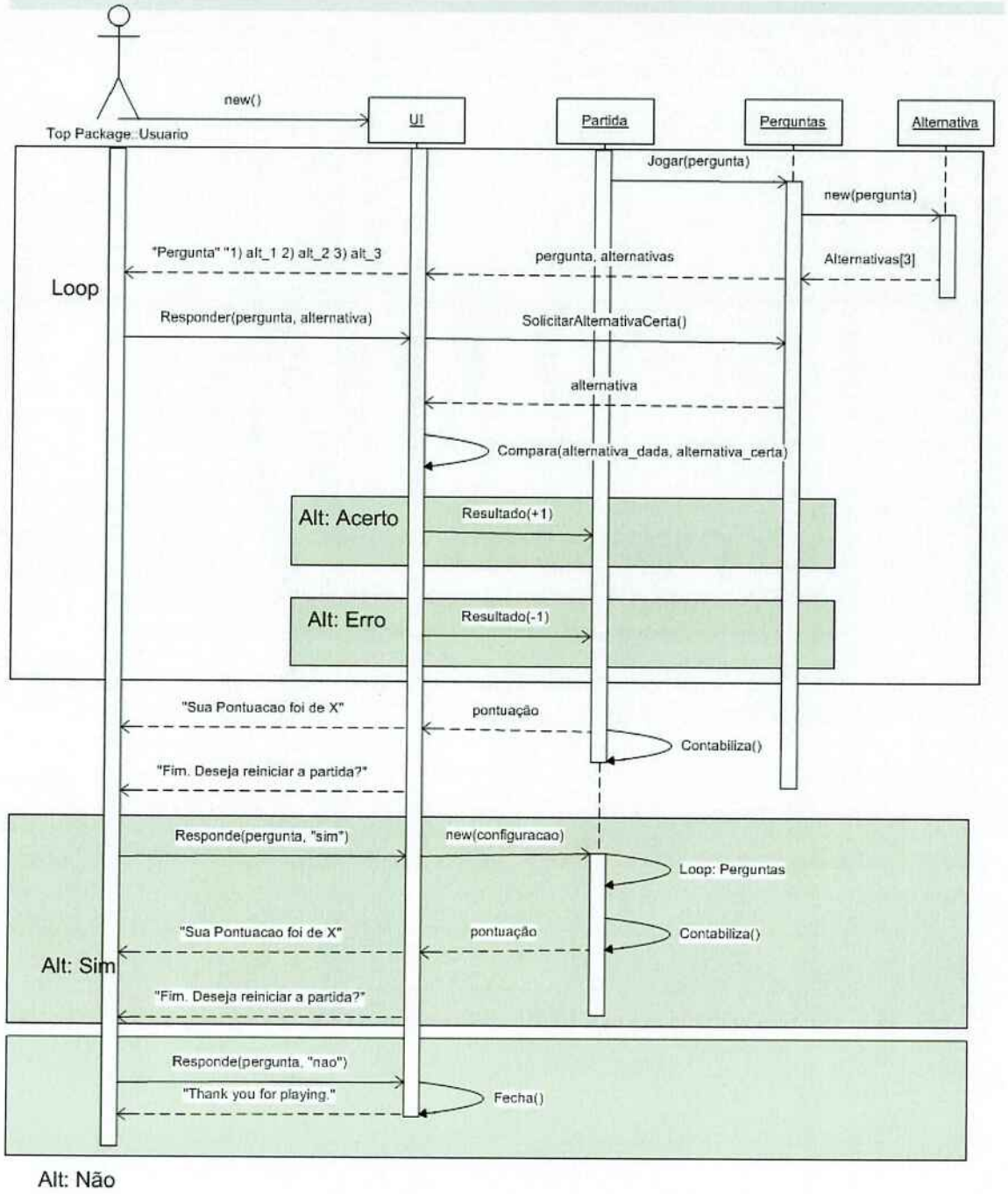
APÊNDICE C - Modelo de Interação

Para o modelo de Interação são apresentados os digramas de sequência das funcionalidades principais do sistema: a configuração de uma partida e o decorrer da partida em si. Nestas funcionalidades, as classes requisitadas são praticamente as mesmas: Partida, Pergunta e Alternativa. As classes referentes ao nível de dificuldade e às opções de quantidade de perguntas foram omitidas, sendo consideradas como "atributos" de uma Partida, simplificando a interpretação do diagrama.

- Diagrama de Seqüência: Configuração de Partida

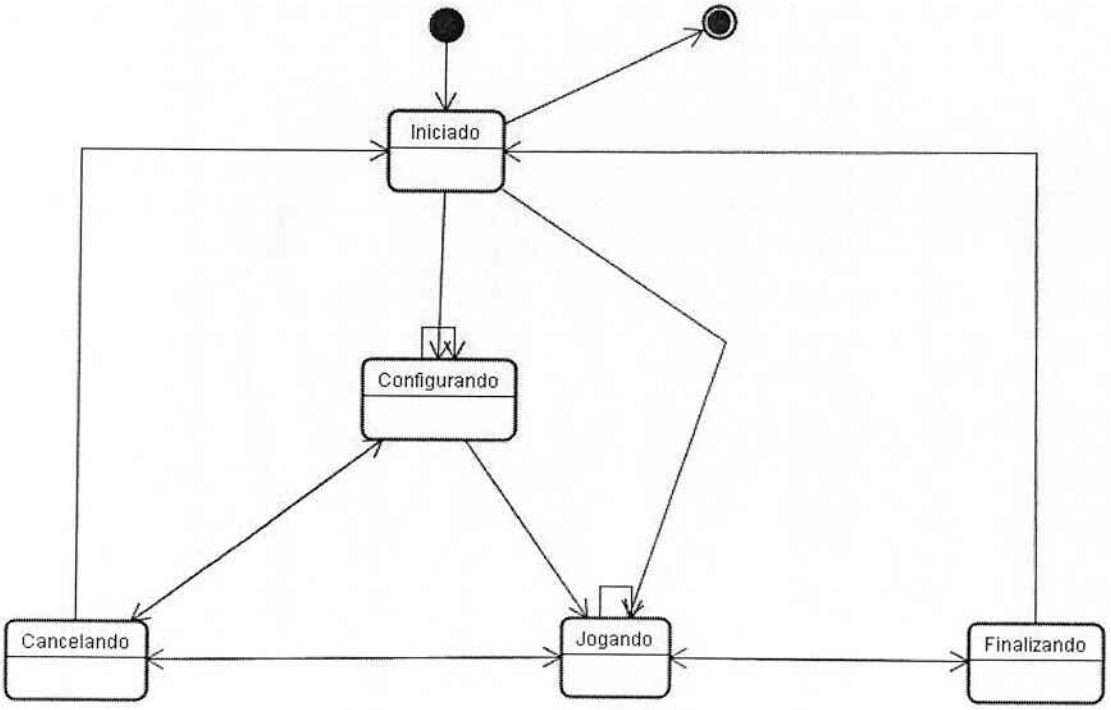


- Diagrama de Seqüência: Decorrer de uma Partida



APÊNDICE D - Modelo de Estados

- Diagrama de Estados



- Descrição dos Estados

Estado 1:

Iniciado.

Descrição:

Este estado representa a tela inicial do sistema, quando é perguntado ao usuário se este deseja iniciar uma partida padrão ou previamente configurada.

Transições:

1. Estado Inicial → Iniciado: Usuário abre aplicativo.
2. Cancelando → Iniciado: Usuário confirma cancelamento de ação.
3. Finalizando → Iniciado: usuário opta por não iniciar uma nova partida.
4. Iniciado → Configurando: usuário opta por iniciar partida configurada.
5. Iniciado → Jogando: usuário opta por iniciar partida padrão.

Estado 2:

Configurando.

Descrição:

Este estado representa o processo de configuração de uma nova partida.

Transições:

1. Iniciado → Configurando: usuário opta por iniciar partida configurada.
2. Configurando → Jogando: usuário termina configuração e sistema cria nova partida.
3. Configurando → Cancelando: usuário cancela configuração, sistema pede confirmação.
4. Cancelando → Configurando: Usuário desiste de cancelamento de configuração.

Estado 3:

Jogando.

Descrição:

Este estado representa a disputa de uma partida pelo usuário.

Transições:

1. Iniciado → Jogando: usuário opta por iniciar partida padrão.
2. Configurando → Jogando: usuário termina configuração e sistema cria nova partida.
3. Jogando → Finalizando: Terminam as perguntas e o sistema finaliza partida perguntando se o usuário deseja fazer nova partida.
4. Finalizando → Jogando: usuário opta por iniciar nova partida.
5. Jogando → Cancelando: usuário cancela partida, sistema pede confirmação.
6. Cancelando → Jogando: usuário desiste de cancelamento de partida.

Estado 4:

Finalizando.

Descrição:

Este estado representa a finalização do jogo após a última pergunta. O sistema informa a pontuação do jogo e pergunta ao usuário se este deseja iniciar uma nova partida.

Transições:

1. Jogando → Finalizando: Terminam as perguntas e o sistema finaliza partida perguntando se o usuário deseja fazer nova partida.
2. Finalizando → Jogando: usuário opta por iniciar nova partida.
3. Finalizando → Iniciado: usuário opta por não iniciar uma nova partida.

Estado 5:

Cancelando.

Descrição:

Este estado representa o cancelamento de uma ação pelo usuário. O sistema pede sempre uma confirmação do cancelamento.

Transições:

1. Jogando → Cancelando: usuário cancela partida, sistema pede confirmação.
2. Cancelando → Jogando: usuário desiste de cancelamento de partida.
3. Configurando → Cancelando: usuário cancela configuração, sistema pede confirmação.
4. Cancelando → Configurando: Usuário desiste de cancelamento de configuração.
5. Cancelando → Iniciado: Usuário confirma cancelamento de ação.

APÊNDICE E - Descrição das Interfaces Externas

Dado que os comandos e mensagens não são visualizáveis, mas sim fornecidos vocalmente, somente faz sentido descrever suas entradas e saídas, baseado nos estados descritos acima.

A) Mensagem de Boas Vindas

Esta é a primeira mensagem do sistema. Como apresentação é dita uma frase de boas vindas e o sistema pergunta se o usuário deseja iniciar o jogo com as configurações padrão ou se deseja configurar o sistema

Saída: *"Seja bem vindo ao jogo de perguntas e respostas. Deseja jogar com as configurações Padrão? Responda Sim ou Não."*

Entrada (Resposta vocal do usuário após a saída):

- **Sim:** O jogo é iniciado com configurações padrão. Confirme a resposta na tela de confirmação (**I**). Se confirmado, pula à tela (**D**).
- **Não:** O jogador tem a chance de configurar seu jogo. Confirme a resposta na tela de confirmação (**I**). Se confirmado, pula à tela (**B**).

B) Comandos de Configuração – Nível de Dificuldade

Esta é a primeira configuração possível do sistema. Nela o usuário escolhe a dificuldade do jogo.

Saída: *"Selecione o nível de dificuldade.*
Alternativa 1) Fácil
Alternativa 2) Médio
Alternativa 3) Difícil"

Entrada (Resposta vocal do usuário após a saída):

- **Um:** O jogo é iniciado somente com perguntas de nível fácil. Confirme a resposta com o comando de confirmação (**I**). Se confirmado, vai para próxima tela de configuração (**C**).
- **Dois:** O jogo é iniciado somente com perguntas de nível médio. Confirme a resposta com o comando de confirmação (**I**). Se confirmado, vai para próxima tela de configuração (**C**).

- **Três:** O jogo é iniciado somente com perguntas de nível difícil. Confirme a resposta com o comando de confirmação (**I**). Se confirmado, vai para próxima tela de configuração (**C**).

C) Comandos de Configuração – Quantidade de Perguntas

Esta é a segunda configuração possível do sistema. Nela o usuário escolhe a quantidade de perguntas que serão sorteadas

Saída: *"Selecione o número de perguntas.*
Alternativa 1) 5
Alternativa 2) 10
Alternativa 3) 15"

Entrada (Resposta vocal do usuário após a saída):

- **Um:** O jogo é iniciado com 5 perguntas. Confirme a resposta com o comando de confirmação (**I**). Se confirmado, vai para o início do jogo, na tela (**D**).
- **Dois:** O jogo é iniciado com 5 perguntas. Confirme a resposta com o comando de confirmação (**I**). Se confirmado, vai para o início do jogo, na tela (**D**).
- **Três:** O jogo é iniciado com 5 perguntas. Confirme a resposta com o comando de confirmação (**I**). Se confirmado, vai para o início do jogo, na tela (**D**).

D) Mensagens e Comandos – Sistema Fazendo uma Pergunta

Modo em que o jogo acontece propriamente dito. O sistema lê uma pergunta sorteada dentro da configuração desejada e a faz esperando uma resposta ou um estouro no tempo de espera.

Saída: *"Pergunta X.*
Alternativa 1) a
Alternativa 2) b
Alternativa 3) c"

Entrada (Resposta vocal do usuário após a saída):

- **Um:** O usuário seleciona como resposta a alternativa 1. Confirme a resposta com o comando de confirmação (**I**). Se confirmado, a resposta

é avaliada, caso seja um acerto vai para a mensagem de acerto (**E**), caso seja um erro, vai para a mensagem de erro (**F**).

- **Dois:** O usuário seleciona como resposta a alternativa 2. Confirme a resposta com o comando de confirmação (**I**). Se confirmado, a resposta é avaliada, caso seja um acerto vai para a mensagem de acerto (**E**), caso seja um erro, vai para a mensagem de erro (**F**).
- **Três:** O usuário seleciona como resposta a alternativa 3. Confirme a resposta com o comando de confirmação (**I**). Se confirmado, a resposta é avaliada, caso seja um acerto vai para a mensagem de acerto (**E**), caso seja um erro, vai para a mensagem de erro (**F**).

E) Mensagem de Acerto

Depois que a reposta é recebida pelo sistema, esta é comparada com o gabarito e esta mensagem representa um acerto nessa comparação, somando 1 ponto ao usuário. Depois, vai automaticamente, sem a intervenção do usuário, para a próxima pergunta.

Saída: *"Parabéns! Você acertou!"*

F) Mensagem de Erro

Depois que a reposta é recebida pelo sistema, esta é comparada com o gabarito e esta mensagem representa um erro nessa comparação, somando 1 ponto ao usuário. Depois, vai automaticamente, sem a intervenção do usuário, para a próxima pergunta.

Saída: *"Infelizmente você errou."*

G) Mensagem de Fim de Jogo

Depois que a última pergunta foi respondida, o sistema passa automaticamente para a mensagem de fim de jogo, mostrando a pontuação do usuário e perguntando se ele que reiniciar o jogo.

Saída: *"Parabéns! Sua pontuação foi de X. Deseja jogar novamente?"*

Entrada (Resposta vocal do usuário após a saída):

- **Sim:** O jogo é iniciado com configurações anteriores. Confirme a resposta com o comando de confirmação (**I**). Se confirmado, pula ao modo (**D**).

- **Não:** O jogador deseja interromper o jogo. Vai para o modo de finalização, (**H**).

H) Comando de Escape

Este comando é a saída do jogo. Pode ser acessado a qualquer momento quando o usuário diz a palavra sair ou depois que ele chegou ao fim do jogo. Basicamente, este comando agradece e logo depois o sistema morre.

Saída: *"Muito obrigado por jogar!"*

I) Comando de Confirmação de Resposta

Para tornar mais aceitável o jogo, o sistema depois de ouvir qualquer resposta, sempre repete a alternativa escolhida e vê se o usuário realmente escolheu aquela.

Saída: *"Alternativa recebida é X. Deseja confirma esta resposta?"*

Entrada (Resposta vocal do usuário após a saída):

- **Sim:** O jogo trata aquela resposta como verdadeira e permite que a alternativa respondida seja computada, seguindo seu fluxo anterior.
- **Não:** O jogo trata aquela resposta como falsa e solicita que o usuário repita a resposta desejada.