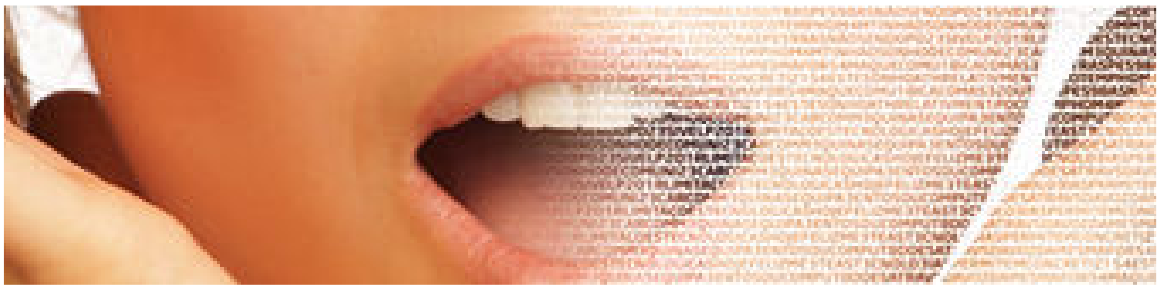
Luís Carlos Castro Monteiro

Reconhecimento de Voz:

Sistema de Reconhecimento do Orador, Baseado em Modelos de Markov, Compactado num Objecto COM para Windows



Dissertação submetida à Universidade do Minho para obtenção do título de Mestre em Engenharia Electrónica Industrial e Computadores.

Guimarães, 2007

Tese realizada sob orientação do Doutor Carlos Lima e co-orientação do Doutor Adriano Tavares, Professores Auxiliares do Departamento de Electrónica Industrial da Universidade do Minho.

Agradecimentos

Há muitas pessoas a quem devo agradecer pela ajuda que me prestaram durante o tempo deste projecto, que começou há um ano atrás. Não me irei, certamente, lembrar de todas, por isso desde já as minhas desculpas para aquelas que ficaram esquecidas.

Em primeiro lugar, é devida uma palavra de agradecimento aos meus orientadores, Professor Doutor Carlos Lima e Professor Doutor Adriano Tavares. Só a boa vontade com que acolheram e apoiaram este trabalho, bem como o incentivo, orientação e colaboração disponibilizadas ao longo deste ano tornaram possível esta tese.

Quero agradecer, de igual modo, à Agência de Inovação (ADI) pela Bolsa de Iniciação Científica (BIC), com a referência UMINHO/ PMDT/ TECNOVOZ/ BIC/ 01/ 2006 no âmbito do projecto de investigação "TECNOVOZ". A ela devo a indispensável ajuda do ponto de vista financeiro.

Um agradecimento especial à Maricica Nistor e às primas Carla Castro e Dulce Monteiro que se mostraram sempre disponíveis para me ajudar e contribuíram significativamente para a melhoria da apresentação e redacção deste documento.

Agradeço com carinho aos meus pais, irmão e cunhada pelo amor e apoio sempre demonstrados. A eles devo a força e a coragem para seguir e alcançar os meus sonhos.

Para finalizar, quero dedicar esta dissertação à Maricica Nistor.

Resumo

Esta dissertação descreve a implementação de um sistema de reconhecimento do orador que foi desenvolvido no âmbito de um projecto de modernização da economia (PME) com a referência PMDT 01/165, que se denomina projecto TECNOVOZ. Este tipo de projectos têm como principal objectivo transferir a tecnologia dos meios científicos para a comunidade.

O reconhecimento do orador vai ser usado com a finalidade de restringir o acesso a informação confidencial (e.g. relatórios médicos) em ambientes tipo "mãos livres" ou ainda no controlo de acessos a áreas reservadas. Ambas as aplicações requerem desempenho elevado do Reconhecedor, pelo que se optou pelo reconhecimento de palavras isoladas dependentes do texto (ISR - Isolated Word Recognition), o que significa também mais requisitos computacionais.

Este sistema de reconhecimento do orador ("Reconhecimento de "Voz") é baseado em HMM´s de observações contínuas e teve como ponto de partida um Reconhecedor, desenvolvido no contexto de um trabalho de doutoramento [3]. Este sistema está implementado em C++ (uma linguagem de programação) através da ferramenta Visual C++ e está inserido num objecto COM para Windows. Com a criação deste objecto COM, converte-se esta implementação num *software* dinâmico, isto é, num sistema que possa ser interligado com qualquer outro *software* que necessite de reconhecimento de "Voz". Este é um requisito fundamental dado que o Reconhecedor vai ser inserido em aplicações já existentes, desenvolvidas em diversas plataformas.

A utilidade prática do Reconhecedor do orador implica alguma flexibilidade ao nível da sua estrutura, como seja a adaptação a um novo conjunto de oradores, o que acontece sempre que por exemplo mais um orador seja inserido no sistema.

Esta flexibilidade de adaptação a um novo conjunto de oradores foi conseguida pela disponibilização de modelos independentes do orador, mas dependentes

do texto. Deste modo, a inserção de um novo orador pode fazer-se tomando como modelo inicial o modelo independente do orador.

Foi, ainda, implementada uma pequena aplicação de interface com o objecto COM. O objectivo deste *software* é testar os serviços, de treino e de reconhecimento, que este COM disponibiliza, visto que é um sistema isolado que necessita de interacção por parte de alguma aplicação para fornecer os serviços.

Abstract

This paper describes the system implementation of the speaker recognition which was developed under a project for modernizing the economy (PME) with the PMDT 01/165 reference, which called TECNO-VOICE project. Such projects have the primary aim of transferring the technology to scientific processes for the community.

The recognition of the speaker will be used in order to restrict access to confidential information (e.g. medical reports) in environments such "free-hands" or in the control of access to restricted areas. Both applications require high performance of recognizer, so it was opted for the recognition of isolated words dependent on the text (ISR - Isolated Word Recognition), which also means more computational requirements.

This system of the speaker recognition ("Recognition of" Voice ") is based on HMM's of continuous observations and took as a point of departure a recognizer, developed in the context of a doctoral work [3]. This system is implemented in C + + (a programming language) through Visual C + + tool and is inserted into a COM object for Windows. With the creation of COM object, it converts this implementation in dynamic software, that is, a system that can be interconnected with any other software that requires recognition of "Voice". This is a fundamental requirement as the recognizer will be inserted into existing applications, developed on several platforms.

The practical usefulness of the speaker recognizer implies some flexibility in terms of its structure, such as adjusting to a new set of speakers, what happens always when, for example, another speaker is inserted on the system. This flexibility of adaptation to a new set of speakers was achieved by providing independent models of speaker, but dependent on the text. Thus the insertion of a new speaker may be using as an initial model an independent model of the speaker.

It was also implemented a small interface application with the COM object. The purpose of this software is to test the training and recognition services, which COM provides, since it is an isolated system that needs interaction from any application to provide the services.

Índice

Índice de ilustrações

Lista de códigos exemplos

Lista de tabelas

Abreviaturas

COM..... Component Object Model

HMM.....Hidden Markov Model

LPC..... Linear Predictive Coding

ZCR.....Zeros Crossing Rate

STE..... Short Time Energy

EM..... Expectation Maximisation

MFCC..... Mel Ferquency Cepstral Coefficients

PLP.....Perceptual Linear Prediction

i.i.d. .....Independentes e Identicamente Distribuídas

GUID..... Globally Unique Identifier for the Database

SNR.....Signal-to-Noise Ratio

ISR.....Isolated Word Recognition

PME.....Projecto de Modernização da Economia

ms.....Milissegundos

AT&T.....American Telephone and Telegraph

Capítulo 1

Introdução

Desde a origem da humanidade que a forma mais comum e natural de comunicação entre as pessoas é a "Voz". Entretanto, com a gradual e normal evolução humana, o homem começou a desenvolver máquinas. No início, essas máquinas eram pequenas e simples, mas nos dias de hoje, com a ajuda da ciência e com o grande desenvolvimento que a tecnologia sofreu nos últimos tempos, são criadas e desenvolvidas máquinas bastante complexas, onde os interfaces Homem – máquina nem sempre são intuitivos para o utilizador comum. Devido à problemática desses interfaces, a voz, como forma natural de comunicação, começou a ganhar relevo como forma alternativa de interacção entre Homem e máquina.

Comunicar com uma máquina da mesma forma que comunicar com as pessoas é um sonho antigo do Homem, mas tem sido impossível de se realizar, em parte, devido a limitações tecnológicas que incluem a falta de capacidade de modelar convenientemente as variabilidades do sinal de fala, mesmo para o mesmo orador e para a mesma oração.

Existem alguns tipos de variabilidades que dificultam o reconhecimento do orador, das quais se destacam a variabilidade intra – orador (quando para a mesma informação linguística o mesmo orador não produz a mesma informação acústica) e a variabilidade ambiental (distorções ou ruídos introduzidos pelo meio ambiente). Estes obstáculos vão tentar ultrapassar-se através da técnica dos HMM's (uma vez que estes se adaptam bem a variações de velocidade e forma de pronunciação das palavras) e da parametrização do sinal de fala através da análise

predição linear (técnica de extracção de características, apenas, do sinal de fala, porém, só apresenta bons resultados para relações sinal ruído superiores a 25 dB).

A crença na possibilidade de tornar esse sonho humano realidade, o interesse pessoal sobre esta área do conhecimento (processamento de sinal, programação, reconhecimento de padrões, etc.) que muitas vezes parece "magia", a responsabilidade de entrar num projecto grandioso e português na área da Fala designado por TECNOVOZ e o apoio de pessoas capazes e experientes nesta matéria (orientador e co-orientador) foram as minhas maiores motivações para a realização deste trabalho.

A contribuição desta dissertação para o grande sonho humano de colocar homens e máquinas a "entenderem-se" através da fala, vai ser na área do Reconhecimento/Verificação do orador.

Estes sistemas têm-se demonstrado cada vez mais importantes e fundamentais em controlo de acesso a equipamentos, especialmente, para pessoas com mobilidade limitada. São, de igual modo, muito atractivos e úteis para pessoas com mobilidade normal mas que estejam num ambiente de trabalho que requer o uso de equipamento do tipo "mãos livres", como por exemplo, um médico na execução de um exame.

A realização deste trabalho tem como principal objectivo o desenvolvimento de um sistema de reconhecimento de voz, fiável e de simples utilização.

Este sistema deverá:

- Conter a capacidade de detectar voz automaticamente, para facilitar e simplificar o interface com o utilizador;
- Ser capaz de reconhecer o orador, ou palavra e orador, partindo de uma palavra conhecida do sistema, ou seja, palavra previamente treinada com a voz do orador que se pretende reconhecer;
- Ser flexível de modo a poder ser integrado facilmente em outros *softwares* que necessitem de reconhecimento de orador ou de palavras, palavras essas

que podem ser interpretadas como acções. Para conseguir essa flexibilidade este sistema deverá ser implementado num objecto COM para Windows.

Este trabalho engloba ainda uma pequena aplicação com a integração do sistema de reconhecimento. Este *software* vai permitir testar todo o sistema de reconhecimento de Voz, que estará implementado num objecto COM e servirá para apresentação do projecto.

Este trabalho foi desenvolvido no âmbito do projecto TECNOVOZ que a seguir se descreve.

1.1 O que é o TECNOVOZ?

O TECNOVOZ é um projecto em parceria que envolve várias instituições do sistema científico nacional e várias instituições do sistema tecnológico nacional. O principal objectivo é dotar sistemas previamente existentes (e.g. gestão hospitalar) com facilidades de interface por fala.

Com o projecto TECNOVOZ, Portugal tem a oportunidade de contribuir activamente para o desenvolvimento das tecnologias da "fala" nomeadamente para a versão do português Europeu. Deste modo, no âmbito deste projecto as mais recentes tecnologias serão transferidas para o tecido empresarial, dando origem a novos produtos e sistemas, alguns dos quais de carácter inovador.

Para a indústria e sociedade civil em geral, pretende-se demonstrar a existência de uma "poule" tecnológica nacional capaz de industrializar sistemas inovadores ao mesmo ritmo e de igual nível tecnológico do que ocorre no resto do Mundo Ocidental.

Em síntese, são objectivos fundamentais do TECNOVOZ:

- Criar um corpo de conhecimento sobre as tecnologias da fala, com incidência particular na utilização da língua portuguesa, que seja de imediato consubstanciado numa série de aplicações que se traduzem em produtos para

o mercado e que posteriormente constitua uma base de desenvolvimento tecnológico que permita à indústria nacional acompanhar a inovação nestas áreas;

- Despoletar um mercado em potencial, que está actualmente à mercê das empresas internacionais que actuam nesta área e que começam a ameaçar apoderar-se desse mesmo mercado;
- Sendo uma área tecnológica onde o crescimento do volume de negócios associado se mantém elevado num futuro próximo, pretende-se assegurar que o mercado para a língua portuguesa (principalmente o Português Europeu) seja explorado por empresas nacionais.

O TECNOVOZ é um projecto ambicioso, grandioso e complexo, que só poderia ser realizável através da criação de um consórcio. Este foi então criado e pode ser dividido em três partes distintas: a área científica, responsável por desenvolver as tecnologias de fala; a área mais comercial, que vai integrar essas tecnologias de voz em projectos comercializáveis fazendo assim a ponte do TECNOVOZ com o exterior (comunidade); e a entidade consultora que se propôs a gerir o projecto e definir estratégias de trabalho.

A parte científica é composta pelas seguintes entidades do sistema científico nacional:

- INESC-ID – Instituto de Engenharia de Sistemas e Computadores – Investigação e Desenvolvimento
- INESC INOVAÇÃO – Instituto de Novas Tecnologias
- Instituto de Telecomunicações – Pólo de Coimbra
- Universidade do Minho

A parte mais comercial é composta pelas seguintes entidades do sistema tecnológico nacional:

- CPCHS – Companhia Portuguesa de Computadores – Healthcare Solutions, S.A.;
- Anditec – Tecnologias de Reabilitação, Lda.;
- Datelka – Engenharia e Sistemas, Lda.;

- EDISOFT – Empresa de Serviços e Desenvolvimento de *Software*, S.A.;
- Priberam Informática, S.A.;
- Promosoft SIS – *Software* de Sistemas, S.A.;
- Rádio e Televisão de Portugal, S.A.;
- Tecmic – Tecnologias de Microelectrónica, S.A.;

A entidade consultora é empresa MultiSector Norte – Consultores em Estratégia, Tecnologia e Gestão Industrial, Lda..

De seguida é apresentado uma breve síntese dos assuntos abordados nos restantes capítulos.

No capítulo 2, que se intitula "Descrição teórica do sistema", vamos descrever todo o sistema no ponto de vista teórico. Aqui apresentamos todos os módulos constituintes deste sistema e o porquê dessa estrutura de ligações entre eles. Explicamos, igualmente, as funções de cada um dos módulos e toda a teoria que os suporta.

No capítulo 3, vamos explicar a teoria dos HMM's, focando mais a arquitectura de Bakis, que é mais simples e a utilizada neste sistema de reconhecimento de fala. Apresentamos, ainda, um exemplo para completar essa explicação teórica em partes de difícil compreensão.

No capítulo 4, cujo nome é " Implementação do sistema ", vamos proceder ao esclarecimento do sistema a nível implementacional. Nesta parte vamos apresentar a arquitectura do programa e as partes, que se pensou ser, mais importantes e relevantes do *software*.

Por fim, no capítulo 5, onde apresentamos os resultados, as conclusões sobre o sistema e sugestões para complementar e melhorar este sistema de reconhecimento.

Capítulo 2

Descrição teórica do sistema

O sistema desenvolvido neste projecto é um Reconhecedor do orador onde as orações são palavras isoladas (*Isolated Word Recognition*) e o mecanismo de reconhecimento é baseado em HMM's observações contínuas (*Continuous Density Hidden Markov Models*), com segmentação automática e continua das palavras pronunciadas pelo orador dado o início da gravação.

Este capítulo começa por apresentar alguns conhecimentos sobre estatística, que se nos afiguram uma ajuda necessária na compreensão de toda a teoria apresentada neste projecto. De seguida, apresenta-se muito resumidamente outros tipos de abordagens usadas, actualmente, em reconhecimento automático da fala, nomeadamente a abordagem acústico – fonética e a abordagem baseada em inteligência artificial. Quanto à abordagem por reconhecimento de padrões, esta será explicada mais detalhadamente, durante este capítulo e o capítulo seguinte, dado que foi usada na elaboração deste trabalho.

O objectivo primordial deste capítulo é apresentar uma descrição funcional e detalhada de todos os componentes que constituem o Reconhecedor excepto a parte dos modelos de Markov que serão explicados no capítulo 3. Inicialmente, este sistema é apresentado, globalmente, através de um diagrama de blocos sendo depois descrito cada um dos blocos, separadamente. Partes importantes do Reconhecedor com o detector fala/pausa (bloco de processamento de sinal - *Endpoint*) são aqui apresentados com bastante detalhe.

2.1 Conceitos de estatística

O reconhecimento da fala em geral e o reconhecimento do orador em particular são muito fundamentados em conceitos de natureza estatística. Por esse motivo começa-se por facultar uma breve explicação de alguns conceitos no sentido de ajudar o leitor a compreender mais facilmente a teoria exposta e ainda estabelecer a notação usada ao longo desta tese. Os conceitos irão ser descritos separadamente e ordenados por complexidade crescente.

Comecemos então com o conceito de probabilidade.

2.1.1Probabilidade

A Probabilidade define-se como o quociente entre o número de casos favoráveis e o número total de casos possíveis numa experiência aleatória.

$$P(x) = \frac{n^{o}\, casos\ \ favoráveis}{n^{o}\, casos\ \ possíveis} \tag{2.1}$$

2.1.2 Teorema de Bayes

O teorema Bayes postula que a probabilidade directa de uma hipótese *H* condicionada a um corpo de dados *E*, $P(H/E)$, está relacionada com o inverso da probabilidade dos dados *E* condicionados à hipótese *H*, $P(H/E)$ é descrita Matematicamente por:

$$P(H/E) = \frac{P(E/H)\cdot P(H)}{P(E)} = \frac{P(E \cap H)}{P(E)} \tag{2.2}$$

2.1.3 Variância

A variância de uma variável aleatória é um valor não negativo, que dá uma medida de quão dispersas se encontram as observações dessa variável relativamente à média. Quanto maior for a variância, mais dispersas se encontram as

observações em relação à sua média. A variância de uma variável aleatória $x(n)$, que geralmente representa-se por $Var[x(n)]$ ou σ_x^2, é definida como:

$$\sigma_x^2 = Var[x(n)] = \mathrm{E}\left[\left(x(n) - m_x(n)\right)^2\right] \tag{2.3}$$

Onde $\mathrm{E}[.]$ representa a esperança matemática e m_x é média ou valor esperado de $x(n)$, isto é $m_x = \mathrm{E}[x(n)]$.

2.1.4 Covariância

A covariância é uma medida de semelhança entre variações conjuntas de duas variáveis aleatórias, dado um número de observações. Quanto maior a covariância, menor a diferença que existe entre as variações das duas variáveis. Difere da variância na medida em que esta mede apenas o nível variação de uma variável. A covariância entre duas variáveis $x_1(n)$ e $x_2(n)$, que normalmente representa-se por $Cov[x_1(n), x_2(n)]$, é definida como:

$$\begin{aligned} Cov[x_1(n), x_2(n)] &= \mathrm{E}\left[\left(x_2(n) - m_{x_2}(n)\right)\cdot\left(x_1(n) - m_{x_1}(n)\right)\right] \\ &= \mathrm{E}\left[x_1(n)\cdot x_2(n)\right] - m_{x_1}(n)\cdot m_{x_2}(n) \end{aligned} \tag{2.4}$$

Se duas variáveis aleatórias são independentes, então a sua covariância é zero. Isto verifica-se porque se duas variáveis são independentes, então:

$$\mathrm{E}\left[x_1(n)\cdot x_2(n)\right] = m_{x_1}(n)\cdot m_{x_2}(n) \tag{2.5}$$

No entanto, a preposição inversa pode não se verificar, ou seja, se a covariância entre duas variáveis for zero, elas não são necessariamente independentes, porém, designam-se por não correladas.

É de notar também, que a covariância de uma variável aleatória é igual à variância dessa variável, como se demonstra de seguida:

$$\begin{aligned} Cov[x(n), x(n)] = Cov[x(n)]^2 &= \mathrm{E}\left[\left(x(n) - m_x(n)\right)\cdot\left(x(n) - m_x(n)\right)\right] \\ &= \mathrm{E}\left[\left(x(n) - m_x(n)\right)^2\right] \end{aligned} \tag{2.6}$$

2.1.5 Distribuição Normal ou *Gaussiana*

A distribuição Normal ou Gaussiana é descrita por uma função densidade de probabilidade da distribuição normal que é denominada tradicionalmente como "curva em forma de sino". Esta função é completamente determinada por valores numéricos de dois parâmetros, a média μ e a variância σ^2. Pode também ser representada por $p(x) \sim \mathrm{N}(\mu, \sigma^2)$ que é compreendido como " x é distribuído normalmente com média μ e variância σ". Como se pode ver na Figura 2.1 esta distribuição é simétrica sobre a média, o máximo ocorre em $x = \mu$ e a largura do "sino" é proporcional ao desvio padrão σ. Outra característica desta distribuição é ter probabilidades bem conhecidas numericamente definidas por:

$$p(|x-\mu| \le \sigma) = 0.68\,;\ p(|x-\mu| \le 2\sigma) = 0.95\,;\ p(|x-\mu| \le 3\sigma) = 0.997\,.$$

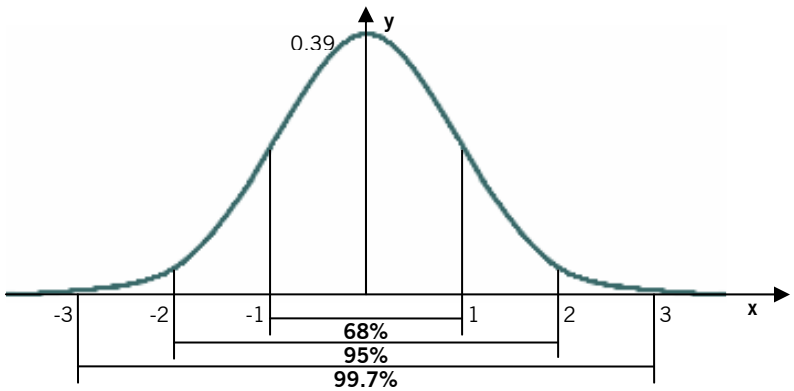


Figura 2.1 – Distribuição normal de média nula e variância unitária com respectivos intervalos de probabilidade.

Um dos resultados mais importantes da teoria da probabilidade é o teorema do limite central. Este teorema afirma que sob diversas condições, a distribuição de qualquer soma de muitas variáveis aleatórias independentes com mesma distribuição de probabilidade tende para uma distribuição designada normal ou *Gaussiana*. Por este motivo, a função densidade de probabilidade da distribuição *Gaussiana* é muito importante e a mais usada, tanto em razões teóricas como práticas. Numa dimensão, é definida por:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{2.7}$$

2.1.6 Mistura Gaussiana

Estes modelos são frequentemente utilizados quando as observações estão concentradas em torno de vários pontos distintos e relativamente distantes (várias médias), tornando a simples *Gaussiana* uma má aproximação para esse tipo de distribuições.

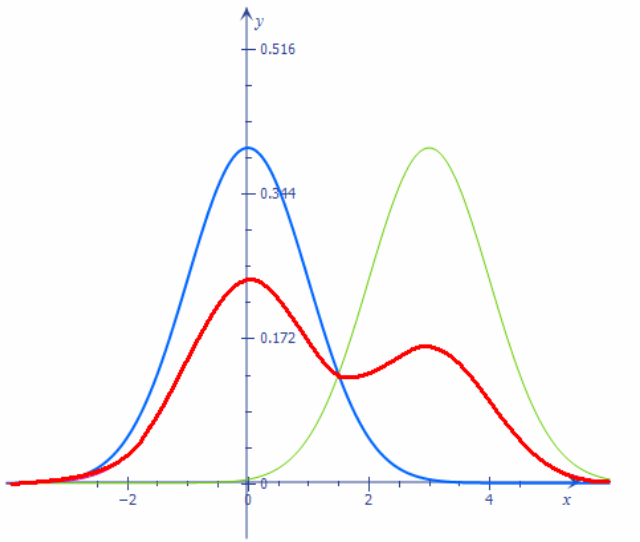


Figura 2.2 – Mistura Gaussiana com duas componentes (Bimodal) e Gaussianas que lhe deram origem.

Uma mistura *Gaussiana* resume-se a um somatório de *N Gaussianas* pesadas, onde *N* define o número de componentes da misturas e o tipo de mistura *Gaussiana*; por exemplo *N*=2 bimodal e *N*> 2 multimodal (ver figura 2.2). Estes tipos de distribuições têm funções densidade de probabilidade que podem ser expressas por:

$$f(x) = \sum_{n=1}^{N} p_n \cdot \mathrm{N}\left(x, \mu_n, \sigma_n^2\right) \tag{2.8}$$

Onde N(.) representa a distribuição *Gaussiana*; μ_n e σ_n^2 a média e a variância respectivamente da $n^{ésima}$ *Gaussiana*; e p_n que define o peso dado à Gaussiana *n*. É de realçar, também, que o somatório de todos os p_n tem que ser igual a 1 para que esta função possa ser realmente uma função densidade a probabilidade, ou seja, o integral dessa função tem que ser unitário o que representa probabilidade de 100%.

2.1.7*Gaussiana* Multidimensional

Uma *Gaussiana* multidimensional é usada normalmente quando as observações de uma dada variável aleatória são multidimensionais ou, dito de outro modo, são observações definidas por várias coordenadas (a Figura 2.3 mostra uma *Gaussiana* bidimensional de observações com duas variáveis).

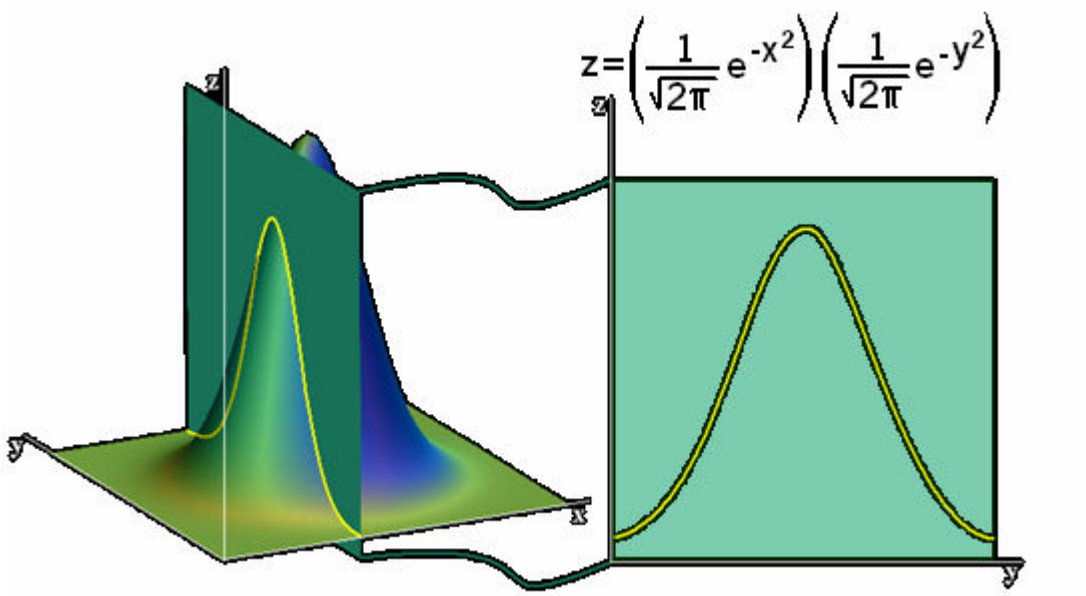


Figura 2.3 – Vista de corte de uma Gaussiana Bidimensional.

Consideremos um vector observação $x = \{x_d\} = \{x_1, x_2, ..., x_D\}$ cujas componentes x_d são variáveis aleatórias com distribuição *Gaussiana*. Se as variáveis aleatórias são i.i.d. (independentes e identicamente distribuídas) então, a correlação entre elas é nula e a função densidade de probabilidade de x é dada pela multiplicação das distribuições marginais das suas componentes, como mostra a equação (2.9).

$$f(x) = \prod_{d=1}^{D} \frac{1}{\sigma_d \sqrt{2\pi}} \cdot e^{\frac{-(x_d - \mu_d)^2}{2\sigma_d^2}} \tag{2.9}$$

Onde $x = \{x_d\}$ representa as coordenadas das várias dimensões, *D* o número de dimensões, μ_d média e σ_d^2 variância da respectiva dimensão *d.*

2.2 Outros tipos de reconhecedores de fala

As abordagens usadas actualmente em reconhecimento automático da fala podem ser agrupadas nas três categorias seguintes:

- A abordagem do reconhecimento de padrões (a utilizada neste projecto),
- A abordagem acústico-fonética, e
- A abordagem da inteligência artificial.

Na abordagem acústico-fonética, tenta-se descodificar o sinal de fala de uma forma sequencial baseando-se não apenas nas observações mas também nas relações conhecidas entre símbolos fonéticos e características acústicas. Apesar da grande variabilidade acústica das unidades fonéticas, quer entre diferentes oradores quer entre fonemas vizinhos para o mesmo orador assume-se que as regras que governam tão grande variabilidade podem ser aprendidas e aplicadas em situações práticas. Porém, por várias razões, a abordagem acústico-fonética não alcançou na prática o sucesso alcançado por métodos alternativos.

A abordagem do reconhecimento da fala por métodos de inteligência artificial é essencialmente uma abordagem híbrida, pois explora ideias e conceitos da abordagem acústico-fonética e do reconhecimento de padrões. A ideia é explorar a mecanização do processo de reconhecimento tal como se aplica a inteligência na visualização, análise e finalmente na tomada de decisão baseada nas observações acústicas. Nesta classe de métodos estão os sistemas de segmentação e etiquetagem automática que usam geralmente mais que a informação acústica usada por métodos acustico-fonéticos puros. Usualmente os primeiros necessitam além da informação acústica, de informação fonémica, léxica, sintáctica e semântica. Tem sido proposto o uso de redes neuronais para aprendizagem das relações entre eventos fonéticos e outros (acústicos, lexicais, sintácticos, semânticos, etc.).

Frequentemente a inteligência artificial é associada ao uso de Redes Neuronais (NNs). Porém, o uso de NNs representa essencialmente uma abordagem estrutural separada, um novo classificador para aplicações de reconhecimento da

fala. As NNs constituem uma arquitectura ao nível implementacional com grande potencial para beneficiar as três abordagens clássicas.

2.3 O sistema

A figura 2.4 representa a arquitectura do sistema de reconhecimento do orador integrada com as funcionalidades de treino e reconhecimento de padrões de fala. No início, o áudio é capturado a partir de um microfone, depois seguem-se os blocos *Endpoint*, Pré-ênfase, extracção de características, Biblioteca de padrões e por fim o bloco Reconhecedor. As secções seguintes apresentam uma descrição funcional detalhada de cada bloco constituinte do Reconhecedor.

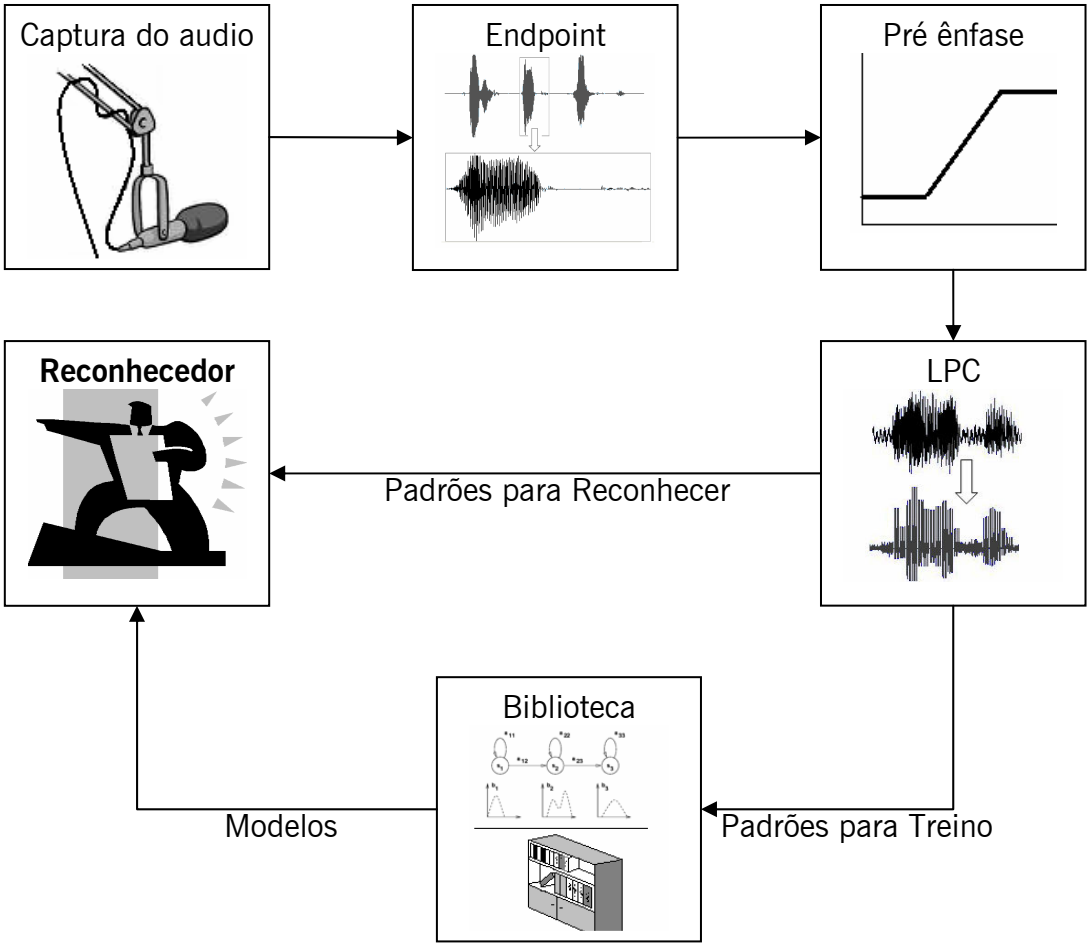


Figura 2.4 – Diagrama de Blocos do sistema de reconhecimento do orador.

2.4 Captura do Áudio

Este bloco representa a gravação do sinal para posteriormente ser processado nos blocos seguintes.

O áudio é capturado através de um microfone e guardado na memória do PC. A frequência de amostragem pode ser qualquer sendo usual a de 8000 amostras por segundo (8 KHz). Também típica é a resolução em amplitude, que é normalmente de 16 bits, permitindo diferenciar 65.536 amplitudes diferentes num intervalo de [- 32.768; 32.768].

Segundo o teorema de amostragem, a frequência de amostragem tem que ser o dobro da frequência máxima do sinal para que não ocorra aliasing, ou seja, para que não haja perda de informação sobre o sinal que se quer guardar. A maior parte da informação relevante para a identificação da fala humana, mediante testes de percepção auditiva, encontra-se entre 60 e 3400 hertz o que significa que para uma frequência de amostragem de 8 KHz é possível conter todas a informação até 4Khz (4 KHz> 3400 Hz) e, com isso, é salvaguardada toda a informação considerada mais importante da voz humana.

2.5 *Endpoint*

Neste bloco é apresentado um método de pré – processamento do sinal de voz cujo objectivo é retirar as partes do sinal de áudio que não tem informação sobre o orador. O detector fala/pausa ou *Endpoint* é um algoritmo, baseado na modelação do ruído de fundo inicial, que dará indicação ao sistema do que realmente poderá ser voz, isolando assim os momentos de fala dos de não-fala do locutor, durante a fase de gravação do som.

O problema da eficiência do detector fala/pausa e as suas implicações no desempenho do sistema de reconhecimento é conhecido desde a década de 80 e reconhecido deste então como um problema complexo. Para separar a fala da não-fala já foram abordadas várias formas e criadas diversas técnicas das quais se des-

tacam a técnica STE (Short Time Energy) e a técnica ZCR (Zeros Crossing Rate) [2].

Todavia, as técnicas ZCR e STE não são suficientemente robustas e eficientes na detecção de fala especialmente quando se pretende taxas de erro baixas em condições não isentas de ruído. Com o intuito de minorar este problema, desenvolveu-se um novo método baseado na união dos pontos fortes destas técnicas mais convencionais com uma técnica mais recente [1], no sentido de resultar um método com maior robustez e eficiência.

2.5.1 Método

Este detector de fala/pausa (*Endpoint*) foi implementado com base no conhecimento das propriedades básicas da voz humana. O sinal gerado pela nossa voz tem uma variação relativamente lenta no domínio do tempo, e quando analisado em períodos de tempo suficientemente curtos (entre 5 e 100 milissegundos), as suas características são aproximadamente estacionárias. Contudo, para períodos relativamente longos de tempo (na ordem dos 200 ms ou mais) as características do sinal já não são estacionárias produzindo assim os diferentes sons. Considera-se que o conhecimento das condições acústicas (ruído de fundo) pode ajudar a decidir entre ruído e fala, uma vez que o início da oração trará certamente alterações na estrutura do sinal. O problema é que estas alterações podem dever-se apenas a alterações ambientais levando a erros na segmentação. A abordagem seguida supõe que os primeiros 200 milissegundos ou mais (1600 amostras a taxa de amostragem que é utilizada, 8k amostras/segundo) correspondem a silêncio ou a ruído de fundo dado que frequentemente o microfone já está ligado quando a pessoa começa a falar. É nesse momento (nos primeiros 200 milissegundos) que se vai analisar o ruído de fundo. Neste método é considerado, sem perda de generalidade dado o teor do limite central, que o ruído de fundo assume uma distribuição Gaussiana. Esta técnica para além de permitir detectar pausas inter-palavra, também é capaz de detectar pausas intra-palavra.

Nesta implementação decidiu-se pelo uso de janelas não sobrepostas de 20 milissegundos para a análise do sinal. Este valor foi definido através de um com-

promisso entre o tempo que seria aceitável para analisar o ZCR e o STE e o tempo para o qual o sinal continua com as suas características estacionárias.

Todos os valores máximos e mínimos para os parâmetros utilizados neste método foram atribuídos após a análise de muitas palavras pronunciadas por diferentes oradores.

Esta técnica permite também alterar algumas variáveis temporais no sentido de se adaptar melhor à situação e ao local onde este Reconhecedor vai trabalhar. Dois exemplos dessas variáveis são: o tempo de silêncio após a fala que determina o fim da gravação (se o reconhecimento for efectuado, apenas, com uma palavra este tempo pode ser curto porque não é necessário esperar por fala adicional) e o tempo inicial que espera pelo começo da voz do locutor (se não houver fala num tempo predefinido a gravação pára automaticamente).

2.4.2 Algoritmo

A primeira grande vantagem do algoritmo, representado na figura 2.5, é a possibilidade de ser executado paralelamente à gravação. Com isso, minimiza o tempo de espera por parte do utilizador pelo fim da sua execução e torna possível o controlo do fim de gravação com base no tempo de não-fala.

Este algoritmo está dividido em duas fases. A primeira fase consiste na análise do ruído de *background*. Nos primeiros 200 milissegundos de gravação (ruído) é estudada a estatística do sinal, mais propriamente a obtenção da sua média e do seu desvio padrão. A segunda fase resume-se à identificação de momentos de fala e não-fala. Para o reconhecimento destes fragmentos de fala e não-fala o sinal é dividido em janelas de 20 milissegundos, que são depois analisadas tendo em conta a estatística do ruído, a energia do sinal numa determinada janela (STE) e a percentagem de passagens por zero do sinal (ZCR). A partir destas análises são diferenciadas janelas consideradas “voz” das restantes e agrupadas em “sequências voz”, isto é, momentos do sinal capturado que contêm apenas janelas “voz”. Este procedimento está ilustrado na figura 2.5 (através do fluxograma do *Endpoint*) e na figura 2.6 (através do esquema da real análise que se faz ao sinal de áudio).

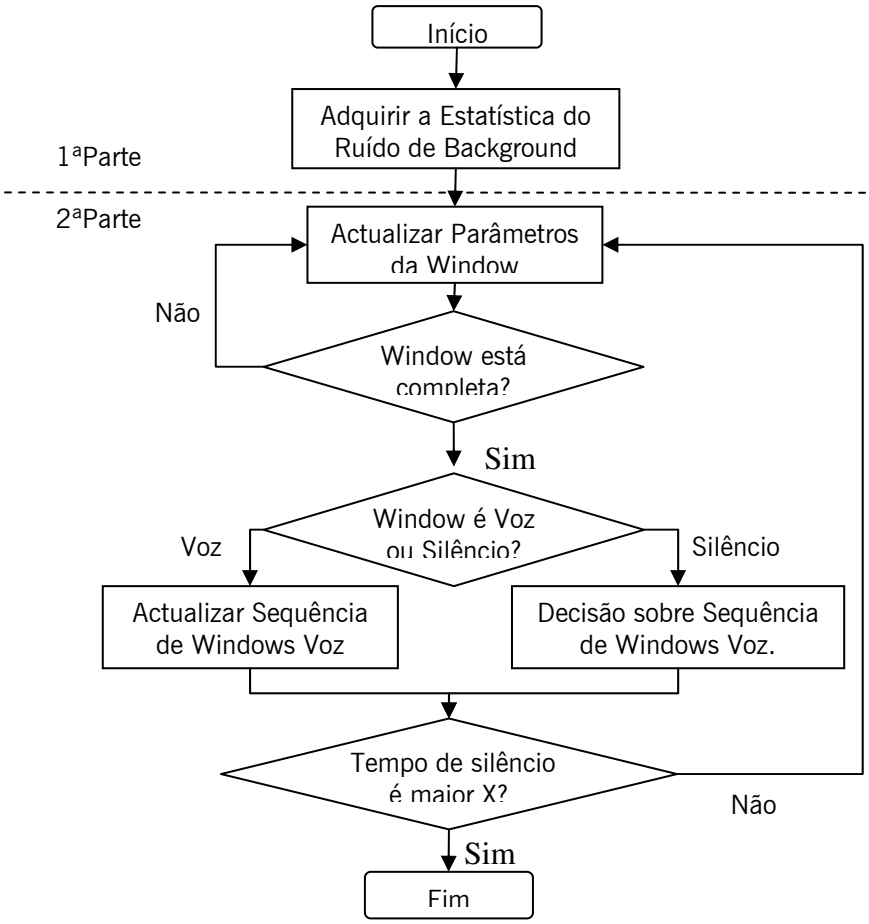
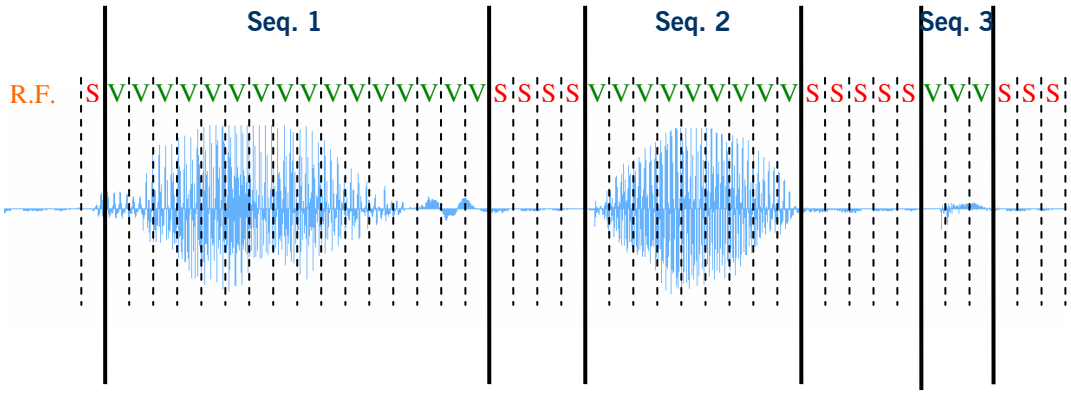


Figura 2.5 – Fluxograma do Endpoint.

As "sequências voz" são posteriormente avaliadas através de medidas baseadas em STE, ZCR e na energia do sinal em toda a sequência. Dependendo dessa avaliação essas sequências poderão ser descartadas, consideradas fala, ou até unir duas sequências consecutivas. Como no caso de palavras com longas pausas intermédias, como é exemplo a palavra "eight" que é constituída por 2 sons separados por uma pausa longa "ei-ght".



R.F. captura da estatística do ruído de fundo; Seq. Sequência Voz; V janela Voz; S janela Silêncio

Figura 2.6 – Análise do sinal através de janelas.

Posto isto, falta simplesmente dar resposta às seguintes questões:

- Quando é que uma janela é Voz ou Silêncio?
- Quando é que uma "Sequência Voz" é Fala?
- Quando são unidas duas sequências?

A janela é Voz ou Silêncio?

Para que uma janela seja considerada "voz" tem que passar em dois testes:

- Em primeiro lugar, a janela tem que conter um número de amostras maioritário que não pertençam a distribuição do ruído com uma probabilidade de 99.7% ou $Esr > 0$ através da equação (2.10). Nesta tese a expressão (2.10) denomina-se por energia sinal /ruído.

$$Esr = \sum_{i=0}^{W} \left(\frac{|x_i - \mu_r|}{\sigma_r} > 3 \right) \tag{2.10}$$

  Onde $\frac{|x_i - \mu_r|}{\sigma_r} > 3$ representa 1 ou -1 se a condição for verdadeira ou falsa respectivamente, *i* é o índice da amostra e W o tamanho da janela.

  Esta expressão indica exactamente a diferença que existe entre pontos que se encontram fora da distribuição do ruído e pontos que se encontram dentro com probabilidade alta (99.7%).

- Em segundo lugar, a janela tem que conter um ZCR diferente de zero, o que implica a não validação como "voz" de janelas com pouca vibração. Exemplos disso são os momentos da nossa respiração que também são capturados na fase de gravação. A expressão do ZCR é dada por:

$$ZRC = \frac{n^{o}\ de\ \ passagens\ \ por\ zero}{n^{o}\ de\ \ pontos\ considerados - 1} \tag{2.11}$$

A "Sequência Voz" é Fala?

Para resolução deste problema são utilizadas as seguintes medidas no contexto da sequência: a energia sinal/ruído *Esr, o ZCR,* o tamanho e um outro tipo de medição que indica a variação média de energia sinal/ruído. A equação que define a variação média de energia é dada por:

$$Ve_M = \frac{1}{S}\sum_{j=1}^{s}\left|Esr_{j+1} - Esr_j\right| \qquad (2.12)$$

Onde *S* é o tamanho da sequência e *j* é o índice da janela na sequência.

Tendo em conta as análises que foram feitas a várias sequências que eram efectivamente Voz, chegou-se a um conjunto de condições que têm que ser verificadas para que uma determinada "sequência voz" seja considerada fala. Estas condições são:

- O ZCR da sequência tem que estar entre 1.5% e 50%, e
- A variação média de Energia tem que ser maior do que 7% e menor do que 60%, percentagens relacionadas com o valor máximo (dimensão de uma janela).

Para além de serem consideradas fala a estas "sequências voz" é atribuído um classificador dependendo de outras condições. Existem 3 classificadores para "sequências voz" que são: "completa", final" ou "inicial".

- Completa – sequências que são maiores do que 470 milissegundos;
- Final – sequências que têm energia sinal/ruído inferior a 30% do valor máximo e de tamanho menor do que 700 milissegundos;
- Inicial – sequências que não são completas nem finais.

Todos estes valores percentuais foram obtidos por medições puramente estatísticas e por isso com validade eventualmente dependente das condições ambientais em que foram obtidas. Essas condições eram levemente ruidosas.

Unir duas sequências?

Existem várias palavras em que durante a sua pronunciação são encontrados momentos em que a energia do sinal atinge valores muito baixos e onde o ruído se evidencia em relação à voz. Um exemplo desse caso é a palavra ”eight” ilustrada na figura 2.7.

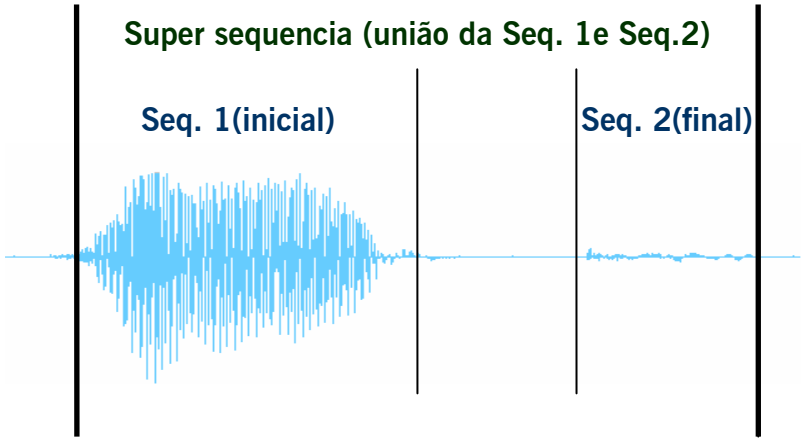


Figura 2.7 – União de sequências.

Quando isso acontece o algoritmo, numa primeira fase, separa essas palavras em duas “sequências voz” distintas. Numa segunda fase, essas sequências são avaliadas por um conjunto de condições e caso sejam aprovadas elas serão unidas formando uma super sequência contendo a palavra completa, como mostra a figura 2.7. As condições que têm que ser verificadas para a união de duas “sequências voz” são as seguintes:

- A sequência anterior tem que ser classificada como “inicial” e a posterior como “final”.
- A sequência “inicial” tem que ter uma energia real superior à sequência “final”.
- E a variação média de energia Ve_M tem que ter um valor maior na “final” do que na “inicial”.

2.4.3 Implicações do ruído elevado no algoritmo

Com o aumento do nível de ruído de fundo a função densidade de probabilidade que modela o mesmo sofre também um acréscimo na sua largura. Uma vez

que nesta tese consideramos que o ruído de fundo segue uma distribuição Normal podemos dizer que desvio padrão da distribuição é proporcional à quantidade de ruído.

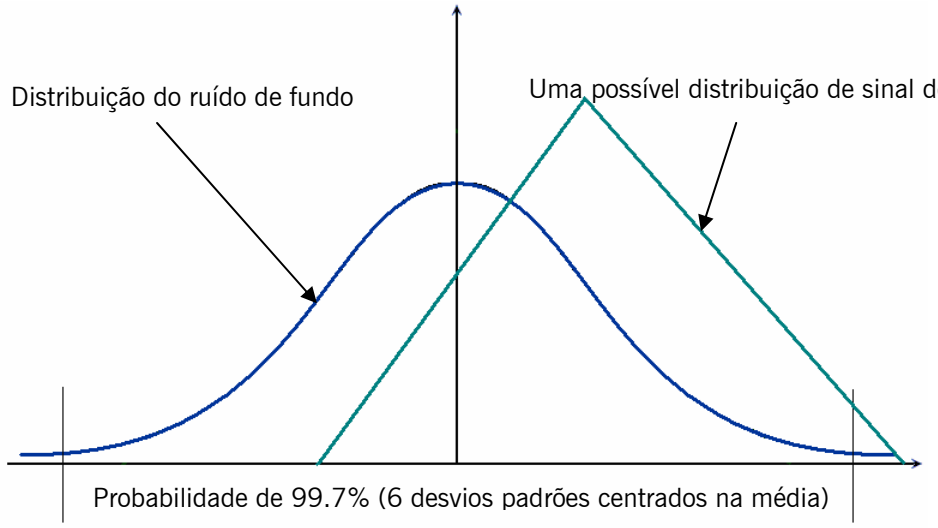


Figura 2.8 – Distribuição do ruído e do sinal quando o ruído é elevado.

Como se verifica na figura 2.8, por vezes esses desvios padrões são indesejavelmente grandes o que faz com que o intervalo definido pela condição $\frac{|x_w - \mu_r|}{\sigma_r} > 3$ (99.7% de probabilidade de ser ruído) contenha quase toda a distribuição do sinal. Nestas condições não é possível, com grande grau de confiança, inferir acerca da natureza da amostra (fala ou não fala).

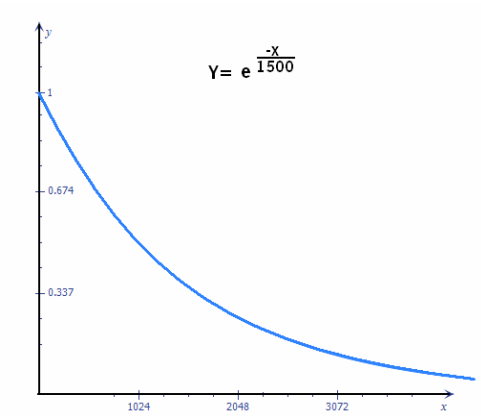


Figura 2.9 – Função que define o factor sinal ruído.

A solução encontrada para resolução do problema do aumento do ruído consiste em diminuir gradualmente a probabilidade de ser ruído no teste de uma determinada amostra (diminuição do grau de confiança). Para diminuir gradual-

mente essa probabilidade utilizou-se um factor que é multiplicado à condição de ser ruído da seguinte forma:

$$\frac{|x_w - \mu_r|}{\sigma_r} > 3 * \mathrm{F}sr \qquad (2.13)$$

Onde $\mathrm{F}sr$ é o factor de correcção de probabilidade.

A expressão deste factor que é apresentada de seguida, foi adquirida através de uma série de valores obtidos que maximizavam o desempenho do detector fala/pausa numa gama estrema de condições de ruído.

$$Fsr = e^{\frac{-x}{1500}} \qquad \text{(figura 2.9)} \qquad (2.14)$$

Onde *x* é o desvio padrão do ruído de fundo.

2.5 Pré-ênfase

O Pré-ênfase é usado para aumentar, numa certa gama de frequências usualmente altas, a amplitude de certas frequências em relação à amplitude de outras frequências usualmente baixas. Com este procedimento pretende-se equalizar o efeito labial. A função usada no âmbito desta tese é a seguinte:

$$y[n] = x[n] + 0.97x[n-1] \qquad (2.15)$$

2.6 Extracção de características

Neste bloco é usado o *Linear Predictive Coding* (LPC) para extracção das características do sinal de fala. O LPC é uma das ferramentas mais poderosas usada, principalmente, em processamento de sinais de áudio, mais propriamente, de fala. É um facto conhecido que os parâmetros LPC não são muito imunes ao ruído, no entanto, apresentam-se muito eficazes para fala limpa.

A extracção das características da fala consiste em retirar do sinal um conjunto de coeficientes ou parâmetros específicos desse sinal de fala. Para fazer isso, o método LPC propõe que o sinal deve ser analisado por segmentos e em cada segmento separadamente encontrar o conjunto de coeficientes que identificam as propriedades vocálicas do orador.

Neste sistema, esses segmentos foram extraídos do sinal com sobreposição (ver figura 10) e foram ainda passados por uma janela de *hamming* para reduzir as descontinuidades nos pontos terminais da sequência.

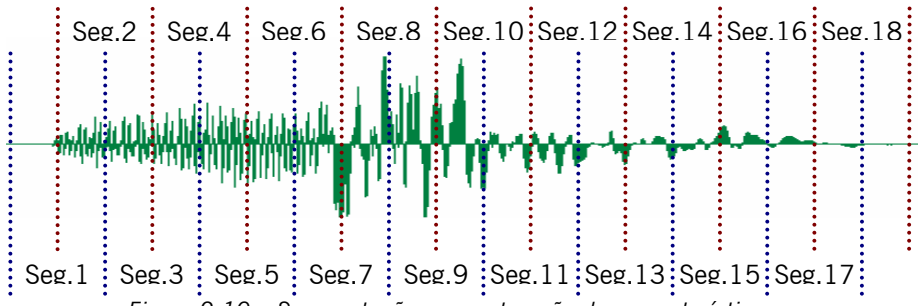


Figura 2.10 – Segmentação para extracção das características.

O cálculo dos coeficientes que melhor modelam um determinado segmento no contexto deste método é feito em duas etapas:

- A primeira etapa assume que a presente amostra do sinal de fala é predita através de uma combinação linear das *M* amostras anteriores através da expressão:

$$\tilde{x}(n) = a_1 x(n-1) + a_2 x(n-2) + \ldots + a_M x(n-m) = \\ = \sum_{i=0}^{M} a_i x(n-i) + Ge_n \qquad (2.16)$$

  Onde $\tilde{x}(n)$ é a predição de $x(n)$, $x(n-i)$ são as amostras passadas e $\{a_i\}$ os coeficientes a encontrar, que são designados por coeficientes de predição.

- A segunda etapa minimiza o erro quadrático entre a amostra real $x(n)$ e a amostra predita $\tilde{x}(n)$ (calculada em (2.16)). O erro quadrático é dado pela seguinte expressão:

$$\varepsilon^2(n) = \left(x(n) - \sum_{i=1}^{M} a_i x(n-i) + Ge_n \right)^2 \qquad (2.17)$$

2.7 Biblioteca

Neste módulo é efectuado o treino dos oradores, dependente do texto, através do método de aprendizagem de modelos apresentado no capítulo 3. É também aqui na biblioteca que serão guardados esses modelos treinados que posteriormente entrarão no reconhecimento na fase de comparação de modelos.

O treino de palavras supõe que as propriedades acústicas da fala podem ser adequadamente caracterizadas por um processo de treino, se um número suficiente de padrões a caracterizar fizer parte do conjunto de treino. A este tipo de caracterização, via treino, chama-se classificação de padrões, porque as propriedades acústicas da fala mais fiáveis e representativas existentes em todos os exemplares de treino para um qualquer padrão são "aprendidas" e guardadas sobre a forma de um modelo que representa o padrão.

Durante a fase de treino o sistema "aprende" os padrões de referência que representam os diferentes sons (palavras e fonemas) que constituem o vocabulário da aplicação. A aprendizagem de cada referência (palavra e respectivo orador) é feita com base em algumas repetições dessa mesma referência, sendo cada referência armazenada sob a forma de modelos que caracterizam as propriedades dos padrões. A fase de aprendizagem necessita de algoritmos de aprendizagem eficientes para munir o sistema de padrões de referência verdadeiramente representativos.

A fase de treino é sempre a mais morosa. Para criar um bom modelo de uma determinada palavra e respectivo orador é preciso: várias recolhas da mesma palavra com o mesmo orador, produção dos coeficientes LPC das respectivas palavras e, por último, gerar o modelo através do algoritmo de treino (capítulo 3). O tempo de treino pode ser amenizado se partirmos com um modelo inicial dessa palavra, independente do orador, ou com um modelo já existente desse orador para essa palavra.

2.8 Reconhecedor

O reconhecimento é efectuado por comparação directa do padrão de entrada desconhecido (palavra pronunciada pelo orador a reconhecer) com os modelos dos padrões constituídos e guardados na fase de treino. O padrão desconhecido é classificado, usando uma medida de semelhança entre padrões de acordo com o grau de semelhança que apresenta relativamente aos padrões de referência (obtidos durante a fase de treino). Essa medida probabilística de semelhança será descrita no capítulo 3. O resultado final do Reconhecedor indica qual é a palavra e o orador mais provável através do conhecimento do modelo que obteve o valor mais elevado durante essas comparações directas.

Como foi dito, o reconhecimento é realizado por comparação directa entre o padrão de entrada e todos os modelos existentes na biblioteca, por esse motivo a fase de reconhecimento pode ou não ser demorada, dependendo da quantidade de modelos existentes na base de dados (biblioteca) e da opção de detectar só orador ou orador e palavra.

Capítulo3

Modelos de Markov Ocultos

Esta secção apresenta de forma superficial a teoria dos modelos de Markov ocultos. Juntamente à elucidação deste método estatístico apresenta-se um exemplo demonstrativo das partes consideradas mais importantes desta técnica que, porventura, poderão ser as mais difíceis de perceber. A inserção deste exemplo tem como principal objectivo facilitar a compreensão de uma técnica relativamente complexa, especialmente para leitores que não disponham de bases sólidas em estatística ou que não tenham qualquer tipo de saber sobre estes modelos. Todos os conhecimentos ou noções apresentados neste capítulo encontram-se analisados detalhadamente em [2] e [3].

Os modelos de Markov ocultos, mais conhecidos por HMM's (Hidden Markov Models), são uma técnica baseada em estatística aplicada à modelação de sinais não-estacionários. Sinais aleatórios não estacionários são aqueles cujos parâmetros estatísticos se alteram ao longo do tempo (os sinais de fala são exemplo disso, por maior número de capturas da mesma palavras os sinais serão sempre diferentes um dos outros). Todos os modelos estatísticos, sendo que estes também não constituem excepção, necessitam de estimativas dos seus parâmetros, normalmente obtidas a partir de realizações de processos aleatórios.

Estes modelos são actualmente dominantes em reconhecimento de fala, pois apresentam uma estrutura bastante flexível, composta por estados e transições entre eles, o que os torna bastante imunes à variação de velocidade de pronunciação da fala e adaptam-se bastante bem à semântica das palavras.

Os HMM's são modelos que pressupõem a existência de dois processos estocásticos independentes. Com isto, a obtenção da saída destes modelos depende do resultado do processo aleatório, dito escondido ou não observável (sequência de estados) e de um outro observável (conjunto de observações a testar). Um processo estocástico pode ser definido como um conjunto de variáveis aleatórias, indexadas normalmente a unidades de tempo ou espaço, que possuem o mesmo espaço amostral da distribuição de probabilidade conjunta. A arquitectura do processo não observável consiste num conjunto de estados ligados por transições, estando estes organizados de forma a criar a melhor estrutura de adaptação ao pretendido, neste caso, à modelação da não-estacionaridade associada à evolução temporal do sinal de fala.

No processo oculto a transição entre estados é caracterizada por:

- Uma Probabilidade de Transição, que define a probabilidade de acontecer uma transição de um estado para outro ou até para ele próprio. Chamou – se $a_{i,j}$ à probabilidade de transição entre o estado *i* e o estado *j*, ou mais genericamente, associando ao domínio tempo, por a_{s_{t-1},s_t} probabilidade de transição do estado no instante *t-1* para o estado no instante *t*.

No processo observável a transição entre estados é representada por:

- Uma função densidade de probabilidade de saída**,** que apresenta a probabilidade condicional de se observar um conjunto de características da fala quando a transição tem lugar. A função densidade de probabilidade mais utilizada é a Mistura Gaussiana Multidimensional, que vai depender da dimensão da observação, ou seja, ao número de coeficientes que irão representar cada segmento de fala. Denominou-se por $f(y_t / s_{t-1}, s_t)$ função densidade de probabilidade do vector de observações na transição efectuada pelo modelo do instante *t-1* para o instante *t* e por $f(y_t / s_{t-1}, s_t, m_t)$ a densidade de probabilidade associada à $m^{\text{ésima}}$ componente da mistura pertencente à transição do estado no instante *t-1* para o instante *t* definida por $f(y_t / s_{t-1}, s_t)$.

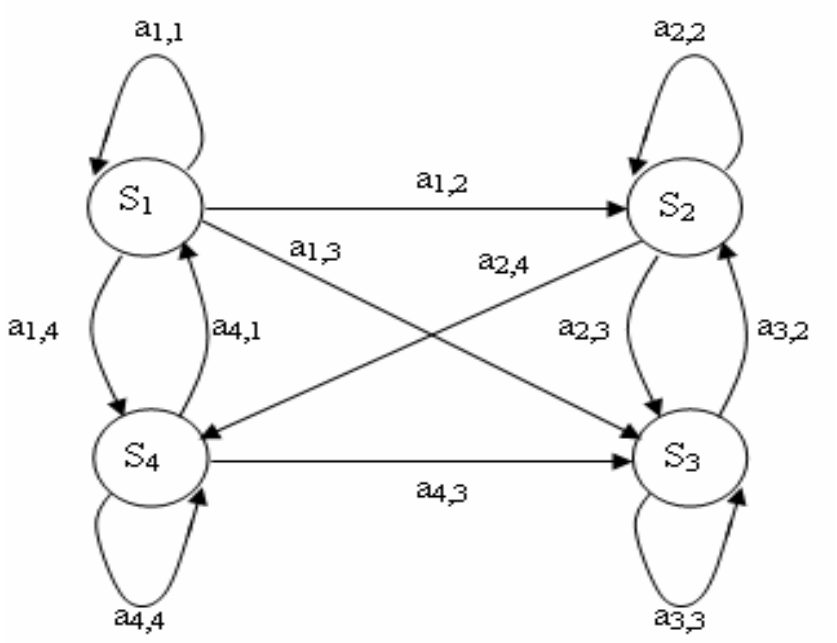


Figura 3.1 – HMM com 4 estados (S1 a S4).

A figura 3.1 mostra um modelo de Markov composto por 4 estados, onde algumas transições não são permitidas (por exemplo, a transição do estado *S2* para o estado *S1*). Nos modelos de Markov discretos, o caminho aleatório seguido, através dos estados do modelo gerado pela máquina de estados estocástica, é conhecido por cadeia de Markov. Se nessa cadeia todas as transições para um determinado estado dependerem apenas do estado anterior, isto é, não dependerem de estados anteriores superiores a um instante de tempo, então diz-se que essa cadeia de Markov é de primeira ordem. Os HMM´s frequentemente utilizados em reconhecimento de fala, usados também nesta dissertação, são os HMM's de *Bakis*. Estes têm uma arquitectura tipo que os caracteriza e geram sempre cadeias de Markov de primeira ordem, como ilustra a figura 3.2. A justificação para a utilização HMM's *Bakis* reside no facto de uma determinada palavra ser, invariavelmente, constituída pela mesma sequência de fonemas.

Dada a arquitectura e os dois processos estocásticos inerentes à teoria dos HMM's, importa agora verificar que estruturas de dados são necessárias para descrever qualquer modelo de Markov. Os elementos básicos de um HMM são: a matriz transição de estados $A = \{a_{i,j}\}$; o vector probabilidade de estado inicial $\pi = \{\pi_i\}$ e a função densidade probabilidade de saída *B* em cada transição. Esta última é a mais complexa, pois é composta por dois vectores (p_{s_{t-1},s_t} e $\mu_{s_{t-1},s_t,m}$) e

uma matriz ($\varepsilon_{s_{t-1},s_t,m}$) para cada transição. O vector p_{s_{t-1},s_t}, cujo tamanho é igual ao número de componentes da mistura, contém os coeficientes da Mistura Gaussiana e o vector $\mu_{s_{t-1},s_t,m}$, cujo tamanho é igual à grandeza do vector observação, contém as médias das variáveis aleatórias que compõem o vector observação. A matriz $\varepsilon_{s_{t-1},s_t,m}$ é quadrada e frequentemente diagonal contendo as covariâncias das variáveis aleatórias que compõem o vector observação. Para compactação de notação, toda estrutura de dados do modelo vai ser representada por uma única variável designada por $\lambda = \{A, B, \pi\}$.

Vejamos a apresentação dos aspectos e definições mais relevantes dos HMM´s conjuntamente com um pequeno exemplo, cujo objectivo é fornecer uma explicação mais detalhada sobre o mecanismo associado a estes modelos.

Exemplo

Consideremos o modelo de Markov para reconhecimento de fala, constituído por 3 estados e por 5 transições organizados como nos mostra a figura 3.2. Neste exemplo, começamos por apresentar a estrutura de dados necessária para descrever o modelo, seguindo as seguintes condições: assumimos que intervalos de 300 milissegundos de fala podem ser, irrealistamente, comprimidos em dois coeficientes através do algoritmo de Codificação Preditiva Linear (LPC-Linear Predictive Coding); supomos que esses dois coeficientes são independentes e identicamente distribuídos; impomos que todas as cadeias de Markov comecem no estado 1 e que sejam de primeira ordem; e adoptamos como função densidade probabilidade de saída para cada estado uma Mistura Gaussiana ou bidimensional com três componentes de mistura.

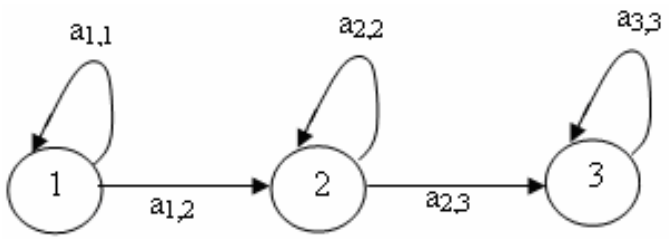


Figura 3.2 – HMM Bakis com 3 estados.

Apresentamos então de seguida os parâmetros que constituem o modelo.

- **Matriz probabilidade de transição de estados** $A=\{a_{i,j}\}$.

Definimos por $a_{i,j}$ a probabilidade de transição do estado *i* para o estado *j*. A matriz probabilidade de transição de estados *A* é uma matriz que contém de forma ordenada estes parâmetros, respectivamente, na linha *i* e coluna *j*.

$$A=\begin{Bmatrix} a_{1,1} & a_{1,2} & 0 \\ 0 & a_{2,2} & a_{2,3} \\ 0 & 0 & a_{3,3} \end{Bmatrix} \tag{3.1}$$

As posições da matriz (3.1) com valores nulos indicam as transições que não podem ser efectuadas ou transições inexistentes. Uma propriedade desta matriz é que o somatório de cada linha tem que ser igual a 1, ou seja, em cada instante a transição efectua-se de facto.

- **Vector probabilidade de estado inicial** $\pi=\{\pi_i\}$.

O vector de probabilidade de estado inicial é composto pelas probabilidades do modelo se iniciar em cada estado. A soma dos valores é unitária, o que significa que o modelo arranca mesmo.

$$\pi=\{1 \quad 0 \quad 0\} \tag{3.2}$$

Este vector, que neste exemplo é definido pela equação (3.2), impõe que todas as cadeias de Markov comecem no estado 1. Como se pode verificar só na posição respeitante a este estado existe probabilidade inicial, tornando assim impossíveis cadeias começadas em outros estados. Em aplicações de fala esta restrição faz sentido dado que, uma palavra começa sempre pelo mesmo fonema.

- **Estrutura de dados para a função densidade probabilidade de saída *B* em cada transição.**

A figura 3.3 apresenta uma possível função densidade probabilidade inerente a uma determinada transição do modelo, onde X representa um dos coeficientes, Y o outro e Z a densidade de probabilidade. A todas as transições do modelo, neste exemplo, estão ligadas funções densidade de probabilidade do tipo Mistura Gaussiana bidimensional com três componentes de misturas. Bidimensional porque as observações ou as variáveis de entrada têm duas dimensões.

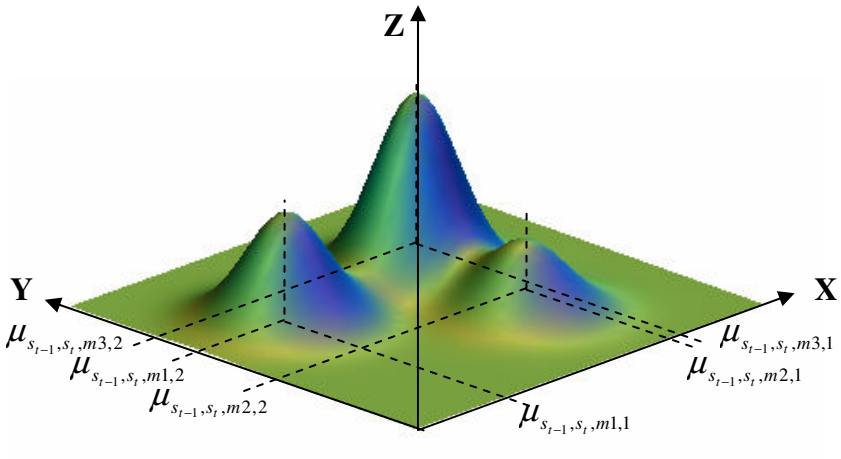


Figura 3.3 – Mistura Gaussiana bidimensional com 3 componentes de mistura.

O número de dimensões neste modelo representa o número de coeficientes de compressão da fala. Escolheu-se, neste caso, gaussianas com duas misturas para tornar o exemplo mais simples. A escolha do número de componentes da mistura constitui um compromisso entre simplicidade e necessidade. Quanto maior o número de gaussianas, melhor o modelo se adapta aos dados, no entanto, o aumento exagerado do número de componentes implica também mais requisitos ao nível de capacidade de cálculo requerida, tendendo a tornar o sistema lento.

De seguida, é apresentada a estrutura de dados normalmente usada para guardar estas funções densidade de probabilidade associadas a cada transição.

- **Vector de coeficientes da Mistura Gaussiana na transição** S_{t-1}, S_t

Este vector, cuja dimensão é igual ao número de componentes na mistura, neste caso três, é responsável por tornar a Mistura Gaussiana bidimensional numa função densidade probabilidade, ou seja, os coeficientes que estão neste vector são multiplicados pelas gaussianas que compõe esta mistura de modo que a soma seja de facto uma função densidade de probabilidade, ou seja, o integral desde $-\infty$ a $+\infty$ seja unitário. Neste exemplo, este vector é definido por:

$$p_{s_{t-1},s_t} = \left\{ p_{s_{t-1},s_t,1} \quad p_{s_{t-1},s_t,2} \quad p_{s_{t-1},s_t,3} \right\} \tag{3.3}$$

- **Vector de médias na transição** S_{t-1}, S_t **dada a mistura *m*.**

Este vector, de tamanho igual à dimensão das observações, tem as coordenadas da média para uma determinada Gaussiana dada a transição e a componente da mistura (ver figura 12). Neste exemplo, este vector é definido por:

$$\mu_{s_{t-1},s_t,m} = \left\{ \mu_{s_{t-1},s_t,m,1} \quad \mu_{s_{t-1},s_t,m,2} \right\} \tag{3.4}$$

Onde *m* representa a componente da mistura.

- **Matriz de covariâncias na transição** S_{t-1}, S_t **dada a mistura *m*.**

Esta matriz, de dimensão 2x2, contém as covariâncias existentes entre dimensões, ou seja, entre os coeficientes de predição da fala, dada a transição e a mistura. Como referido, assumimos que neste exemplo, os coeficientes são independentes e identicamente distribuídos. Nestas circunstâncias a matriz covariância é diagonal sendo os seus elementos não nulos iguais à variância de cada componente do vector observação.

$$\varepsilon_{s_{t-1},s_t,m} = \begin{Bmatrix} \sigma^2_{s_{t-1},s_t,m,1} & 0 \\ 0 & \sigma^2_{s_{t-1},s_t,m,2} \end{Bmatrix} \tag{3.5}$$

Dada a definição de modelo de Markov, interessa resolver os dois seguintes problemas:

- O Problema da Descodificação: dado um modelo e uma sucessão de observações, qual é a probabilidade de as observações terem sido geradas pelo modelo? A solução para este problema pode ser encontrada usando o algoritmo forward-backward [2].

- O Problema da Aprendizagem: dado um modelo e uma sucessão de observações, quais deveriam ser os parâmetros do modelo, de modo a tornar máxima a probabilidade de gerar tal sucessão? A solução para este problema pode ser encontrada usando o algoritmo de *Baum-Welch* [2].

3.1 Problema da Descodificação

Este problema de descodificação pode ser visto também como um problema de "Compatibilidade", isto é, dada uma sequência de observações e um determinado modelo é necessário descobrir uma medida (neste caso uma verosimilhança) que traduza o nível de compatibilidade entre eles. De seguida, mostra-se como é possível encontrar essa medida aplicando os conceitos a modelos de Markov de *Bakis* ou esquerda para a direita de 6 estados e de 5 misturas, cuja estrutura é mostrada na figura 3.4. Tipologia de modelo que é utilizada no Reconhecedor de fala descrito nesta dissertação.

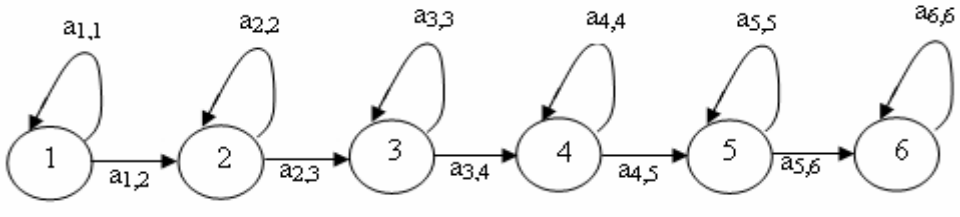


Figura 3.4 – HMM ligado da esquerda para a direita (ou Bakis) com 6 estados.

A arquitectura deste modelo está muito ligada ao tempo e às suas propriedades. À medida que o tempo avança, no domínio do modelo, traduz-se também num avanço numérico de estado ou permanência no mesmo. Tal como a evolução

temporal da pronunciação, a tipologia deste modelo também não permite transições para estados anteriores e não permite, de igual modo, avanços superiores à unidade. A razoabilidade desta aproximação prende-se com o facto de que não é possível inverter a ordem fonética sem que seja alterada a mensagem linguística (Modo –> Domo) e ainda o facto de que se forem retiradas unidades fonéticas também a mensagem linguística será alterada (ligada –> lida).

Outra característica destes modelos de Bakis é impor que a sequência de estados comece sempre no primeiro e acabe sempre no último estado, o que está de acordo com as formas de pronunciação existentes onde cada palavra começa e acaba invariavelmente no mesmo fonema. Dado isto, o vector $\pi = \{\pi_i\}$(probabilidade do estado *i* no instante inicial) e a matriz $A = \{a_{i,j}\}$ (probabilidade de transição do estado *i* para o estado *j*) têm as seguintes propriedades:

$$\pi_{s_t} = \begin{cases} 0 & para \quad s_t \neq 1 \\ 1 & para \quad s_1 = 1 \end{cases} \quad \& \quad a_{i,j} = 0 \quad para \quad j < i \quad \wedge \quad j > i+1 \qquad (3.6)$$

De um modo geral, para além destas condições, que são propriedades desta tipologia *Bakis*, mantêm-se todas as condições mais gerais da teoria dos HMM's. Por exemplo, a soma das probabilidades de transição de um estado para todos os outros é unitária, incluindo a transição para o mesmo estado, dado que em cada instante a transição ocorre de facto.

A função densidade de probabilidade inerente a qualquer transição é tipicamente uma mistura finita, multivariável de gaussianas da forma:

$$f(y_t / s_{t-1}, s_t, \lambda) = \sum_{m=1}^{M} p_{s_{t-1},s_t,m} \cdot G(y_t, \mu_{s_{t-1},s_t,m}, \varepsilon_{s_{t-1},s_t,m}) \; 1 \leq s \leq N \qquad (3.7)$$

Onde y_t é a observação, que frequentemente é multidimensional, no instante *t*; p_{s_{t-1},s_t} é o coeficiente da $m^{ésima}$ componente da mistura na transição $s_{t-1}s_t$; G (...) representa a distribuição normal (Gaussiana multivariável), de dimensão igual à dimensão do vector observação; $\mu_{s_{t-1},s_t,m}$ e $\varepsilon_{s_{t-1},s_t,m}$ (incluídos no conjunto de parâ-

metros representados por λ), representam o vector média e a matriz covariância na transição s_{t-1}, s_t para a componente de mistura *m* sendo *M* o número de misturas e *N* é o número de estados do modelo.

Dada uma sequência de observações $Y = \{y_1, y_2, ..., y_T\}$ de comprimento T, qual a probabilidade de o modelo ter gerado essa sequência de observações?

Seja $S = \{(s_0, s_1); (s_1, s_2); ...; (s_{T-1}, s_T)\}$ uma das possíveis sequências de transições, $f(y_t / s_{t-1}, s_t, \lambda)$ a função densidade probabilidade numa determinada transição, e $\lambda = \{A, B, \pi\}$ os parâmetro do modelo, a probabilidade de o modelo ter gerado essa sequência poderá ser dividida em duas partes:

- Considerando apenas o processo observável, a probabilidade é calculada pela seguinte função de densidade de probabilidade:

$$f(Y / S, \lambda) = f(y_t / s_0, s_1, \lambda) \cdot f(y_t / s_1, s_2, \lambda) ... \cdot f(y_t / s_{T-1}, s_T, \lambda) = \prod_{t=1}^{T} f(y_t / s_{t-1}, s_t, \lambda) \quad (3.8)$$

- Agora, considerando unicamente o processo não observável, a probabilidade da sequência de estados *S* é dada pela seguinte expressão:

$$P(S / \lambda) = \pi_{s_1} \cdot a_{s_0, s_1} \cdot a_{s_1, s_2} ... \cdot a_{s_{T-1}, s_T} \quad \& \quad \pi_{s_1} = \pi_1 = 1 \quad (3.9)$$

Logo,

$$P(S / \lambda) = \prod_{t=1}^{T} a_{s_{t-1}, s_t} \quad (3.10)$$

Juntando os dois processos, tem-se a probabilidade conjunta da sequência de observações *Y* e sequência de estados *S*. Essa probabilidade é representada pela seguinte função:

$$f(Y, S / \lambda) = f(Y / S, \lambda) \cdot P(S / \lambda) \quad (3.11)$$

Finalmente, a medida de compatibilidade entre série de observações e o modelo ou a função densidade de probabilidade marginal de *Y* é obtido por integração da equação (3.11) a todo espaço *S*. Sendo *S* discreto, visto que representa todas as possíveis sequências de estados, o integral resulta numa simples soma como mostra a seguinte equação:

$$f(Y / \lambda) = \sum_{S} f(Y, S / \lambda) \tag{3.12}$$

Estendendo toda a descodificação numa única função, isto é, substituindo as equações (3.7), (3.8) e (3.10) na equação (3.12), obtemos:

$$f(Y / \lambda) = \sum_{S} \prod_{t=1}^{T} \sum_{m=1}^{M} p_{s_{t-1},s_t,m} \cdot G\left(y_t, \mu_{s_{t-1},s_t,m}, \varepsilon_{s_{t-1},s_t,m}\right) . a_{s_{t-1},s_t} \tag{3.13}$$

Exemplo (continuação)

Continuando o exemplo, vamos agora mostrar a determinação da medida de compatibilidade ou, mais correctamente, a verosimilhança entre uma sucessão de observações e o modelo para reconhecimento de fala, descrito anteriormente. Posto isso, vamos assumir que é possível traduzir um sinal de voz por uma sucessão de 4 observações bidimensionais, ou seja, cada observação é um vector com os dois coeficientes de predição da fala, no qual vamos determinar a sua verosimilhança dado um modelo treinado.

Seja $y_t = [cof1, cof2]$ a observação no instante *t* e λ os parâmetros do modelo, vamos começar por determinar a verosimilhança de uma determinada observação numa dada transição. Para a calcular temos que considerar o processo observável e o não observável, que vistos conjuntamente numa determinada transição resultam na seguinte expressão:

$$F\left(y_t / s_{t-1}, s_t, \lambda\right) = \underbrace{\sum_{m=1}^{3} p_{s_{t-1},s_t,m} \cdot G\left(y_t, \mu_{s_{t-1},s_t,m}, \varepsilon_{s_{t-1},s_t,m}\right)}_{observável} \cdot \underbrace{a_{s_{t-1},s_t}}_{não\ observável} \qquad 1 \le s \le 3 \tag{3.14}$$

A figura 3.5 ilustra todas as cadeias de estados possíveis para a sucessão de observações e modelo assumidos anteriormente.

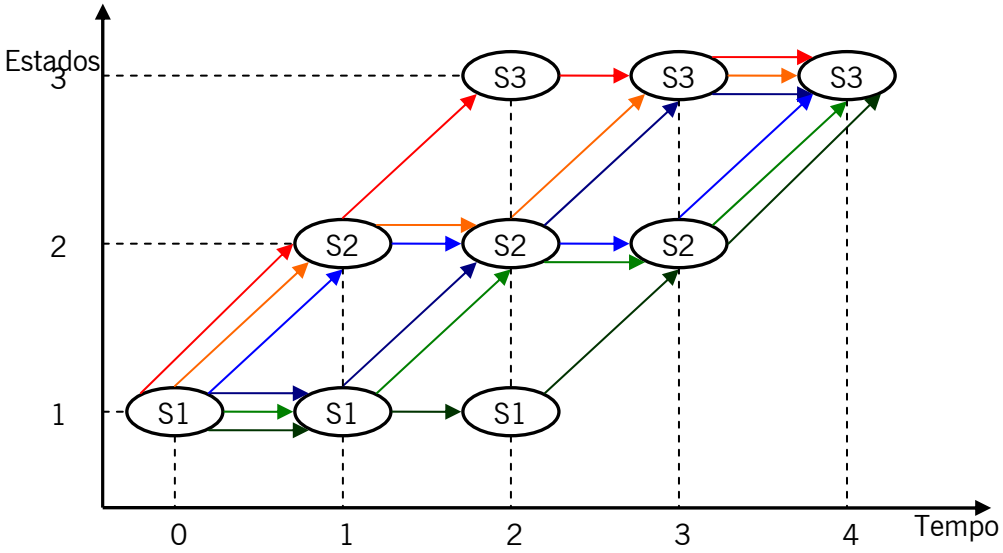


Figura 3.5 – Cadeias de estados para uma sucessão de 4 observações, dado um modelo de Bakis de 3 estados.

A função que torna possível a determinação da probabilidade de o modelo ter gerado a sequência $S = \{(1,2)_1; (2,2)_2; (2,3)_3; (3,3)_4\}$ (uma das possíveis sequências temporais de transições ao longo do modelo – cadeia representada a laranja na figura 14), dado $Y = \{y_1, y_2, y_3, y_4\}$ (sequência de observações), é expressa por:

$$f(Y, S/\lambda) = \prod_{t=1}^{4} \sum_{m=1}^{3} p_{s_{t-1},s_t,m} \cdot G\left(y_t, \mu_{s_{t-1},s_t,m}, \varepsilon_{s_{t-1},s_t,m}\right) \cdot a_{s_{t-1},s_t} \qquad (3.15)$$

Concluindo, para determinar a medida de compatibilidade entre o modelo λ e a sucessão de observações Y basta somar a contribuição das várias sequências de estados, que vai resultar na seguinte expressão:

$$f(Y/\lambda) = \sum_{S} \prod_{t=1}^{4} \sum_{m=1}^{3} p_{s_{t-1},s_t,m} \cdot G\left(y_t, \mu_{s_{t-1},s_t,m}, \varepsilon_{s_{t-1},s_t,m}\right) \cdot a_{s_{t-1},s_t} \qquad (3.16)$$

3.2 Problema da Aprendizagem

O problema mais difícil nos HMM's é encontrar um processo que adapte o mais possível os parâmetros do modelo (**A, B, π**) de modo a satisfazer um dado critério de optimização. Não se conhece nenhum método analítico que permita uma maximização global, isto é, para todos os parâmetros do modelo. Contudo, é possível escolher um $\lambda = \{A, B, \pi\}$ tal que a função densidade de probabilidade $f(Y / \lambda)$ seja maximizada localmente através de um processo iterativo, como é exemplo o método de Baum-Welch (que é um caso particular do método EM (Expectation Maximisation)).

Esta secção descreve muito basicamente o algoritmo EM e mostra como este pode ser aplicado no contexto dos HMM's, nomeadamente no treino deste tipo modelos.

3.2.1 Conhecimentos básicos do algoritmo EM

O algoritmo EM (*Expectation – Maximisation*) é uma técnica muito utilizada no cálculo iterativo de estimativas de máxima verosimilhança em problemas com dados incompletos ou não observáveis. Devido à ausência de parte dos dados (dados não observáveis) o cálculo destas estimativas torna-se uma tarefa bastante complexa. A base deste algoritmo assenta em associar o problema de dados incompletos a um problema de dados completos para o qual o cálculo da estimativa se torna bastante simplificada.

Este algoritmo é composto por duas etapas: a etapa da Esperança (*Expectation step*); e a etapa da Maximização (*Maximisation step*). Na primeira etapa, é calculado o valor esperado dos dados incompletos, através do cálculo de uma estimativa com base nos dados completos e dos dados observáveis. Na segunda etapa, usam-se os dados da primeira como se tivessem sido de facto observados para determinar a estimativa de máxima verosimilhança dos parâmetros da distribuição dos dados completos.

Consideremos então dois espaços *S* e *Y* que correspondem a um espaço não observável e a um espaço observável, respectivamente. Suponhamos ainda que $f(S/\lambda)$ e $f(Y/\lambda)$ são membros da família paramétrica das funções densidade de probabilidade definidas em *S* e *Y*, respectivamente, para o conjunto de parâmetros λ. O objectivo do algoritmo EM é maximizar a verosimilhança dos dados observáveis *y* ($y \in Y$) através da alteração dos parâmetros definidos λ explorando a relação entre $f(y,s/\lambda)$ e $f(s/y,\lambda)$, *onde* $s \in S$.

A função densidade de probabilidade conjunta $f(y,s/\lambda)$ obtém-se facilmente aplicando o teorema de Bayes à relação entre $f(y,s/\lambda)$ e $f(s/y,\lambda)$, resultando na seguinte expressão:

$$f(y,s/\lambda) = f(s/y,\lambda) \cdot f(y/\lambda) \tag{3.17}$$

Normalmente, devido à exponencial existente na distribuição *Gaussiana* e às várias multiplicações utilizadas nos cálculos, utiliza-se o logaritmo para determinar o máximo da verosimilhança, dado que simplifica os cálculos. Aplicando, então, o logaritmo a ambos os membros da equação (3.17) obtém-se:

$$\log f(y/\lambda) = \log f(y,s/\lambda) - \log f(s/y,\lambda) \tag{3.18}$$

Dado dois conjuntos de parâmetros λ'e λ, a esperança matemática do logaritmo da verosimilhança dos dados incompletos (observáveis) $L(y,\lambda') = Q(\lambda,\lambda') - H(\lambda,\lambda')$ sobre os dados completos (*Y*, *S*) condicionados por *Y* é:

$$\begin{aligned} E_s[L(y,\lambda')/y,\lambda] &= E_s[\log f(y/\lambda'/y,\lambda)] \\ &= \int \log f(y/\lambda') \cdot f(s/y,\lambda) ds = \log f(y/\lambda') = L(y,\lambda') \end{aligned} \tag{3.19}$$

Onde $E_s[./y,\lambda]$ é esperança matemática condicionada por *y* e λ sobre os dados completos (*y, s).* Mudando de notação e a partir das equações (3.8) e (3.9) obtém-se:

$$L(y,\lambda') = Q(\lambda,\lambda') - H(\lambda,\lambda') \tag{3.20}$$

Onde:

$$Q(\lambda,\lambda') = E_s\left[\log f(y,s/\lambda')/y,\lambda\right] = \int \log f(y,s/\lambda') f(s/y,\lambda) ds \quad (3.21)$$

e

$$H(\lambda,\lambda') = E_s\left[\log f(s/y,\lambda')/y,\lambda\right] \quad (3.22)$$

A grande vantagem deste algoritmo é o facto de $H(\lambda,\lambda')$ ser sempre menor ou igual a $H(\lambda,\lambda)$, de iteração para iteração (desigualdade de *Jensen*). Logo, este facto implica que a maximização de $L(y,\lambda')$ dependa só da maximização de $Q(\lambda,\lambda')$. Este máximo poderá não ser um máximo global mas é, com certeza, um máximo relativo.

A função Q quando o vector aleatório *S* (não observável) é discreto, como é o caso presente, representa-se por:

$$Q(\lambda,\lambda') = \sum_s \frac{f(y,s/\lambda)}{f(y/\lambda)} \cdot \log f(y,s/\lambda') \quad (3.23)$$

Dada a estimativa actual do conjunto de parâmetros λ e a estimativa seguinte λ', o algoritmo EM pode ser resumido da seguinte forma:

- Primeira etapa, esperança matemática. Calcular $Q(\lambda,\lambda')$ com base na estimativa dos parâmetros actuais.
- Segunda etapa, maximização. Calcular um novo conjunto de parâmetros que maximiza a função $Q(\lambda,\lambda')$.
- Por último, fazer $\lambda = \lambda'$, e repetir o procedimento até obter convergência.

3.2.2 Aplicação do algoritmo EM em HMM's

Como mencionado, no início deste capítulo, os modelos de Markov geram um processo duplamente estocástico composto por uma sequência de estados (dados não observáveis) e uma sequência de observações (dados observáveis). As sequências de observações são chamadas de dados incompletos porque a sequên-

cia de estados é um processo interno dos modelos, logo, tornando os dados observáveis insuficientes. Devido à impossibilidade de obter dados completos logo a partir das observações, os modelos de Markov têm que ser treinados a partir de métodos de maximização de probabilidades especializados em dados incompletos, como é exemplo o algoritmo EM.

Seguindo, então, o algoritmo EM para treinar os modelos de Markov, temos as seguintes etapas:

- **Cálculo da função $Q(\lambda,\lambda')$ com base em parâmetros actuais.**

A função densidade de probabilidade dos dados completos é dada pela equação (3.17), e para um novo conjunto de parâmetros λ' é dada por:

$$f(Y,S/\lambda') = f(Y/S,\lambda')\cdot P(S/\lambda') \tag{3.24}$$

Aplicando esta expressão no contexto da expressão (3.23) obtém-se esta nova expressão:

$$Q(\lambda,\lambda') = \sum_s \frac{f(y,s/\lambda)}{P(y/\lambda)}\cdot \log f(y,s/\lambda') = \sum_s \frac{f(y,s/\lambda)}{P(y/\lambda)}\cdot \log f(y,s/\lambda') \tag{3.25}$$

Estendendo a função $f(y,s/\lambda)$ a partir das equações (3.7) e (3.10) resulta:

$$Q(\lambda,\lambda') = \sum_s \frac{f(y,s/\lambda)}{P(y/\lambda)}\cdot \log \prod_t^T a'_{s_{t-1},s_t}\cdot f(y_t/s_{t-1},s_t,\lambda') \tag{3.26}$$

Como o logaritmo de uma série de multiplicações é a soma dos logaritmos dessa série, logo:

$$Q(\lambda,\lambda') = \sum_s \frac{f(y,s/\lambda)}{P(y/\lambda)}\cdot \log\left\{\sum_{t=1}^T \log a'_{s_{t-1},s_t} + \sum_{t=1}^T \log f(y_t/s_{t-1},s_t,\lambda')\right\} \tag{3.27}$$

Estendendo a função $f(y_t/s_{t-1},s_t,\lambda)$ a partir da equação (3.7) obtém-se:

$$Q(\lambda,\lambda') = \sum_{m=1}^{M}\sum_{s}\frac{f(y,s/\lambda)}{P(y/\lambda)}\cdot\log\left\{\begin{array}{l}\sum_{t=1}^{T}\log a'_{s_{t-1},s_t} + \\ \quad + \sum_{t=1}^{T}\log p'_{s_{t-1},s_t,m} + \\ \quad\quad + \sum_{t=1}^{T}\log G\left(y_t, \mu'_{s_{t-1},s_t,m}, \varepsilon'_{s_{t-1},s_t,m}\right)\end{array}\right\} \quad (3.28)$$

- **Estimativa dos novos parâmetros** λ'

A função (3.28) mostra claramente que a função $Q(\lambda,\lambda')$ é separável em três termos independentes, sendo o primeiro dependente só da probabilidade de transição de estado, o segundo dependente das misturas *Gaussianas* e o último depende apenas do vector observação. Posto isto, cada um destes três termos independentes pode ser visto como uma função auxiliar de $Q(\lambda,\lambda')$, permitindo assim maximizar $Q(\lambda,\lambda')$, separadamente, a partir da maximização de cada uma das auxiliares. As três funções auxiliares são expressas por:

- $Q(\lambda, a'_{i,j}) = \sum_{m=1}^{M}\sum_{s}\frac{f(y,s/\lambda)}{P(y/\lambda)}\cdot\log\left\{\sum_{t=1}^{T}\log a'_{s_{t-1},s_t}\right\}$ (3.29)
- $Q(\lambda, p'_{i,j,m})\sum_{m=1}^{M}\sum_{s}\frac{f(y,s/\lambda)}{P(y/\lambda)}\cdot\log\left\{\sum_{t=1}^{T}\log p'_{s_{t-1},s_t,m}\right\}$ (3.30)
- $Q(\lambda, G(.)') = \sum_{m=1}^{M}\sum_{s}\frac{f(y,s/\lambda)}{P(y/\lambda)}\cdot\log\left\{\sum_{t=1}^{T}\log G\left(y_t, \mu'_{s_{t-1},s_t,m}, \varepsilon'_{s_{t-1},s_t,m}\right)\right\}$ (3.31)

Todas estas funções auxiliares são do tipo:

$$\sum_{j=1}^{N} w_j \cdot \log y_j \quad (3.32)$$

Como a função y_j está sujeita às restrições $\sum_{j=1}^{N} y_j = 1$ e $y_j \geq 0$ (a soma de uma função densidade de probabilidade tem que ser igual a 1 e só existem probabilidades positivas) este tipo de funções, com estas restrições, têm um máximo global dado por:

$$y_j = \frac{w_j}{\sum_i^N w_i}, \quad j = 1,2,...,N \tag{3.33}$$

Usando a equação (3.33) na equação auxiliar $Q(\lambda, a'_{i,j})$, obtém-se as novas probabilidades de transição de estado que maximizam $Q(\lambda, a'_{i,j})$. Essas probabilidades são dadas por:

$$a'_{i,j} = \frac{\sum_{t=1}^{T-1} f(\mathbf{Y}, s_t = i, s_{t+1} = j / \lambda)}{\sum_{t=1}^{T-1} f(\mathbf{Y}, s_t = i / \lambda)} \tag{3.34}$$

Para obter os novos coeficientes das misturas basta seguir o raciocínio anterior na equação $Q(\lambda, p'_{j,m})$. A equação que permite calcular estes novos coeficientes obtém-se substituindo (3.33) em (3.30) e é dada por:

$$p'_{j,m} = \frac{\sum_{t=1}^{T} f(\mathbf{Y}, s_t = j, m / \lambda)}{\sum_{t=1}^{T} f(\mathbf{Y}, s_t = j / \lambda)}$$

(3.35)

Por fim, a determinação do novo vector média e da nova matriz covariância são obtidos pelo método usual de maximização (derivar e igualar a zero) dado estes parâmetros não estarem sujeitos às restrições implícitas na equação (3.33). Dado isto, o máximo do vector $\mu'_{s_{t-1}, s_t, m}$ e o máximo da matriz $\varepsilon_{s_{t-1}, s_t, m}$ são calculados através das seguintes expressões:

$$\mu'_{n,m,i} = \frac{\sum_{t=1}^{T} \gamma_t(n,c) y_{t,i}}{\sum_{t=1}^{T} \gamma_t(n,c)} \tag{3.35}$$

$$\sigma'^{2}_{n,m,i} = \frac{\sum_{t=1}^{T} \gamma_t(n,c)(y_{t,i} - \mu'_{n,m,i})^2}{\sum_{t=1}^{T} \gamma_t(n,c)} \tag{3.36}$$

Onde $\gamma_t(n,m) = f(y_t / s_{t-1}, s_t, m, \lambda)$ e a transição entre estados $n = s_{t-1}, s_t$.

Exemplo (finalização)

Finalizando o exemplo, vejamos como é feito o treino do modelo de Markov que se vem desenvolvendo neste capítulo.

Para modelar ou treinar as características de uma determinada palavra num modelo de Markov é necessário, obrigatoriamente, um conjunto de padrões de fala dessa mesma palavra. No entanto, para simplificar e reduzir esta explicação vamos considerar que o modelo de Markov, deste exemplo, vai ser treinado apenas com um padrão de fala $Y = \{y_t\}$ com dimensão $T = 4$ e que o resultado do modelo fica logo optimizado no fim da primeira iteração.

- **Primeiro passo, determinar as funções auxiliares de $Q(\lambda, \lambda')$.**

$$Q(\lambda, a'_{i,j}) = \sum_{s} \sum_{m=1}^{3} \frac{f(Y, s/\lambda)}{P(y/\lambda)} \cdot \log\left\{ \sum_{t=1}^{4} \log a'_{s_{t-1}, s_t} \right\} \tag{3.37}$$

$$Q(\lambda, p'_{i,j,m}) = \sum_{s} \sum_{m=1}^{3} \frac{f(Y, s/\lambda)}{P(y/\lambda)} \cdot \log\left\{ \sum_{t=1}^{4} \log p'_{s_{t-1}, s_t, m} \right\} \tag{3.38}$$

$$Q(\lambda, G(.)') = \sum_{s} \sum_{m=1}^{3} \frac{f(Y, s/\lambda)}{P(y/\lambda)} \cdot \log\left\{ \sum_{t=1}^{4} \log G\left(y_t, \mu'_{s_{t-1}, s_t, m}, \varepsilon'_{s_{t-1}, s_t, m}\right) \right\} \tag{3.39}$$

- **Segundo passo, maximizar as funções auxiliares.**

  O $a'_{i,j}$ que maximiza $Q(\lambda, a'_{i,j})$ é:

$$a'_{i,j} = \frac{\sum_{t=1}^{3} f(\mathbf{Y}, s_t = i, s_{t+1} = j / \lambda)}{\sum_{t=1}^{3} f(\mathbf{Y}, s_t = i / \lambda)} \tag{3.40}$$

O $p'_{j,m}$ que maximiza $Q(\lambda, p'_{i,j,m})$ é

$$p'_{j,m} = \frac{\sum_{t=1}^{4} f(\mathbf{Y}, s_t = j, m / \lambda)}{\sum_{t=1}^{4} f(\mathbf{Y}, s_t = j / \lambda)} \tag{3.41}$$

Por ultimo, o $\mu'_{n,m,i}$ e $\sigma'^{2}_{n,m,i}$ que maximiza $Q(\lambda, G(.)')$ são:

$$\mu'_{n,m,i} = \frac{\sum_{t=1}^{4} \gamma_t(n,c) y_{t,i}}{\sum_{t=1}^{4} \gamma_t(n,c)} \tag{3.42}$$

E,

$$\sigma'^{2}_{n,m,i} = \frac{\sum_{t=1}^{4} \gamma_t(n,c)(y_{t,i} - \mu'_{n,m,i})^2}{\sum_{t=1}^{4} \gamma_t(n,c)} \tag{3.43}$$

- **Último passo, e para terminar a primeira iteração, fazer:**

$$a_{i,j} = a'_{i,j}, \tag{3.44}$$

$$p_{j,m} = p'_{j,m}, \tag{3.47}$$

$$\mu_{n,m,i} = \mu'_{n,m,i} \text{ e} \tag{3.48}$$

$$\sigma^{2}_{n,m,i} = \sigma'^{2}_{n,m,i} \tag{3.49}$$

Capítulo 4

Implementação do sistema

Este método foi implementado em C++ (uma linguagem de programação) através da ferramenta Visual C++ e inserido num objecto COM para Windows. Com a criação deste objecto COM, convertemos esta implementação num *software* dinâmico, isto é, num sistema que possa ser interligado com qualquer outro *software* que necessite de reconhecimento de "Voz".

O principal objectivo deste capítulo é descrever a implementação do Reconhecedor, apresentando as partes mais importantes do *software*. Este capítulo começa por apresentar, muito resumidamente, outros tipos de *softwares* que existem actualmente em reconhecimento automático da fala; de seguida, aborda o conceito de COM (Component Object Model) onde foi inserido a implementação do método; posteriormente, apresenta o diagrama de classes que descreve toda a arquitectura da implementação; e, por fim, explica o código das partes mais importantes deste sistema de reconhecimento.

4.1 Outros *softwares* de Reconhecimento

Este ponto foi criado com o objectivo de dar a conhecer outros sistemas de reconhecimento da identidade do orador existentes no mercado. Outro objectivo é mostrar a que nível se encontra este tipo de sistemas, actualmente, e onde diferem em relação ao apresentado nesta dissertação.

Com a pesquisa feita unicamente na Internet, vários sistemas foram encontrados relativos ao reconhecimento do orador. Porém, alguns deles estão apenas

documentados e outros patenteados. No entanto, foram encontrados dois sistemas de duas empresas, a comercializarem já o produto final, uma é a VoiceVault e a outra a VoiceVerified®.

VoiceVault é um sistema de Reconhecimento/verificação do orador por biometria. Ele reconhece e verifica correctamente a identidade do orador seja por telefone, pela Internet ou por rede de área local. O reconhecimento é efectuado através de palavras isoladas e a verificação é através da pronunciação de um PIN ou palavra-chave dado o *username*. Esta tecnologia é usada de modo a obter aplicações comerciais e governamentais.

Figura 4.1 – Logótipo VoiceVault.

VoiceVerified é, de igual modo, um sistema de reconhecimento/verificação por biometria tal como o anterior, contudo, este sistema apenas funciona pelo telefone.

Figura 4.2 – Logótipo VoiceVerified.

As aplicações que estas empresas comercializam são várias, mas todas elas baseadas nos seus sistemas de verificação que podem ser mais ou menos complexos consoante a aplicação.

4.2 Component Object Model (COM)

Um sistema que utilize programação orientada a objectos e que necessite de vários módulos implementados por entidades diferentes, funcionará correctamente se os vários componentes conseguirem interagir uns com os outros. Para a interacção ter êxito todos os módulos têm que utilizar uma estrutura padrão. O *Component Object Model* é uma arquitectura padrão binária para objectos de *software*, que torna um componente independente da linguagem de programação utilizada no seu desenvolvimento. Um objecto COM é composto apenas por código máquina. Com esta característica, o COM tem a vantagem de ser directamente executado sem necessidade de compilação, sempre que um cliente interagir com ele. Outro benefício do código máquina é tornar quase impossível a descodificação da funcionalidade que se encontra implementada no seu interior.

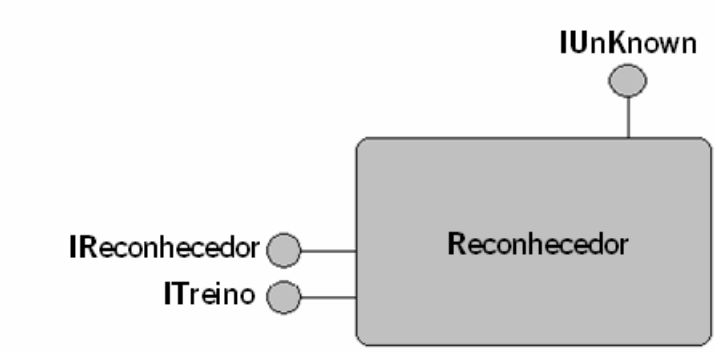


Figura 4.3 – Esquema do componente Reconhecedor com os respectivos interfaces.

O símbolo utilizado para definir estes objectos é composto por um rectângulo que representa todos os serviços que estão implementados nesse COM e por um conjunto de bolas que representam os interfaces ou interacções para os serviços que estão disponíveis, como ilustra a figura 4.3.

4.2.1 Algumas vantagens de objectos COM

Os objectos COM apresentam algumas vantagens, nomeadamente:

- Independência da linguagem (um componente COM pode ser feito em qualquer linguagem de programação e permite a partilha de código binário entre aplicações de linguagens diferentes).

- Facilidade em estender funcionalidades a uma aplicação, sem necessidade de voltar a compilar de novo todo o projecto.
- Reutilização da arquitectura de aplicações (aplicações com a mesma organização e funcionalidades diferentes como, por exemplo, as aplicações do Microsoft Office).
- Registo do componente no Windows (permitindo a chamada do componente e das suas funcionalidades através de um simples identificador (GUID), independentemente do local onde se encontra a COM).
- Difícil descodificação (estes objectos encontram-se em código máquina o que torna a descodificação, do sistema que se encontra nele implementado, numa tarefa quase impossível).

4.2.2 Arquitectura

Uma aplicação, que pretenda interagir com um objecto COM, nunca pode comunicar directamente com as suas funcionalidades, porque um COM é um sistema isolado e independente da aplicação. A interacção entre um componente COM e uma aplicação, que necessite das suas funcionalidades ou serviços, faz-se através de estruturas designadas por interfaces. Concretamente, neste sistema existem dois interfaces, um para Treino dos modelos e outro para Reconhecimento ou Verificação de padrões de fala, separando assim as duas aptidões do sistema, tornando a estrutura do sistema mais elegante e simples.

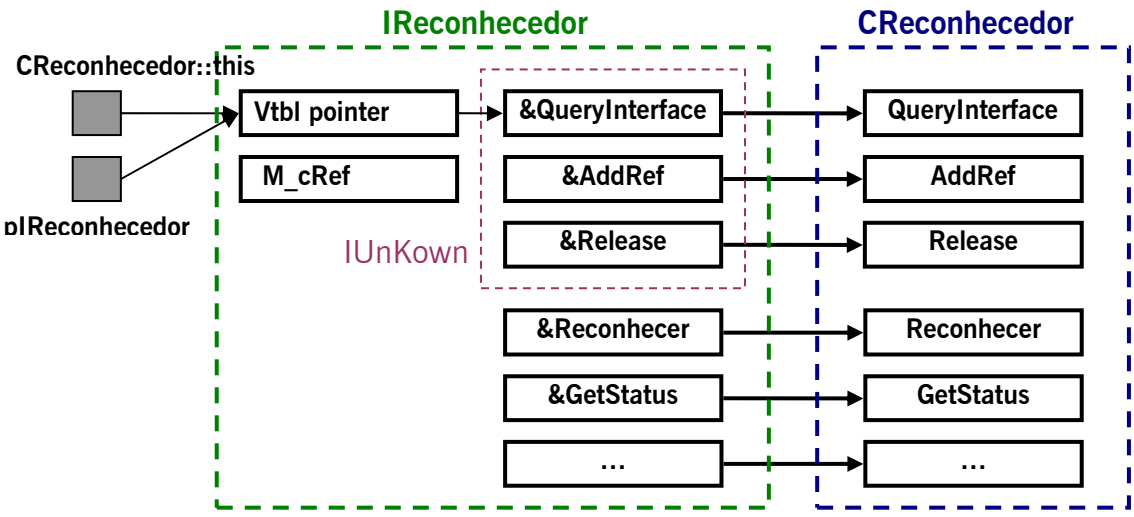


Figura 4.4 – Disposição de memória do objecto Reconhecedor.

Um interface, basicamente, resume-se a uma tabela de endereços de funções e de um apontador para função. Todas as funções que necessitam de interacção com o exterior (aplicação) têm que estar referenciadas na tabela de endereços do interface, para que possam ser executadas através do apontador para a função existente, de igual modo, no interface. Exemplo disso, são as funções *Reconhecer* e *GetStatus* do interface Reconhecedor (ver figura 4.4). Para além dos apontadores para funcionalidades, na tabela de endereços existem também três referências para as funções: *QueryInterface*, *AddRef* e *Release*. Estas são comuns a todos os componentes COM e a finalidade delas consiste em criar a ligação entre a aplicação e o componente, contar as referências dos COM criados (o número de referencias está na variável *M_cRef* do interface) e libertar a memória do objecto, respectivamente. Devido ao facto dessas funções serem importantíssimas para o funcionamento do COM, e serem funções comuns a todos os Componentes COM, o interface para elas encontra-se num interface base, o *IunKown*, de onde todos os outros são derivados.

4.3 Arquitectura da implementação

A figura 4.5 representa o diagrama de classes que descreve toda a arquitectura do sistema de reconhecimento no contexto implementacional. As *coclass* (classes da estrutura do objecto COM) que estão apresentadas no diagrama com cores quentes (laranja e vermelho) são os únicos pontos de interacção que existe entre o exterior e o resto do sistema. As restantes, que estão apresentadas no diagrama com cores mais frias (azul, verde, roxo), são classes mais internas que implementam funcionalidades mais específicas, porém indispensáveis neste sistema de reconhecimento.

As secções seguintes mostram e explicam fracções da implementação que se pensou serem mais relevantes e pertinentes no contexto desta tese. Com isso, é também explicado o porquê de algumas interligações entre classes representadas no diagrama da figura 4.5.

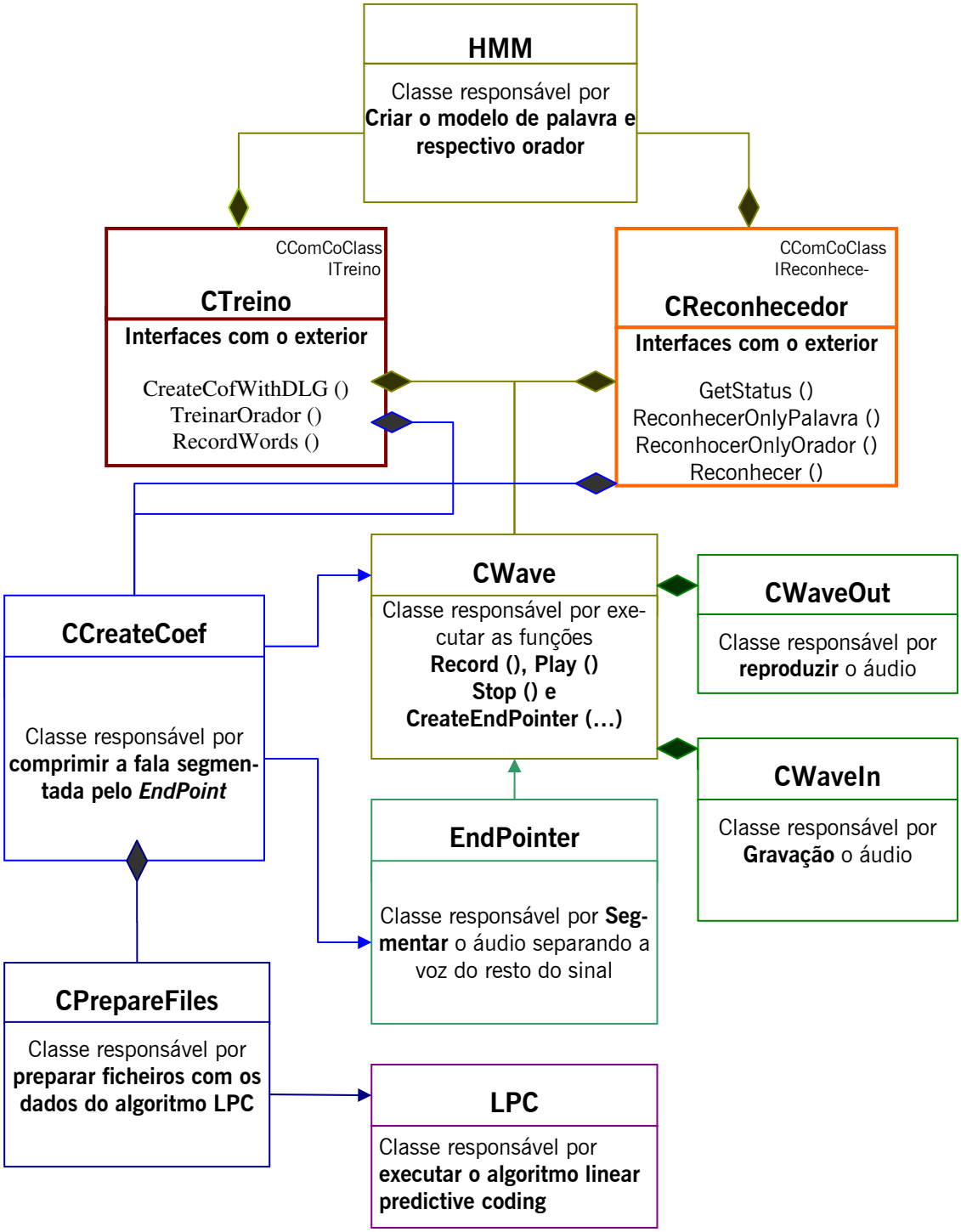


Figura 4.5 – Diagrama de classes do sistema implementado.

4.4 Interacções com o sistema

Para interagir com o nosso sistema, que está implementado numa COM, é necessário uma aplicação que interaja com ele, visto que, um objecto COM é um

sistema isolado. Para verificação do funcionamento do sistema foi criado um *software* de interacção muito simples contendo apenas dois botões, como ilustra a figura 4.6. Um dos botões solicita as funcionalidades do interface de treino da COM e o outro invoca as funcionalidades do interface de reconhecimento da COM.

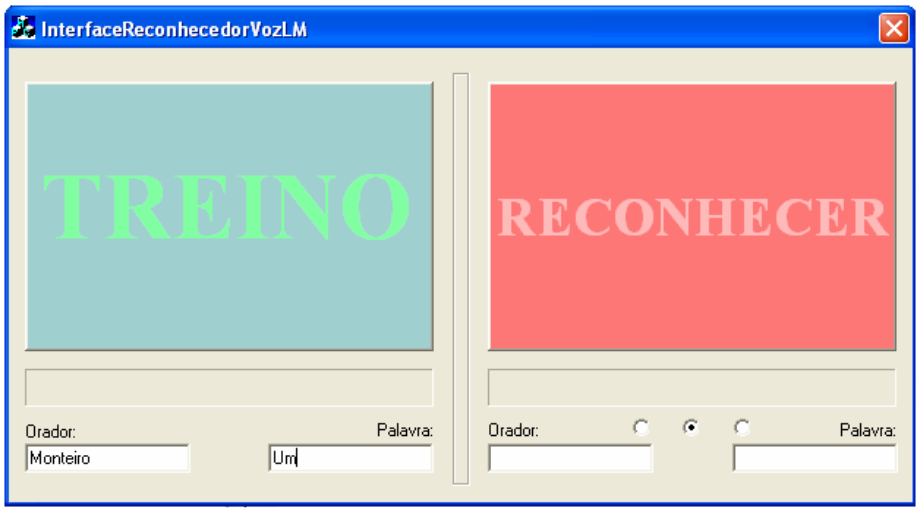


Figura 4.6 – Software de interacção com a COM

Para invocar uma funcionalidade de uma COM é necessário proceder a alguns passos. Primeiro, é necessário criar a base da conexão com a função *CoInitialize* (Inicialização das bibliotecas COM), depois, criar uma instância para o objecto e respectivo interface com a função *CoCreateInstance* e, por último, chamar uma das funcionalidades do COM. Esta sequência de eventos são mostrados no código exemplo 1.

Código exemplo 1. Interacção com a funcionalidade de gravação do interface de treino.

```
ItreinoLM* pITreino;
hr = CoInitialize(0);
if(SUCCEEDED(hr))
{
    hr = CoCreateInstance( CLSID_Treino, NULL,
                           CLSCTX_INPROC_SERVER,IID_Itreino, (void**) &pITreino);
    if(!SUCCEEDED(hr))
    {
        AfxMessageBox("CoCreateInstance Failed for treino.");
        Cdialog::OnClose();
    }
}else
{
    AfxMessageBox("CoCreateInstance Failed for treino.");
    Cdialog::OnClose();
 }
 pITreino->RecordWords();
```

4.4.1 Escolha dos padrões para o treino

O treino dos modelos deve-se efectuar, apenas, com padrões de fala que sejam o mais representativos possível da palavra que se pretende treinar. Por isso, a selecção dos padrões que entrarão no treino do modelo é uma tarefa de extrema importância. Durante a gravação da fala podem acontecer imprevistos, como ruídos mais fortes, que podem ser interpretados como fala. Quando isso acontece é necessário retirar essas partes da gravação para que o treino do modelo seja feito apenas com as características da palavra e orador.

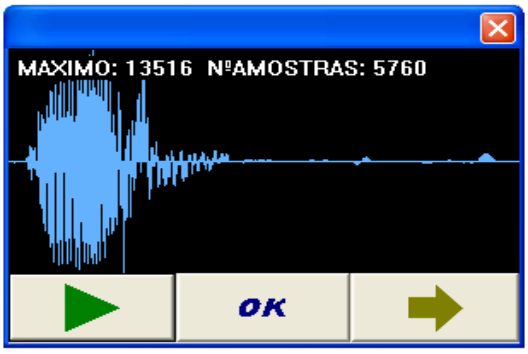


Figura 4.7 – Janela de selecção das palavras que entrarão no treino.

Para fazer a selecção dos padrões de fala que entram no treino é apresentado ao utilizador uma janela como apresenta a figura 4.7. Nessa janela é possível ouvir e observar cada uma das palavras que foram detectadas e segmentadas pelo detector fala/pausa (*Endpoint)* e a partir dessa audição e observação decidir sobre a aceitação ou rejeição da palavra. Deste modo, só orações completas e mais representativas serão usadas para o treino dos modelos.

4.4.2 Interface do treino

Para fazer o treino é necessário proceder aos seguintes passos:

- **Gravar uma série de repetições da mesma palavra para o mesmo locutor.**

  A gravação de uma série de palavras é feita invocando a função *Record-Words*. Nessa função é gravado o áudio e, simultaneamente, é feita a segmentação do áudio através do detector fala/pausa. (código exemplo 2).

Código exemplo 2. Função RecordWords do interface Treino.

```
STDMETHODIMP CtreinoLM::RecordWords()
{
   AFX_MANAGE_STATE(AfxGetStaticModuleState())

   if(!m_WaveTR.Record(1))return S_FALSE;

   return S_OK;
}
```

- **Extracção de características através do método LPC.**

A parametrização da fala é feita com o método *CreateCofWithDLG* (código exemplo 3) que, por sua vez, abrirá uma janela para fazer selecção só das orações que estejam perfeitas e integralmente gravadas.

As características das respectivas orações vão ser guardadas num único ficheiro formando assim o ficheiro com o conjunto de padrões de entrada para o treino.

Código exemplo 3. Função Criar coeficientes do interface Treino.

```
STDMETHODIMP CtreinoLM::CreateCofWithDLG(BSTR nome, BSTR palavra)
{
   AFX_MANAGE_STATE(AfxGetStaticModuleState())
   Cstring Nome=(Cstring)nome;
   Cstring Palavra=(Cstring)palavra;
   CCreateCofDLG CofDLG(&m_WaveTR);

   CofDLG.SetPath(m_pathCof);
   CofDLG.SetOrador(Nome);
   CofDLG.SetPalavra(Palavra);

   CofDLG.DoModal();
   m_WaveTR.Stop();
   m_makeTreino=CofDLG.GetMakeTreino();
   return S_OK;
}
```

- **Criação do modelo e treino.**

Dado o ficheiro que contém as características da fala, a criação do modelo pode ser efectuado de diferentes formas:

  - Criação do modelo de raiz. Neste caso o modelo é treinado como se o sistema nada mais conhecesse além da fala gravada e apresentada para treino (este treino é utilizado quando não existe modelo do orador nem modelo inicial da palavra).

- Criação de modelos dependentes do orador a partir de modelos independentes do orador. Esta modalidade pressupõe que as características da oração possam ser treinadas a partir de fala de qualquer orador. A adaptação do modelo a um orador pode agora fazer-se por retreino ou adaptação com fala do orador pretendido. A vantagem desta modalidade é que a adaptação requer muito menos orações que o treino de um modelo de raiz (este treino é usado para minimizar tempo de treino quando já existe modelo inicial da palavra).
- Adaptação de modelos já existentes com o objectivo de os melhorar.

Seguindo o código exemplo 4, é possível verificar quando e em que condições são utilizadas cada uma destas formas de treino possíveis.

Código exemplo 4. Função Treinar Orador do interface Treino.

```
STDMETHODIMP Ctreino::TreinarOrador(BSTR nome, BSTR palavra)
{
   AFX_MANAGE_STATE(AfxGetStaticModuleState())

   Cstring Nome=(Cstring)nome;
   Cstring Palavra=(Cstring)palavra;
   CcomplementFiles CF;
   try
   {
       if(!m_makeTreino)return S_OK;

       if(ExisteModelo(Nome,Palavra,m_pathModels))
          TreinarModelo (…);
       else
       {
          if(ExisteModelo("Inicial",Palavra,m_pathModelsIniciais ))
               TreinarModelo (…);
          else
               TreinarModelo(…);
       }
   }
   catch (Cexception* e)
   {
       LPTSTR erro=NULL;
       e->GetErrorMessage(erro,3);
       AfxMessageBox(erro);
   }
   return S_OK;
}
```

O nome dos modelos criados tem uma sintaxe própria de onde é possível extrair o nome do orador e a oração que corresponde ao respectivo modelo. A sintaxe é a seguinte: ORADOR_PALAVRA.HMM, onde ORADOR é o nome do orador e PALAVRA é o nome da palavra como por exemplo: Monteiro_Um.HMM.

4.4.3 Interface Reconhecedor

Neste interface estão todas as funcionalidades necessárias para fazer o reconhecimento do orador dependente do texto.

Para fazer o reconhecimento é invocado um método chamado *Reconhecer* que recebe como parâmetros de entrada dois ponteiros para *strings*, um dos quais para a *string* nome do orador e o outro para a *string* palavra. Esses ponteiros são responsáveis por devolver o nome do orador e palavra reconhecida. Esta função começa por fazer a captura do áudio, depois cria os coeficientes e, por fim, descobre o modelo que se ajusta melhor a esses coeficientes, através do maior valor de verosimilhança, como mostra o código exemplo 5.

Código exemplo 5. Função Reconhecer do interface Reconhecedor.

```
STDMETHODIMP CReconhecedor::Reconhecer(BSTR *nome, BSTR *palavra)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    if(!m_WaveR.Record(1,110))
    {
        m_status="R_ERROAUDIO";
        *nome=m_status.AllocSysString();
        *palavra=m_status.AllocSysString();
        return S_FALSE;
    }
    if(!CreateCof())
    {
        m_status="R_NOWAVE";
        *nome=m_status.AllocSysString();
        *palavra=m_status.AllocSysString();
        return S_FALSE;
    }
    Cstring Nome;
    Cstring Palavra;
    if(!BestModel(&Nome,&Palavra))
    {
        m_status="R_NOMODELO";
        *nome=m_status.AllocSysString();
        *palavra=m_status.AllocSysString();
        return S_FALSE;
    }
    m_status="R_OK";
    *nome=Nome.AllocSysString();
    *palavra=Palavra.AllocSysString();
    return S_OK;
}
```

Por vezes, podem ocorrer imprevistos ou erros que, à primeira vista, não são de fácil detecção. Como são exemplos, a detecção de um número de palavras diferente ao esperado, a não existência de nenhum modelo na biblioteca, os erros durante a gravação do áudio, etc. Para resolver esse problema é disponibilizada a

função *GetStatus* que devolve uma *string* com o estado do sistema (Código exemplo 6).

Código exemplo 6. Função Get Status do interface Reconhecedor.

```
STDMETHODIMP Creconhecedor::GetStatus(BSTR *status)
{
   AFX_MANAGE_STATE(AfxGetStaticModuleState())
   *status=m_status.AllocSysString();
   return S_OK;
}
```

4.5 Sincronismo entre Gravação e *Endpoint*

A gravação e a segmentação da fala devem ser simultâneos para permitirem funcionamento contínuo do Reconhecedor. Esta funcionalidade foi implementada recorrendo a duas *threads*, sendo uma responsável pela gravação e outra responsável pela detecção de fala/ pausa. A figura 4.8 mostra um esquema que contém a interacção e mecanismos de sincronismo entre as *threads* de gravação e detecção de fala/ pausa.

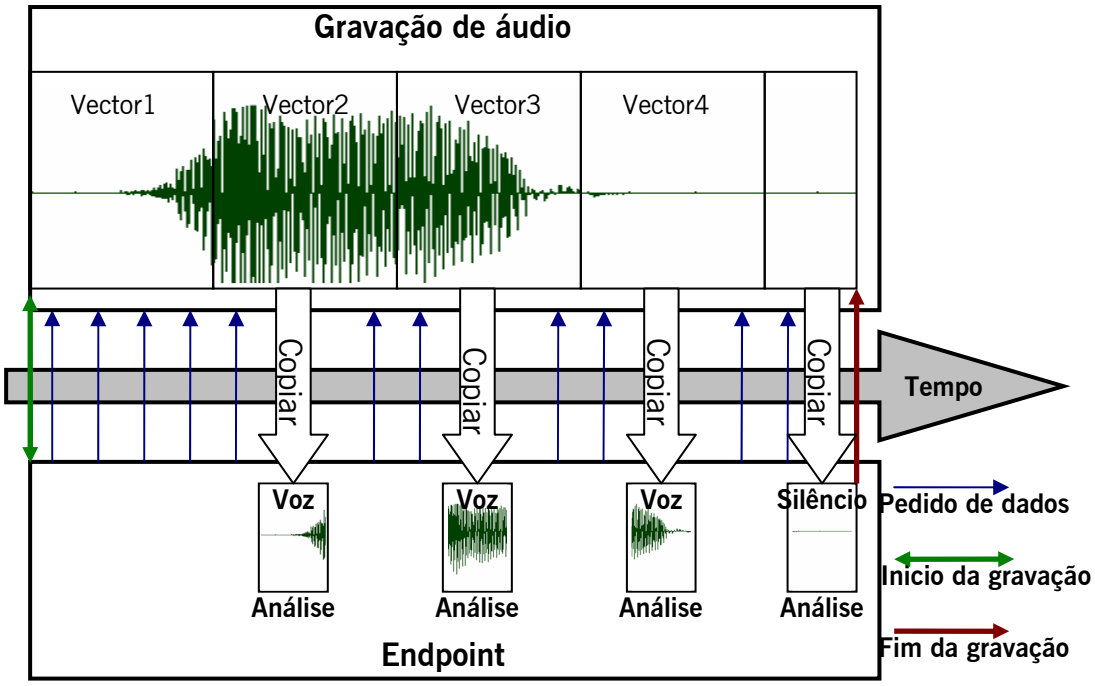


Figura 4.8 – Esquema de interacção e sincronismo entre a gravação e o Endpoint.

Pela figura 4.8, pode verificar-se que a interacção e sincronismo entre as duas *threads* são realizados do seguinte modo:

- Após a indicação do início da Gravação são criadas as threads da gravação e a *thread* do *Endpoint*.
- A gravação do áudio é guardada numa lista ligada de vectores de tamanho fixo. A utilização dessa lista permite que o tempo de gravação não tenha que ter um limite pré-definido.
- Durante a gravação a *thread* do *Endpoint* vai fazendo periodicamente (de 100 em 100 milissegundos) pedidos de dados. Porém, esses dados apenas serão fornecidos quando o vector actual estiver completo.
- Após o vector estar totalmente preenchido e permitida a cópia dos dados gravados, esses dados serão analisados através do algoritmo do *Endpoint* condicionado com valores da análise do vector anterior (por exemplo, se está numa fase de fala ou silêncio).
- Por fim, a gravação será imediatamente parada, quando na análise dos dados se detectar um silêncio demasiado prolongado para significar tempo entre palavras numa oração.

A implementação destes passos é apresentada na porção de código do exemplo 7, onde a variável *posicao* contém o endereço do último vector da lista que foi analisado. Caso entrem novos vectores na lista de gravação para analisar, facilmente serão identificados através de uma comparação directa de endereços.

Código exemplo 7. Comunicação entre a gravação de áudio e *End-point*

```
while (m_Wave->GetState()==RECORD)
{
   if(waveIN->RefeshWave(posicao))
   {
        if(Analizar())
           m_Wave->Stop();
    }else
        Sleep(100);
}
```

4.6 Tipo de *threads* utilizadas

As *threads* utilizadas neste objecto COM, excepto as relacionadas com o áudio, foram desenvolvidas no sentido de não bloquear o programa que está em interacção com o objecto quando este processa uma tarefa demorada. A grande

vantagem deste tipo de *threads* é continuar com o serviço que foi pedido inicialmente ao COM, no final de execução da sua tarefa. Após a análise do código exemplo 8, que representa uma *thread* desse tipo, é possível verificar que logo após o começo da *thread*, o COM fica numa espécie de *sleep* mas com a vantagem de processar todas as mensagens que são enviadas pela aplicação.

Código exemplo 8. *Thread* utilizada no *Endpoint*.

```
M_eventHandle = CreateEvent(NULL, TRUE, FALSE, NULL);
DWORD □hreaded;
CreateThread(NULL, 0, &ThreadEndPointer,this, 0, &□hreaded);
MSG msg;
while(::WaitForSingleObject(m_eventHandle, 0) == WAIT_TIMEOUT)
{
       //get and dispatch messages
   if(::PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
   {
       ::TranslateMessage(&msg);
       ::DispatchMessage(&msg);
   }
}
CloseHandle(m_eventHandle);
```

Todo o manuseamento de áudio (*record, play, stop*) que foi necessário para este sistema foi feito com a ajuda das *Waveform Functions* que são um conjunto de API's de Multimédia que a Microsoft disponibiliza para interagir com a placa de som.

Capítulo 5

Conclusão

Este capítulo apresenta os resultados de alguns testes que se realizaram ao sistema, bem como algumas conclusões baseadas na interpretação desses resultados. Por fim, para terminar o capítulo e este trabalho, sugerem-se algumas alterações ou complementos que podem contribuir para o melhoramento global do sistema de reconhecimento.

5.1 Resultados

A este sistema foram feitos os seguintes testes:

- Teste 1 – Verificação do desempenho do sistema com a diminuição da relação sinal ruído (RSR).
- Teste 2 – Verificação do desempenho do Reconhecedor com o aumento de padrões na biblioteca (oradores e/ou diversidade de orações).
- Teste3 – Verificação do grau de reconhecimento com o aumento de padrões que entram na fase de treino.

Os resultados destes testes foram os seguintes:

- No teste 1, verificou-se o comportamento do sistema em duas fases: a fase da segmentação da voz (*Endpoint*) e a fase do reconhecimento propriamente dito (LPC+ HMM's). O objectivo deste teste é conhecer as limitações do detector de fala no desempenho global do sistema.

O desempenho do sistema na fase de segmentação vai diminuindo lentamente com a diminuição da relação sinal/ruído até que chega a um determinado ponto que decai bruscamente. A queda inicial deve-se à perda de porções finais em algumas palavras que têm silêncios intra-palavra e terminam com pouca energia (por exemplo a palavra eight). A razão para a queda brusca advém da condição (2.10) que impõe que 50% do sinal que é considerado voz esteja fora da distribuição do ruído. (tabela 1)

O desempenho do sistema é apresentado na tabela 1, para uma base de dados em inglês, oriunda da AT&T, onde foram considerados apenas dígitos pronunciados isoladamente. A base de dados já se encontra segmentada, a fala é limpa e o ruído é *Gaussiano* gerado artificialmente.

	Fala limpa	SNR – 15dB	SNR – 10dB	SNR – 5dB	SNR – 0dB
LPC+HMM's	99.5%	56.5%	39.5%	30%	16.25%
Endpoint	100%	95%	60%	10%	0%
Sistema total	99.5%	54%	25%	3%	0%

Tabela 1 – Desempenho do Reconhecedor para diferentes SNR.

- O teste 2 teve dois objectivos, por um lado, verificar o grau de reconhecimento com o aumento dos modelos na biblioteca (modelos que serão comparados na fase de reconhecimento) e, por outro, verificar a diferença entre o reconhecimento do orador através da mesma palavra e o reconhecimento de palavras diferentes pronunciadas pelo mesmo orador.

  Os resultados que obtivemos estão na tabela 2. Da análise desses valores, que estão na tabela, podemos afirmar que o aumento dos modelos na biblioteca provoca um aumento nas falhas de reconhecimento e que, como era de esperar, o Reconhecedor confunde-se menos no reconhecimento de diferentes palavras ditas pelo mesmo orador.

Outro resultado, e que não está apresentado na tabela, é que o tempo necessário para o reconhecimento também aumenta com o número de modelos existentes na biblioteca.

	5 Modelos	10 Modelos	15 Modelos
Reconhecer orador através de modelos com mesma palavra	99%	80%	70%
Reconhecer palavras diferentes	100%	95%	90%

Tabela 2 – Desempenho do sistema com o aumento do número modelos.

- O teste 3 teve como propósito testar a influência do treino no sistema de reconhecimento. Os resultados obtidos, e mostrados na tabela 3, indicam que quanto maior é o número de padrões que entram na fase de treino, melhor é o comportamento do sistema. No entanto, todos os modelos têm que ser, aproximadamente, treinados com o mesmo número de padrões, porque se isso não acontecer corre-se o risco de serem criados super modelos (overtraining) que se sobrepõe a todos os outros e, com isso, o reconhecimento cairá sempre no orador e palavra que esses moledos representam.

	5	10	15	20
Reconhecimento em função nº de repetições no treino	60%	90%	99%	99.5%

Tabela 3 – Desempenho do sistema com o aumento do treino.

5.2 Conclusões

As conclusões que são enunciadas, de seguida, têm por base os resultados obtidos na fase de teste do sistema. Salienta-se no entanto que o conjunto de testes efectuados está ainda aquém do necessário para inferir conclusões definitivas no que respeita ao desempenho do Reconhecedor do orador. Uma das limitações deve-se ao facto da base de dados existente ter poucas repetições de orações para o mesmo orador e, também, poucos oradores. Por isso foi necessário acrescentá-la com mais oradores, mas, devido a pronunciação diferente (sotaque português) é

possível a manipulação não intencional de resultados. Contudo, chegou-se às seguintes conclusões:

- O principal inimigo deste sistema é o ruído. Este Reconhecedor, como se verificou, funciona relativamente bem para ambientes de fala limpa. Porém, para ambientes ruidosos o seu desempenho baixa significativamente. Para melhorar este sistema, nesse tipo de ambientes, terá que ser alterado o método de segmentação, pois, uma parte significativa dos erros é devido ao desempenho limitado do detector de fala/pausa nestas condições. A técnica de extracção de características também tem que ser alterada para uma das já existentes que apresentam maior robustez ao ruído.
- O treino de um modelo é a fase mais crítica de todo o sistema. Ao treinar um modelo é necessário ter em atenção a qualidade dos padrões de entrada e a quantidade de padrões que vão entrar no treino. A qualidade, porque o modelo deverá conter apenas as características das palavras e do orador e a quantidade, para que o modelo seja o mais abrangente possível, mas sempre com o cuidado de não ser criado um super modelo (overtraining). Este compromisso nem sempre é fácil de obter e tem como base aspectos de natureza prática como é o desempenho medido.
- Como se verificou, de igual modo, nos resultados dos testes, quanto mais modelos existirem na biblioteca do sistema mais confuso fica o sistema e mais demorado é o reconhecimento (o tempo de reconhecimento é directamente proporcional ao número de modelos da biblioteca), o que nos leva a concluir que este sistema será mais eficiente no reconhecimento de poucos oradores. Quanto à performance de reconhecimento, esse problema poderá ser revolvido com o reconhecimento do orador através de várias palavras (tipo password) e não com, apenas, uma única palavra (como está implementado neste sistema). Em relação ao tempo necessário para o reconhecimento, a solução está em máquinas de processamento paralelo. Estas são as razões fundamentais pelas quais se opta frequentemente pelo uso da verificação do orador em detrimento do reconhecimento do orador, visto que, para a verificação é apenas necessário comparar o modelo do orador com o modelo conjugado desse orador (modelo com todos os oradores

excepto esse orador), o tempo de verificação do orador é, aproximadamente, sempre o mesmo.

- O reconhecimento de palavras diferentes com o mesmo orador obteve melhores resultados do que o reconhecimento do orador através de uma única palavra. Da análise destes resultados juntamente com a conclusão anterior (que indica que o Reconhecedor funciona melhor com bibliotecas relativamente pequenas) permite-nos concluir que este sistema poderá ser útil no reconhecimento de comandos isolados (por exemplo em domótica: acender luzes, abrir portas, etc., tudo isso ajustados ao orador).

5.3 Trabalho Futuro

O trabalho futuro centra-se essencialmente em três aspectos:

- **Robustez do detector de fala/pausa ("*Endpoint* detector")**

  Uma parte significativa dos erros, em reconhecimento automático da fala especialmente na modalidade ISR, tem a sua origem em erros de detecção dos segmentos de fala e não em "confusão" entre orações. Basicamente, o sistema confunde fala e ruído e depois não consegue reconhecer correctamente a fala que afinal está ausente. Este é um assunto ainda não completamente resolvido especialmente em aplicações práticas. Sugerimos, a modificação em 2 aspectos fundamentais:

  - Modelar a estatística conjunta do número de passagens por zero ("Zero-crossing rate") e da energia do sinal, uma vez que estas técnicas já são usadas isoladamente com as vantagens e desvantagens de cada uma.
  - b) Usar um classificador neuronal, baseado em Funções de Base Radial, treinado com uma grande diversidade de situações de fala com ruído.

- **Robustez na extracção de características**

  Dado que os coeficientes de predição linear (LPC) não apresentam grande robustez ao ruído, sugere-se como trabalho futuro implementar outras técnicas (MFCC, PLP, Wavelets)

- **Robustez na modelização da fala com ruído**

  Não se conhece nenhuma técnica de extracção de características com robustez suficiente ao ruído, que possa por si só garantir um desempenho de reconhecimento aceitável em condições ruidosas. Uma das abordagens possíveis para melhorar o desempenho de reconhecedores de fala no geral é modelar conjuntamente a fala e o ruído, dado que um modelo para o ruído pode ser estimado a partir de segmentos de silêncio classificados pelo detector fala/pausa (ruído de fundo). Os modelos de fala existem, sendo por isso possível combinar ambos os modelos numa função densidade de probabilidade conjunta que é um HMM se ambos, fala e ruído, forem modelados, de igual modo, por um HMM. Esta é a abordagem que vamos seguir no desenvolvimento do módulo de reconhecimento/verificação do orador a disponibilizar no âmbito do projecto TECNOVOZ.

Referências bibliográficas

[1] G. Saha1, Sandipan Chakroborty, Suman Senapati "A New Silence Removal and Endpoint Detection Algorithm for Speech and Speaker Recognition Applications".

[2] Lawrence R. Rabiner, Ronald W. Schafer, "Digital Processing of Speech Signals", Prentice-Hall Series in Signal Processing.

[3] Carlos Manuel Gregório Santos Lima, "A Robust Feature Extraction for Automatic Speech Recognition in Noisy Environments".

[4] Aria Ansari, COM from scratch, http://www.codeproject.com/com/COM_from_scratch_1.asp, 2004.

[5] Linear Predictive Coding, Jeremy Bradbury, December 5, 2000

[6] Lawrence R. Rabiner, Ronald W. Schafer, Digital Processing of Speech Signals, Prentice-Hall Series in Signal Processing

[7] Adriano Tavares, Component Object Model

[8] www.tecnovoz.pt, 2007

[9] www.voiceverified.com, 2007

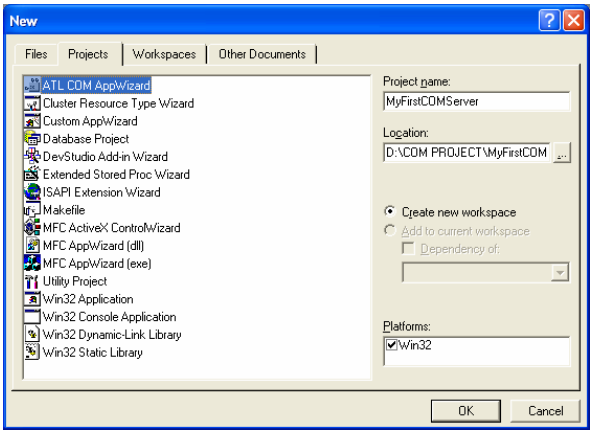
[10] www.voicevault.com, 2007

[11] "Microsoft Developer Network", Http://msdn2.microsoft.com/

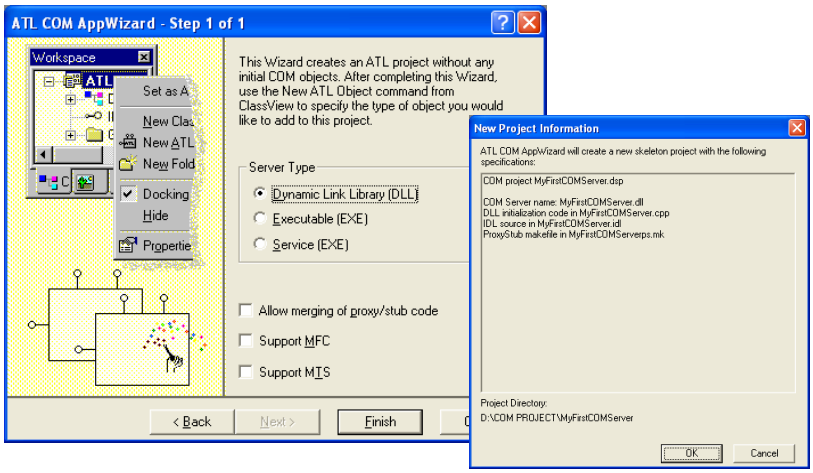
Anexo I – Criação de um objecto COM rapidamente através do visual c++

Este anexo apresenta, passo a passo, a criação de um *objecto COM* utilizando o visual C++.

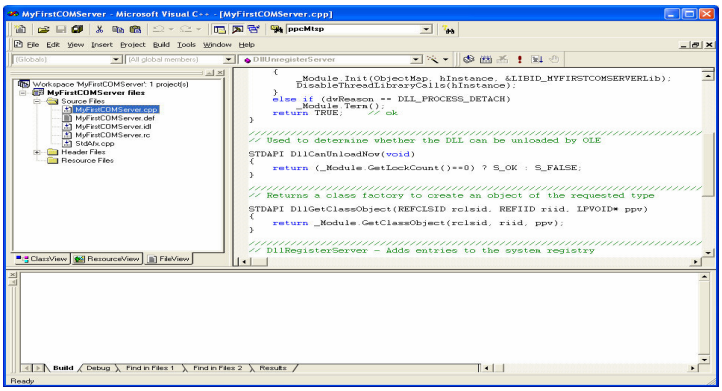
- **Criar um novo projecto VC++ e seleccionar "ATL COM AppWizard"**



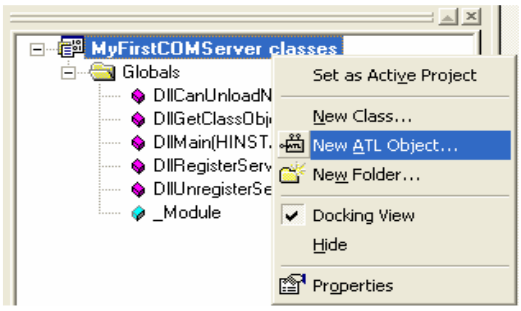
- **Seleccionar um Servidor baseado em DLL (In-Process) com as seguintes opções e premir *Finish***



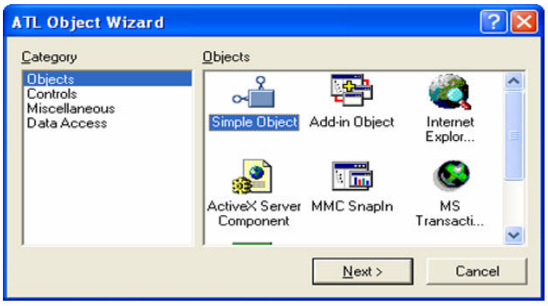
- **Antes de adicionar um objecto COM ao servidor ir ao "Workspace View" e analisar cada um dos ficheiros**



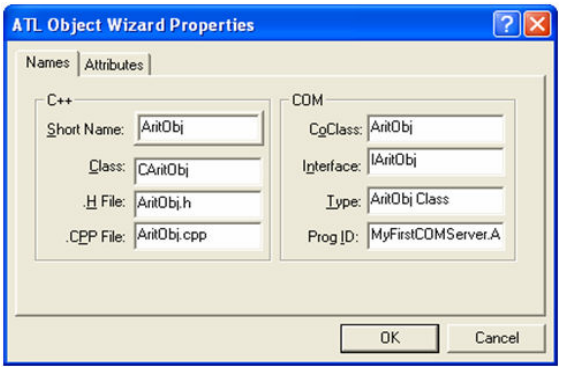
- **Criar um objecto COM**
  - Com o botão direito do rato



  - Seleccionar "*Simple object*" e premir "*Next*"

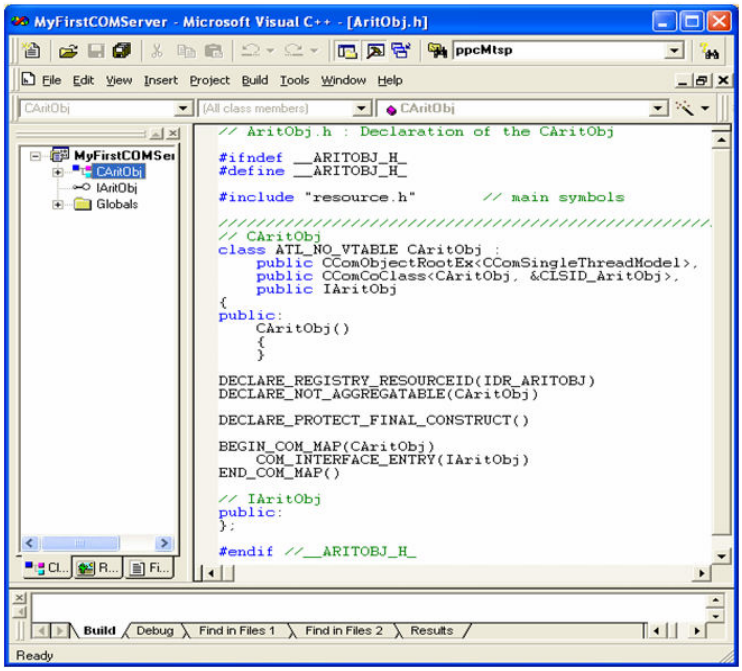


  - Introduzir o nome "*AritObj*" e premir "*OK*"

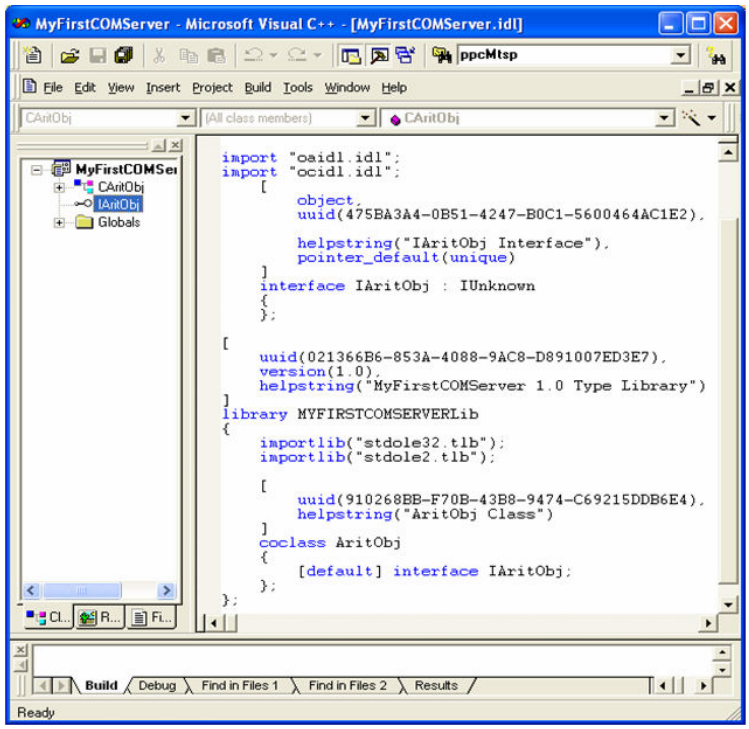


  - Seleccionar os seguintes "*attributes*" e premir "*OK*"

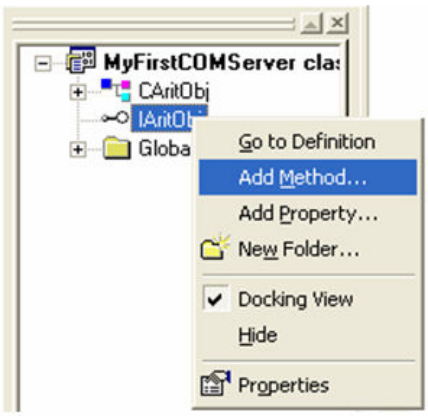
- **Antes de adicionar um método ao servidor analisar o código do Objecto COM e da interface**
    - Ficheiro da coclass



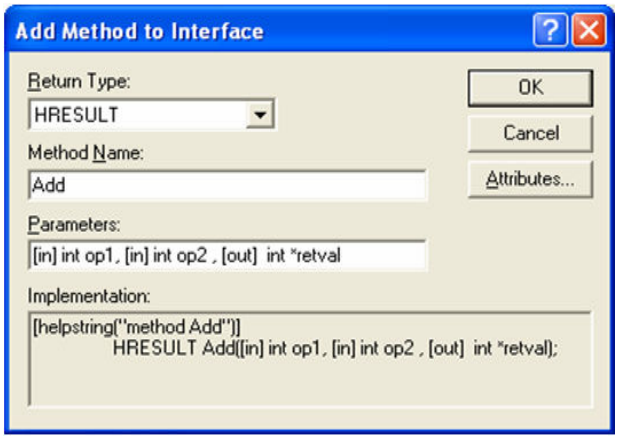
    - Ficheiro IDL do interface



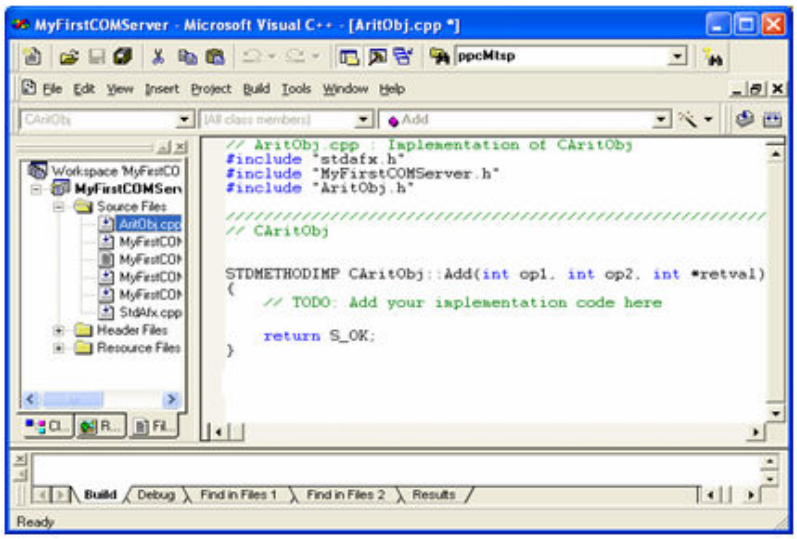
- **Adicionar um método ao servidor COM**
  - Com o botão direito do rato



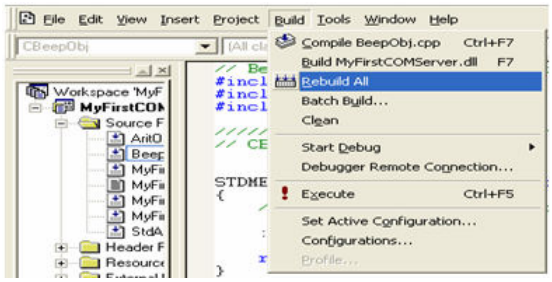
  - Atribuir o nome e parâmetros ao método e premir OK



  - Abrir os ficheiros de interface (MyFirstCOMServer.idl) e da coClass (AritObj.cpp e AritObj.h) para ver as alterações introduzidas



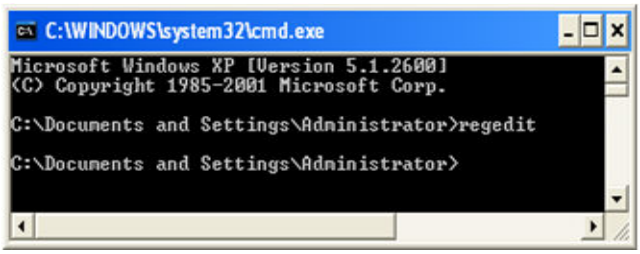
- **Salvaguardar todos os ficheiros fazendo "*build project*"**



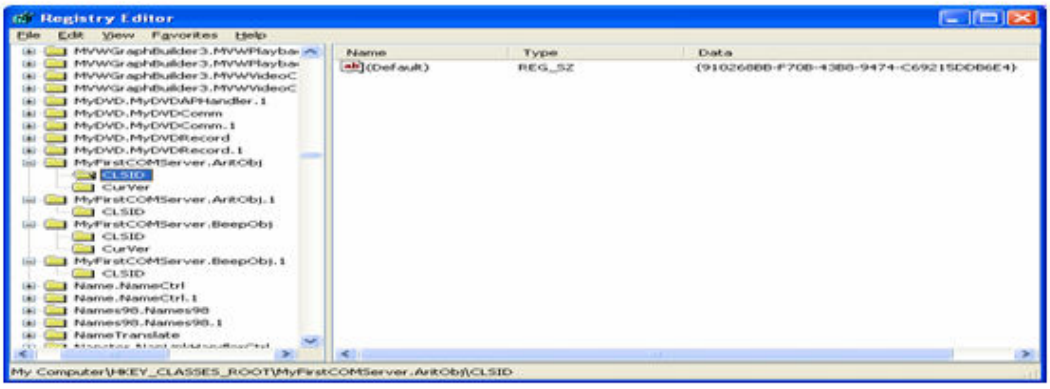
Developer Studio efectuou os seguintes passos:

  - Executou o compilador MIDL para gerar código e *type libraries*
  - Compilou os ficheiros fontes
  - Na "linkagem" criou a MyFirstCOMServer.dll
  - Registou os componentes COM
  - Registou a DLL com **RegSvr32** para que possa ser automaticamente carregada quando necessário

- **Analisar o Registry para certificar se de facto o servidor COM foi instalado:**
  - Do prompt do DOS executar Regedit



  - Seleccionar HKEY_CLASSES_ROOT\CLSID e mandar pesquisar por MyFirstCOMServer



Anexo II – Funcionamento do sistema através do *software* de interface

O *software* de interacção com o objecto COM que foi desenvolvido permite, apenas, testar os dois serviços, treino e reconhecimento, que objecto COM disponibiliza através dos seus dois interfaces.

De seguida, apresenta-se detalhadamente a realização do treino e do reconhecimento, consecutivamente, a partir deste *software* de interface.

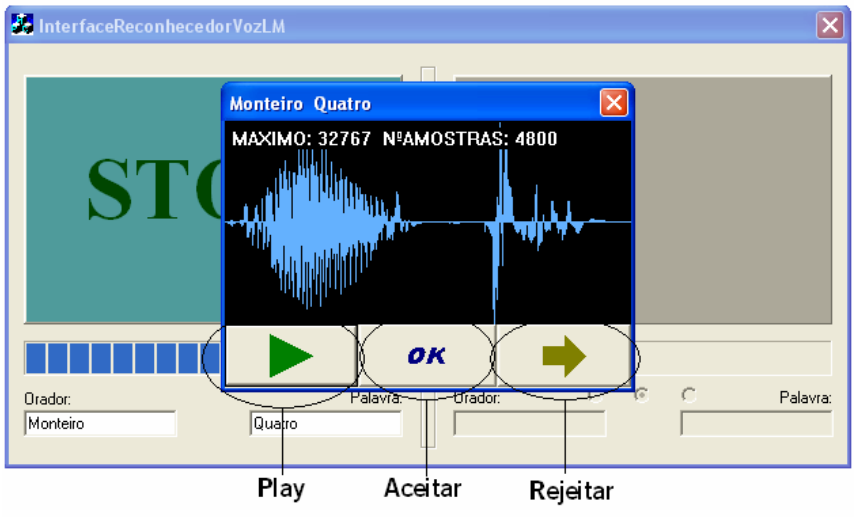
Treino

O módulo de treino pressupõe o conhecimento da palavra e do nome do orador para poder efectuar o treino do orador dependente do texto. A invocação deste módulo passa pelas seguintes etapas:

- Escrever nos campos respectivos o nome no orador e a palavra que quer treinar.

- Premir o botão TREINO e repetir algumas vezes a palavra a treinar (por exemplo dizer um dado número de vezes a palavra quatro). O detector de fala /pausa contará automaticamente o número de palavras.

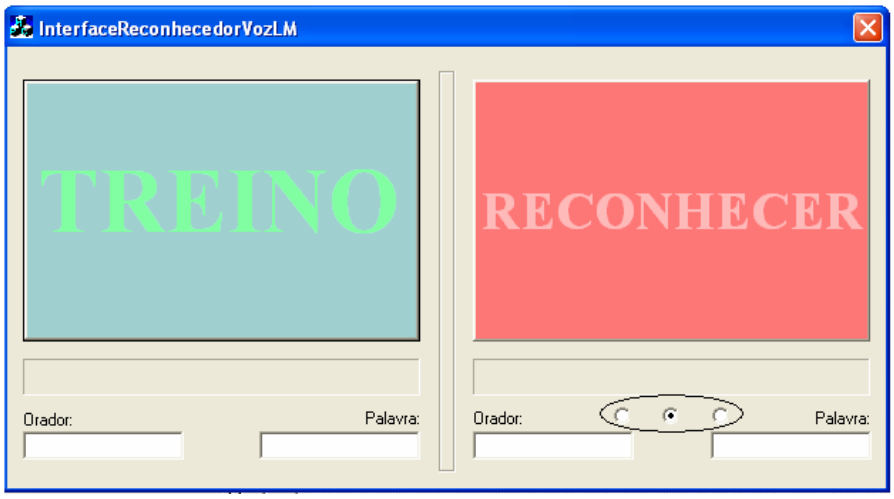
- Após dois segundos de silêncio aparecerá a janela que permitirá a selecção das palavras que entrarão no treino. O objectivo desta funcionalidade é garantir que apenas as palavras perfeitas, seleccionadas pelo detector fala/pausa, possam entrar no treino. Por exemplo palavras cortadas ou ruídos não deverão entrar no treino



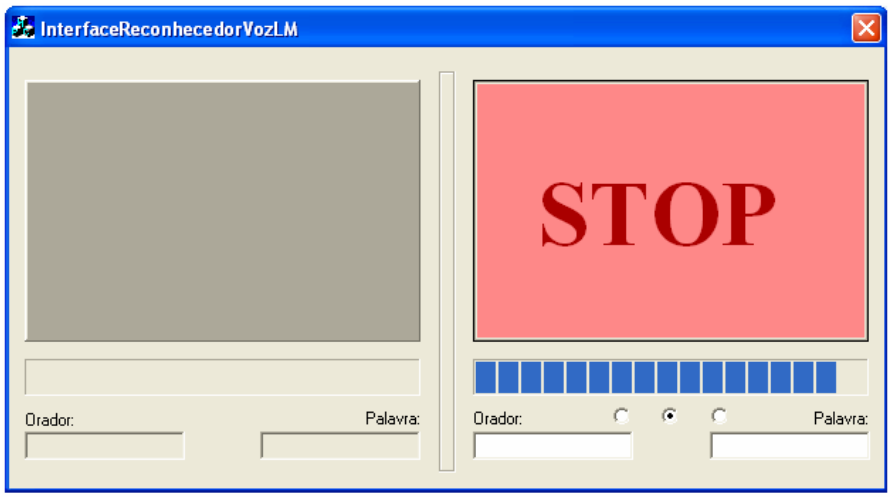
- Por fim, o sistema através da COM realiza o treino. Esse treino tem um tempo indefinido e só quando a janela principal deste *software* ficar novamente activa é que se tem a indicação de final de treino.

Reconhecimento

- Esta aplicação permite escolher o grupo de modelos que vão ser comparados na fase de reconhecimento com o objectivo do utilizador testar o sistema para cada uma dessas situações. O reconhecimento, então, poderá ser feito através da comparação do padrão a reconhecer com todos os modelos, apenas com modelos que contenham uma determinada palavra ou só com modelos de palavras de um determinado orador. Dado isto, o primeiro passo é escolher um grupo de três possíveis através de três selectores (rádio buttons).



- Premir o botão reconhecer e pronunciar a palavra.



- Após um segundo de silêncio, aparecerá nos campos respectivos o resultado do reconhecimento.