UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIAS DA COMPUTAÇÃO
(Bacharelado)

CLASSES AUTOMATION OLE PARA RECONHECIMENTO DE FALA ATRAVÉS DA REDE NEURAL ARTIFICIAL RBF-FUZZY ARTMAP

RELATÓRIO DO TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS DE DISCIPLINA COM O NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO BACHARELADO

ARNOLDO UBER JUNIOR

BLUMENAU, JUNHO/2001

2001/1-09

CLASSES AUTOMATION OLE PARA RECONHECIMENTO DE FALA ATRAVÉS DA REDE NEURAL ARTIFICIAL RBF-FUZZY ARTMAP

ARNOLDO UBER JUNIOR

ESTE TRABALHO DE CONCLUSÃO DE CURSO FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS DA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE :

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Marcel Hugo, M. Eng. – Orientador na Furb

_____

Prof. José Roque Voltolini da Silva – Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Marcel Hugo, M. Eng.

_____

Prof. Maurício Capobianco Lopes, M. Eng.

_____

Prof. Roberto Heinzle, M. Eng.

AGRADECIMENTOS

Aos meus pais Anselmo e Olga, aos meus irmãos José e Jacqueline, pelo apoio, compreensão e dedicação oferecidos a mim durante toda minha vida. E a minha namorada Luciane, pelos mesmos motivos durante o tempo em que a conheço.

Agradeço ao meu amigo, orientador e Prof. Marcel Hugo, pela confiança, conhecimento e perseverança empregados na elaboração deste trabalho.

Ao Dr. Gerson Tontini por ser o idealizador da RNA *RBF-Fuzzy Artmap*, além dos esclarecimentos práticos e teóricos durante o desenvolvimento deste trabalho, e ao Dr. Malcon Anderson Tafner, pelo tempo disponibilizado a mim no esclarecimento de dúvidas referentes a aquisição de som e pré-processamento de sinal.

À todas as pessoas que fizeram parte da minha vida acadêmica, aos professores, amigos e em especial, aos integrandes dos grupos “Amigos do Barney” (www.amigosdobarney.com.br) e “Os Mano” (www.osmano.com.br).

À Universidade Regional de Blumenau, por disponibilizar à comunidade Blumenauense e região, o curso de Bacharel em Ciências da Computação, além da oportunidade de desenvolver este trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

ABREVIATURAS

RNA – Rede Neural Artificial

RBF – Rede de Funções de Base Radial

ART – Teoria da Ressonância Adaptiva (*Adaptive Ressonance Theory*)

RESUMO

Este trabalho tem como objetivo facilitar a utilização de redes neurais artificiais (RNA) aplicadas ao reconhecimento de fala no desenvolvimento de software. Será possível ao desenvolvedor, treinar a RNA (modelo *RBF-Fuzzy Artmap*) com as palavras que deseja. Gerar uma base de conhecimento resultante do treinamento da RNA e através de uma interface *Automation OLE* (*Object Linking and Embedding*), incorporar esta tecnologia em seus aplicativos. A interface *Automation OLE* acionará à RNA quando necessitar reconhecer uma palavra falada, tendo como objetivo, a execução de uma determinada ação. Para demonstrar a utilização das classes desenvolvidas, foi especificado como exemplo, um aplicativo de reconhecimento de fala que as utiliza, efetuando ações acionadas pela pronúncia dos números de 0 (zero) a 9 (nove) pelo usuário.

ABSTRACT

This work has as objective facilitates the use of artificial neural network (ANN) applied to the speech recognition in the software development. It will be possible to the developer, to train ANN (*RBF-Fuzzy Artmap* model) will be used to workout, with the words that he wants. To generate a knowledge base resulting from the training of ANN and through an interface Automation OLE (Object Linking and Embedding), to incorporate this technology in their applications. The interface Automation OLE will work ANN when he needs to recognize a spoken word, tends as objective, the execution of a certain action. To demonstrate the use of the developed classes, it was specified as example, an application of speech recognition that uses them, making actions worked by the pronunciation of the numbers of 0 (zero) at 9 (nine) for the user.

1. INTRODUÇÃO

1.1 TEMA

Com o passar do tempo, a informática foi se aperfeiçoando cada vez mais, ao ponto de facilitar muitas atividades diárias do ser humano, e até mesmo, tornar-se indispensável para realizar algumas tarefas (Bruns, 1995).

Os computadores ficaram mais populares em muitas partes da sociedade, tornando-se claro que a maioria das pessoas possuem grande dificuldade de aprendizado em utilizá-los, principalmente caso não tenham nenhum conhecimento em informática. Os usuários têm dificuldades para a manipulação dos programas e precisam aprender comandos que para eles não tem significado algum para poderem usufruir de seus softwares. Além disso, essa interação homem - computador é feita lentamente, por meio de teclados ou mouses. Torna-se quase evidente que é preciso de um método mais rápido para fazer esta interação, sendo então proposto utilizar um software que reconheça a fala do usuário. Desta forma o usuário poderá interagir com seu computador através de comandos de fala.

Há pouco tempo, a maioria das pessoas dificilmente imaginaria que em algum dia seria possível fazer com que palavras faladas por um ser humano pudessem ser interpretadas por um computador. Mesmo porque tanto o hardware, como os softwares existentes em tal época, eram muito mais deficientes e pobres em tecnologia. Realizaram-se estudos, desenvolveram-se novas técnicas, aprofundaram-se os conhecimentos. Houve uma evolução e um aperfeiçoamento na informática. Os computadores e os sistemas tornaram-se mais complexos e observou-se que tais sistemas poderiam ser muito mais expandidos caso fosse utilizada a fala humana como meio de entrada e saída (Shapiro, 1987 apud Klabunde, 1996).

Muitas pesquisas realizadas para o reconhecimento de padrões, entre eles o reconhecimento de fala, estão direcionadas para o uso de Redes Neurais Artificiais (RNA), que são sistemas computacionais de implementação em hardware ou software que imitam

as habilidades computacionais do sistema nervoso biológico, usando um grande número de simples neurônios artificiais interconectados (Loesch, 1996).

Contudo, não faz parte do cotidiano dos desenvolvedores de software a implementação de RNA e seu uso em diversos aplicativos. Assim, pretende-se fazer com que um desenvolvedor (chamado aqui de usuário) possa incorporar classes de RNA de forma simples em sua aplicação, utilizando a tecnologia *Automation Object Linking and Embedding* (OLE).

Este trabalho tem como objetivo facilitar a utilização de redes neurais artificiais (RNA) para reconhecimento de fala no desenvolvimento de software, possibilitando ao desenvolvedor treinar uma rede com as características desejadas. Neste trabalho será utilizada a RNA Rede de Funções com Base Radial (RBF) *Fuzzy Artmap* (Tontini, 1995). Após treinada a RNA, o desenvolvedor irá gerar a base de conhecimento resultante do treinamento da RNA na ferramenta que será criada, e através de uma interface *Automation* OLE, incorporar está tecnologia em seus aplicativos, que acionarão a RNA, quando necessitarem.

Para demonstrar: a utilização da ferramenta criada, o desenvolvimento utilizando as classes resultantes deste trabalho de pesquisa, a interface em aplicativo e o reconhecimento de fala em si, será desenvolvido um aplicativo exemplo utilizando as tecnologias envolvidas, que responderá a comandos de fala, sendo os números de 0 (zero) a 9 (nove), desempenhando funções apenas demonstrativas.

Será utilizado para o desenvolvimento deste, o conhecimento e experiência relatados nos trabalhos de pesquisa que já realizaram implementações de RNA para o reconhecimento da fala de Bruns (1995), Starke (1996), Fischer (1999) e Klabunde (1996).

Para o desenvolvimento da ferramenta, objetos *Automation* OLE e aplicativo exemplo será utilizada a linguagem de modelagem orientada à objetos UML (*Unified Modeling Language*).

Será tomado como ambiente de desenvolvimento o Delphi, utilizando-se de *Object* Pascal como linguagem de programação, não esquecendo da utilização das *APIs* do Sistema Operacional Windows 2000 e da utilização de *Automation* OLE para possibilitar

portabilidade e por ser crescente a sua aceitação nos ambientes e linguagens existentes atualmente.

1.2 JUSTIFICATIVA

Desde que surgiu, a multimídia vem conquistando espaço no dia-a-dia das pessoas que se beneficiam de um computador. A grande vantagem da multimídia vem ao encontro das necessidades daqueles que têm dificuldades na utilização desses computadores. Softwares que utilizam recursos multimídia quase sempre são bem aceitos por qualquer tipo de usuário – os leigos, bem como os mais experientes, de acordo com Bruns (1995).

Pelo simples fato de quebrar a monotonia e estaticidade de uma interface simples – por exemplo uma interface em modo “caracter” – a multimídia já pode ser considerada um trunfo nas mãos dos desenvolvedores de software. Recursos multimídia dão ao usuário uma visão mais real ou mais direta do uso do software, de acordo com Bruns (1995).

Se interfaces de voz vêm sendo desenvolvidas e aprimoradas como um novo e fácil meio de entrada e saída de informações em um computador, é de extrema importância o domínio da técnica de reconhecimento de fala, pois logo ela se tornará um fator essencial para a competitividade de qualquer software no mercado mundial, de acordo com Hugo (1992).

1.3 OBJETIVOS

1.3.1 OBJETIVO PRINCIPAL

O objetivo principal deste trabalho é facilitar a utilização de RNA aplicadas ao reconhecimento de fala no desenvolvimento de software. Assim será possível ao desenvolvedor treinar a RNA disponível na ferramenta de treinamento, que é a RNA *RBF-Fuzzy Artmap*, com as palavras que deseja. Gerar uma base de conhecimento resultante do treinamento da RNA; e através de uma interface *Automation* OLE, incorporar esta

tecnologia em seus aplicativos, que acionarão a RNA, quando necessitar reconhecer uma palavra falada, para a execução de uma determinada ação.

1.3.2 OBJETIVOS ESPECÍFICOS

Como objetivos específicos pode-se citar:

a) desenvolver classes que fazem o interfaceamento entre a base de conhecimento e a aplicação do usuário;
b) desenvolver uma ferramenta para treinamento da RNA;
c) desenvolver um protótipo demonstração do uso das classes e da ferramenta de treinamento;
d) apresentar um estudo referente à técnica implementada na ferramenta com vantagens e desvantagens da RNA, sendo que para este trabalho será implementada a técnica RBF – Fuzzy Artmap, que será disponibilizada na ferramenta;
e) possibilitar a um desenvolvedor leigo em RNA e reconhecimento de fala, adicionar estes recursos em suas aplicações criando uma interface mais amigável com o seu usuário.

1.4 ESTRUTURA DO TRABALHO

Para facilitar o entendimento deste trabalho de pesquisa pelo leitor, este foi estruturado em 9 capítulos, seguindo uma ordem progressiva dos assuntos estudados para o desenvolvimento do mesmo, conforme descrição que segue.

No primeiro capítulo, encontram-se informações referentes ao tema estudado, importância deste estudo e à que objetivos chegou.

O segundo capítulo apresenta uma fundamentação teórica relacionada a fala, voz humana e ao reconhecimento de fala com conceitos, características e aplicações.

No terceiro capítulo são apresentados conceitos de Redes Neurais Artificiais (RNA), características, vantagens, desvantagens e técnicas já estudadas por trabalhos anteriores.

A RNA utilizada neste trabalho será descrita no quarto capítulo. Será mencionada a sua formação partindo-se do primeiro modelo que a compõe até à *RBF-Fuzzy Artmap*, com suas vantagens e desvantagens.

O quinto capítulo descreve o pré-processamento utilizado neste trabalho, o pré-processamento por médias, que através de suas etapas processando sinais, alimentará à RNA *RBF-Fuzzy Artmap*.

O sexto capítulo foi destinado a conceitos e forma de utilização da tecnologia *Automation OLE* para possibilitar a independência de linguagem as classes criadas (RNA e aquisição de sinal-pré-processamento).

No sétimo capítulo é esclarecido o desenvolvimento, utilização e dinâmica do protótipo, além da sua análise orientada a objetos.

No oitavo capítulo encontra-se uma análise dos resultados obtidos com este trabalho de pesquisa, revelando a performance da RNA.

O nono capítulo expoe as conclusões obtidas, limitações e sugestões para trabalhos futuros.

Finalizando encontram-se as referências bibliográficas utilizadas neste trabalho de pesquisa.

2. RECONHECIMENTO DE FALA

No transcorrer deste capítulo serão vistos conceitos relacionados ao reconhecimento de fala como a voz humana, a palavra e a fala além de abordar o reconhecimento de fala com suas fases, problemas, vantagens e desvantagens.

2.1 VOZ HUMANA

A voz é um elemento da linguagem; é a produção, que o ser humano faz, de sons, através das cordas vocais. É o elemento sonoro da comunicação (Amorim, 1972).

Nos animais, a voz é produzida de maneira simples e rudimentar. Já os seres humanos possuem a capacidade de modificar, acomodar ou diferenciar os sons emitidos, dando-lhes significado. A está articulação, dá-se o nome de fala. A voz é um elemento físico da linguagem e a sua base é anatomofisiológica.

Os órgãos que trabalham para a fonação são chamados fonoarticulatórios. Não existe um aparelho destinado à fonação, com células, tecidos e órgãos diferenciados, especificamente, como acontece para a digestão e respiração. Os órgãos destes últimos aparelhos se adaptaram à função de falar, graças a emissões nervosas que ali estimulam a produção de sons.

Normalmente a voz é produzida numa expiração quando os pulmões expelem o ar, fazendo vibrar as cordas vocais, resultando um som que vai ser ampliado pela ressonância das cavidades ósseas da face. Anormalmente a voz é produzida pelo acúmulo de ar no esôfago, que é expelido em forma de eructação (arroto) e adaptado à articulação, conhecido também como voz esofágica (Amorim, 1972).

A voz, como todo som, é o resultado da vibração das partículas da matéria, portanto possui qualidades como: intensidade, altura, timbre, emissão e ressonância (Amorim, 1981).

A intensidade é o volume de sonoridade que o organismo produz e pode ser mais ou menos intenso, dependendo da quantidade de energia e do ambiente onde é produzido (Amorim, 1981).

Existe uma relação entre intensidade vocal, capacidade pulmonar e músculos torácicos, podendo-se através de cultivo de exercícios respiratórios, melhorar o volume da emissão vocal (Amorim, 1981).

O volume da voz depende muito da respiração, pois quando esta é correta e profunda, evita o cansaço e dispêndio de energia (Amorim, 1981).

A altura é a tonalidade e se apresenta, nas vozes, uma variação desde o som mais grave até o mais agudo. Cada indivíduo possui o seu tom natural, entretanto, nem sempre esse tom natural e verdadeiro é empregado efetivamente. As vozes masculinas se caracterizam pelos sons graves ou registro de peito predominante, enquanto que as femininas se caracterizam pelos sons agudos, com supressão da ressonância peitoral (Amorim, 1981).

Com referência ao timbre, assim como a pessoa tem sua fisionomia e características próprias individualizadas, o som da voz possui o timbre: aspecto diferenciador das fontes sonoras. Cada pessoa possui um determinado timbre devido a formação de seu aparelho fonador (Amorim, 1981).

Emitir a voz é expelir o ar dos pulmões com sonorização, após ter vibrado as cordas vocais (Amorim, 1981).

Algumas pessoas emitem normalmente o som, outras o fazem com suavidade e algumas com tanta violência que a voz é sonorizada bruscamente, num verdadeiro golpe inesperado nas cordas vocais. A emissão normal é aquela não cansativa e que produz um brilho especial no timbre, resultando numa voz de boa qualidade porque fisiologicamente não é cansativa e não agride sonoramente os ouvintes. Trata-se de uma emissão que resulta de uma correta respiração funcional, com ampliação sonora nos ressoadores (Amorim, 1981).

O contrário da emissão impostada ou colocada fisicamente, é a emissão gutural, aquela que sobrecarrega a laringe e provoca cansaço vocal.

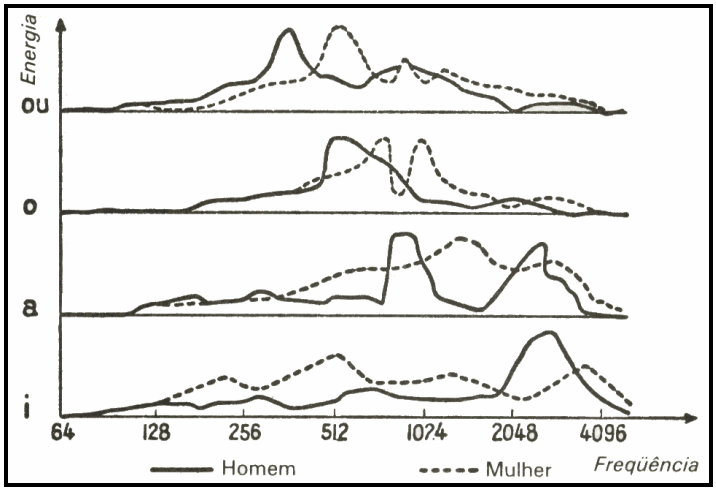
2.2 PALAVRA

Cada som que sai da boca é composto de dois períodos transitórios (períodos de estabelecimento e de extinção do som) e de um período de estabilidade. O período de estabilidade chama-se vogal e os períodos transitórios, quando são perceptíveis ao ouvido, o que nem sempre acontece, chamam-se consoantes (Matras, 1995).

As vogais são produzidas pela vibração das cordas vocais: essa vibração abre ou fecha alternadamente a fenda situada na extremidade da traquéia, na laringe, o que produz uma série de ondas sonoras na garganta. Os ressoadores, que constituem as cavidades vocais do nariz, da garganta e da boca, favorecem certas freqüências variáveis, de acordo com a vogal emitida e lhes expressam sua curva característica de freqüência (Matras, 1995).

As curvas acima na figura 2.1 mostram que todas as vogais têm, pelo menos, duas regiões de freqüência de ressonância, que os sons femininos distinguem-se dos sons masculinos tanto pela altura quanto pelo timbre e que as freqüências utilizadas pelas vogais não ultrapassam 5000 Hz (Matras, 1995).

Figura 2.1 – Energia das vogais em função da freqüência



Fonte: Matras (1995)

As consoantes dividem-se em consoantes vogais (g, r guturais), produzidas pelas cordas vocais e cuja natureza lembra as vogais, exceto no que concerne as suas freqüências, que são mais elevadas, e seus níveis, que são mais baixos; e em consoantes não vogais (p, k, f, s, ...), produzidas exclusivamente pela fricção do ar nos lábios, na

língua, nos dentes, no palato. Essas consoantes têm um número de freqüências de ressonância variável (3 para as consoantes nasais); essas freqüências são particularmente elevadas e os níveis fracos (Matras, 1995).

2.3 FALA

A fala é a produção dos fonemas com articulação na palavra, onde articulação é o contato que os órgãos fonatórios tem, ora nesta região, ora na região bucal, resultando em sons diferentes que se combinam formando a comunicação verbal (Amorim, 1972).

Não se deve confundir a fala com a voz, nem com a linguagem. A voz é simplesmente a sonoridade, já a fala, é a voz articulada. A linguagem oral é a transmissão de conceitos através da voz e da fala (Amorim, 1972).

Na fala são muito importantes não só as regiões de articulação e a correta posição dos órgãos fonoarticulatórios, como também a entonação, ou seja, a variação de tons que vai ocorrendo no momento da comunicação verbal. (Amorim, 1972).

2.4 INTRODUÇÃO AO RECONHECIMENTO DE FALA

Com a evolução tecnológica, o homem rodeou-se de máquinas e equipamentos dos mais diversos tipos, utilizando-os para todos os meios e fins possíveis, seja produção, desenvolvimento, pesquisa, administração, etc. Geralmente a interação do homem com suas tecnologias é feita de uma maneira um tanto fria, às vezes exigindo desenvolver habilidades específicas para operar determinado equipamento. Visando minimizar essa frieza de interação e facilitá-la também, existem várias pesquisas que envolvem o processo de comunicação entre homem e máquina através da linguagem natural: a fala (Dalabrida, 1999).

Através dos tempos, surgiu um ramo da ciência da computação chamado Inteligência Artificial, e dentro desta o conceito de Redes Neurais, que tem como capacidade principal imitar o comportamento do cérebro humano seja no aprendizado e reconhecimento de padrões, como é a linguagem humana. Assim, foi possível para os cientistas utilizarem esses conceitos para o reconhecimento da fala (Dalabrida, 1999).

A utilização da fala para efetuar a interação homem-máquina apresenta uma série de vantagens, primeiramente por tornar a tarefa fácil e natural, já que não necessita habilidade ou experiência. Além disso, é um meio rápido de aquisição de informações, permite a execução de outras tarefas simultaneamente, deixa as mãos e os olhos livres e ainda, pode permitir a locomoção do usuário. O reconhecimento automático da fala é o processo de extração automática da informação lingüística do sinal de voz.

A busca da interface de reconhecimento de fala começou por volta de 1950, com o advento do computador digital. Combinada com ferramentas de capturar e analisar a fala, como conversores analógico-digital e espectrogramas de som, o computador permitiu aos pesquisadores, procurar modos para extrair características da fala que permite a distinção entre palavras diferentes (Baeker, 2000).

Na década de 1960, desenvolveu-se a segmentação automática de fala em unidades de relevância lingüística (como fonemas, sílabas, e palavras), um novo padrão de emparelhamento e também algoritmos de classificação (Baeker, 2000).

Na década de 1970, várias técnicas importantes (tabela 2.1), que hoje são essenciais ao atual “estado da arte” do reconhecimento de fala foram desenvolvidas, grande parte delas no projeto de reconhecimento de fala da Agência de Projetos de Pesquisa Avançados de Defesa (D.A.R.P.A.) (Baeker, 2000).

Tabela 2.1 – Principais sistemas de reconhecimento de fala antes das redes neurais

Sistema	Orador Independente	Orador Contínuo	Vocabulário	Reconhecimento
NTT	Não	Não	200	97.50%
DRAGON	Não	Sim	194	84.00%
HEARSAY II	Não	Sim	1011	87.00%
HARPY	Não	Sim	1011	97.00%
BELL 1	Sim	Não	129	91.00%
FEATURE	Sim	Não	26	90.00%
TANGORA	Não	Sim	5000	97.00%
BYBLOS	Não	Sim	997	93.00%
SPHINX	Sim	Sim	997	96.00%
VOCING	Sim	Sim	Independente	96.00%

Fonte: Kitano (1994)

O reconhecimento de fala teve uma evolução significativa, desde que a D.A.R.P.A. começou a apoiar a pesquisa de reconhecimento de fala em 1971. Foram desenvolvidos inúmeros sistemas, e foram propostas várias aproximações (Kitano, 1994).

Os primeiros sistemas de vocabulários grandes que apareceram nos anos 70 são exemplificados por Hersay II e Harpy. O Hersay II, era um sistema que utilizou uma arquitetura de tábua preta para atingir interação dinâmica de várias fontes de conhecimento. O sistema de Harpy integrou a representação da rede usada em Dragon e a da técnica de procura Beam (Kitano, 1994).

A *Dynamic Time Warp* (DTW) foi proposta em 1975. Já em 1980, uma série de sistemas independentes de orador foram desenvolvidos como o sistema Feature. O sistema Feature está baseado nas características (*features*) do espectro do som das palavras. Na metade da década de 1980, um grupo de pesquisa da IBM desenvolveu o sistema Tangora, que possui 5000 vocábulos e trabalha em linguagem natural. Este foi o primeiro sistema a trabalhar com interface natural (Kitano, 1994).

Os anos foram passando e muitos problemas tecnológicos foram sendo superados. No final da década de 80, o mundo finalmente, conheceria os micro-computadores, os disquetes, os winchesters e, também, a conversão analógico-digital, tecnologia proporcionada por placas dedicadas e recurso essencial para o processamento do sinal de som digitalizado. Essa revolução tecnológica acabou afetando muitas outras áreas da computação, e por tabela, outras ciências de interesse (Tafner, 1996).

Essa revolução não foi diferente para o reconhecimento de fala. A globalização e a popularização dos computadores aumentou significativamente o número de pessoas pesquisando técnicas e tecnologias diversas. Contudo, apesar de toda a tecnologia disponível, um dos maiores trunfos ainda estava por vir. O ressurgimento das redes neurais artificias no final da década de 80 deu um novo impulso à tecnologia do reconhecimento de padrões (Tafner, 1996).

No fim da década de 1980, os sistemas de reconhecimento de fala Sphinx e Byblos foram desenvolvidos, utilizando os *Hidden Markov Models* (HMM). O sistema Sphinx foi estendido a um vocabulário independente e passou a se chamar Vocind (Kitano, 1994).

O modelo de processamento HMM foi extensivamente usado pela IBM sendo substituído por redes neurais posteriormente, quando as redes apresentaram uma maior eficiência em reconhecer as palavras *“bee”, “dee”, “ee”* e *“vee”* (chamado de vocabulário

confuso BDEV). Essas palavras são difíceis de distinguir porque possuem curta duração, baixa intensidade vocal e, praticamente, quase a mesma frequência. Sem entrar em detalhes de aquisição e processamento do sinal, o sistema da IBM, funcionando com o modelo HMM padrão, possuía uma performance de 80 %**.** Realizando experimentos com redes neurais, os cientistas da IBM treinaram uma rede para reconhecer as mesmas 4 palavras e obtiveram uma performance de 90.9 % a mais próxima da performance humana de reconhecimento. A IBM na época, realizou também, testes com as mesmas 4 palavras com seres humanos, a performance humana foi de 94 % (Tafner, 1996).

Havendo tecnologia, tanto de hardware quanto de software, pouco bastava para que as idéias fossem colocadas em prática, era apenas uma questão de tempo. Logo, no final da década de 1980, artigos sobre reconhecedores de fala já começavam a aparecer nos congressos de computação (Tafner, 1996).

Em meados dos anos 1990, já há registro do primeiro sistema de reconhecimento de fala utilizando redes neurais (Kitano, 1994).

2.5 BENEFÍCIOS E APLICAÇÕES

O reconhecimento de fala pode ser utilizado de diversas formas, e em diferentes situações, como por exemplo as que seguem :

a) para a obtenção de números para efetuar a discagem de um telefone ou operação de um eletrodoméstico como forno de microondas, ar condicionado, etc;
b) auxiliar na condução de automóveis, onde o condutor não precisa distrair-se para efetuar uma configuração diferente nos controles do automóvel, como por exemplo : rádio, luzes, marchas, etc;
c) dispensar utilização de periféricos comumente utilizados como: mouse, teclado e joystick;
d) permitir uma interface o mais natural possível com o usuário em softwares.

Segundo Baeker (2000) o reconhecimento de fala é atualmente amplamente utilizado nas linhas de produção onde necessita-se da entrada de fala, ou comandos pronunciados enquanto que as mãos do operador estão ocupadas.

Outra utilização que deverá trazer grandes benefícios é o reconhecimento de fala auxiliando no controle de computadores pessoais, ou, interagindo com aplicações tanto locais como remotas, onde um simples telefonema, e a pronúncia de comandos acionará ações (Baeker, 2000).

2.5.1 APLICAÇÕES PRÁTICAS

Baeker (2000) menciona algumas aplicações do reconhecimento de fala, atualmente sendo utilizadas:

a) a Delco eletronics comprou licença de uso de sistemas de reconhecimento de fala da IBM PC/AT Cherry Eletronics e da Intel RMX86, que são utilizados para a coleta de placas de circuito integrado com defeito, através de comando de voz, enquanto que o operador marca as placas coletadas;
b) inspetores da Southern Pacific Railway, utilizam o sistema de reconhecimento de fala PC-based Votan para entrar com as informações dos vagões inspecionados através de walkie-talkies;
c) a empresa AT&T está avaliando um sistema de reconhecimento de fala para automatizar o processo de discagem nos escritórios da empresa nas regiões do Reno, Nevada e Hayward, estado da Califórnia, Estados Unidos.

2.6 ABORDAGENS

2.6.1 MÉTODO ANALÍTICO X MÉTODO GLOBAL

Existem duas abordagens para o tratamento da informação de entrada vindo da palavra falada, que são: o método global e o método analítico, segundo Hugo (1992).

No método global, cada som digitalizado é visto como uma unidade. Utilizam-se técnicas para comparar globalmente a palavra falada (unidade) de forma que ela seja reconhecida entre palavras previamente armazenadas pelo computador (Bruns, 1995). Segundo Hugo (1992) esta abordagem global se revela insuficiente se pretende-se tratar com grandes vocabulários ou fala contínua. Se faz, então, necessário adotar a abordagem analítica, que consiste em segmentar a mensagem em constituintes elementares (fonemas,

meia-sílabas, sílabas, etc). Após identificados estes últimos, deve-se reconstituir a frase pronunciada por etapas sucessivas: léxica, sintática, etc.

2.6.2 RECONHECIMENTO X COMPREENSÃO

Uma outra classificação foi proposta pelos pesquisadores do projeto ARPA, que introduziram o termo compreensão da fala (*speech understanding*) em oposição ao termo reconhecimento da fala (*speech recognition*) (Hugo, 1992).

Reconhecimento da fala consiste no reconhecimento de fonemas, sílabas, palavras para formar a mensagem original, como foi pronunciada (Hugo, 1992).

Compreensão da fala baseia-se no entendimento do senso, do significado da mensagem, visando fazer com que o sistema execute algo. Para tal são aceitos eventuais erros (Hugo, 1992).

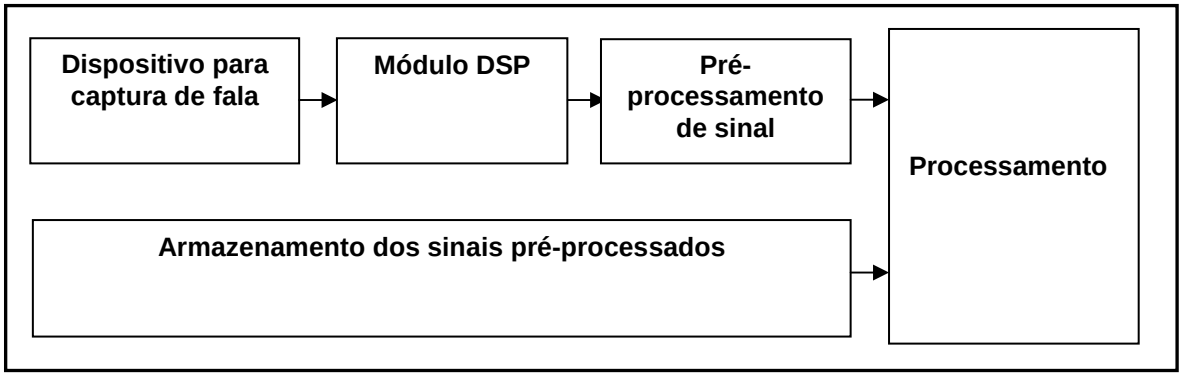
2.7 FASES

Segundo Baeker (2000), a maioria dos sistemas de reconhecimento de fala, possuem 5 (cinco) componentes:

a) um dispositivo para a captura da fala, que geralmente consiste em um microfone associado a um conversor analógico-digital, que codifica digitalmente a forma da onda da fala natural;
b) um módulo de processamento de sinal digital (DSP), onde sua função é de identificar o fim de cada palavra a ser reconhecida;
c) pré-processamento, que tem como objetivo ajustar os sinais que serão processados, fazendo com que o processamento seja executado de forma otimizada e com a melhor performance possível;
d) armazenamento, que possibilita guardar os resultados dos componentes anteriores, possibilitando uma futura reapresentação dos sinais lidos, sem ter que passar novamente pelos componentes 1, 2 e 3;
e) processamento, que consiste em uma estrutura que possibilite distinguir os padrões apresentados a ela.

A figura 2.2 demonstra as fases do reconhecimento de fala segundo Baeker(2000).

Figura 2.2 – Componentes de um sistema de reconhecimento de fala típico



Fonte: Baeker(2000)

Para Revox (1997), o reconhecimento automático da fala é o processo de extração automática da informação lingüística do sinal de voz, e está dividido em três etapas :

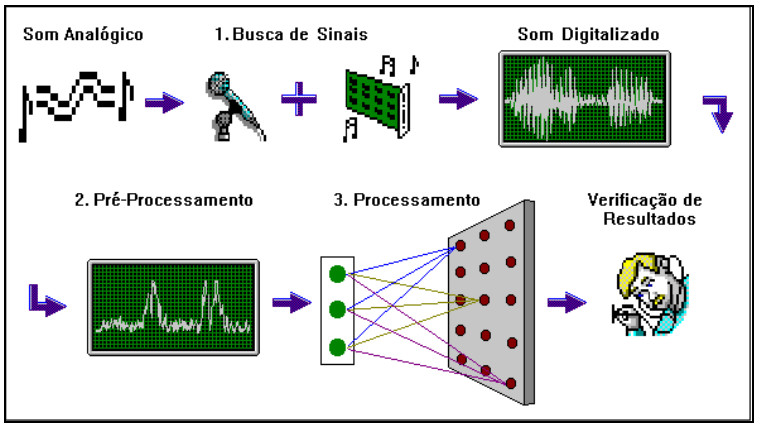
a) aquisição do sinal de voz: através de um dispositivo conversor analógico/digital, obtendo-se o sinal a ser reconhecido;
b) extração de parâmetros: adquirido o sinal, o mesmo será representado através de algum algoritmo de parametrização, por um conjunto de características que descrevem de maneira adequada as propriedades do sinal da voz;
c) reconhecimento do padrão: após a extração das características do padrão, esta fase responsabiliza-se pela identificação dos mesmos, isto é, verifica a que padrão de referência (conhecidos) o padrão de entrada (o qual se deseja reconhecer) se assemelha.

Bruns (1995) divide o reconhecimento de fala em fases, o qual devem ser realizadas em seqüência crescente de necessidade e dependência, sendo elas as fases de: busca de sinais sonoros, pré-processamento de sinais digitais, o processamento dos sinais obtidos e a verificação dos sinais obtidos.

Neste trabalho de pesquisa, serão utilizadas praticamente as mesmas fases utilizadas por Bruns (1995), pelo fato de conter em sua seqüência de etapas, as etapas citadas por Revox (1997) e Baeker (2000), adicionada da verificação dos sinais obtidos.

Como primeira etapa para este trabalho de pesquisa, será feita a coleta dos sinais analógicos, e através da interface de som, a conversão analógico-digital. Na seqüência será feito o pré-processamento dos sinais através de médias dos sinais, proposto por Tafner (1996) e o processamento realizado pela RNA *RBF-Fuzzy Artmap*. O resultado obtido é comparado ao de outros trabalhos que pesquisaram o reconhecimento de fala utilizando RNA, sem a interface *Automation OLE*. Estas etapas podem ser melhor visualizadas na figura 2.3.

Figura 2.3 – Etapas no reconhecimento de fala



Fonte: Hugo (1995)

2.8 TÉCNICAS

O uso de RNA no reconhecimento de fala tem proporcionado um desenvolvimento progressivo. Isto graças às vantagens das RNAs: adaptabilidade (uma rede pode se adaptar a situações diferentes), generalização (a rede é capaz de extrapolar os conhecimentos obtidos, pelo fato de conhecer situações semelhantes) e flexibilidade (introduzir novos conhecimentos sem a necessidade de reprogramar a rede) (Dalabrida, 1999).

Pode-se encontrar na literatura disponível vários modelos para o reconhecimento de padrões, das quais segundo Loesch (1996) pode-se citar:

a) *Adaptative Ressonance Theory* (ART): publicada em 1983, por G. Carpenter e S. Grossberg, tem como vantagens: capacidade de apreender novos padrões, novas categorias de padrões e de reter as categorias já apreendidas. Como desvantagem, a natureza dos exemplares categóricos podem mudar com o aprendizado;

b) *BackPropagation Perceptron*: publicada em 1974-1986 por P.J.Werbos, D. Parker, D. Rumelhart, tem como vantagens: operação rápida, eficiente na formação de representações internas das características dos dados de entrada ou classificação e outras tarefas. Como desvantagem, o seu tempo de treinamento é muito longo;

c) *Recurrent*: publicada em 1987 por Pineda Almeida, tem como vantagens: melhor emprego para classificação, mapeamento de informações variando com tempo. Como desvantagens: complexidade, difícil treinamento e otimização;

d) *Time Delay*: publicada em 1987 por D.W. Tank e J.J. Hopfield, tem como vantagens: desempenho equivalente aos melhores métodos convencionais, rápida operação. Como desvantagens: janela fixada para a atividade temporal representada, responde desastradamente para diferenças em escala na entrada;

e) *Boltzmann Machine*: publicada em 1984, 1986 por G. Hinton, T. Sejnowski, D. Ackley e H. Szu, tem como vantagens: capacidade de formar representação ótima das características dos padrões. Como desvantagens: tempo de aprendizado longo;

f) *Redes de Kohonen*: publicada em 1981 por T. Kohonen, tem como vantagem a capacidade de auto-organizar representações vetoriais de dados com uma ordenação significativa entre as representações, e como desvantagens o treinamento lento.

g) *RBF-Fuzzy Artmap*: publicada em 1995, por Gerson Tontini, como vantagens Tontini (1995) cita: as qualidades que a rede RBF na rede Fuzzy Artmap diminuindo a sensibilidade a ordem dos padrões de treinamento, ao mesmo tempo que mantém a capacidade de aprendizado instantâneo. Como desvantagem Tontini (1995) cita que a *RBF-Fuzzy Artmap* necessita de um número maior de nós, na camada RBF, do que a rede Fuzzy Artmap necessita na rede ARTa.

2.9 PROBLEMAS

A tarefa de reconhecimento da fala não é uma tarefa fácil para um sistema computacional. A construção de programas capazes de se comunicar através da fala requer a implementação dos mesmos recursos da comunicação escrita, acrescentado-lhes, ainda, meios de captar sons, separar ruídos e ambigüidades do sinal de áudio, especialmente as produzidas pela não separação de vocábulos (Tafner, 1996).

Uma mesma palavra ao ser pronunciada várias vezes pode apresentar diferentes formas de onda devido à articulação dos órgãos do aparelho fonador (Revox, 1997). Segundo Fischer (1999), o problema do reconhecimento de fala é adicionalmente complicado pela concatenação, onde consoantes adotam aspectos de consoantes vizinhas.

Problemas com as variações nas características da fala onde podem haver: diferenças acústicas não lineares no tempo (ritmo), em freqüência (timbre) e em amplitude (intensidade). A fala pode ainda conter insuficiência de toda a informação lingüística, como por exemplo, erros de português e sotaque (Revox, 1997).

Outro fator que deve ser levado em conta, segundo Fischer (1999) é a dependência de locutor. Um sistema dependente de locutor é treinado por um único locutor, podendo fornecer resultados de relativa boa acuracidade. Por outro lado, um sistema independente de locutor é treinado para tratar com uma variedade de locutores. Por este motivo, uma maior variabilidade é introduzida, dadas as características individuais e diferenciadas dos diversos locutores, como velocidade de fala, timbre de voz e altura, citados também por Revox (1997).

2.10 IA E RECONHECIMENTO DE FALA

As novas técnicas de inteligência artificial (IA) têm sido desenvolvidas através de um aspecto curioso, o retorno do homem à natureza. Técnicas como redes neurais que se baseiam exatamente na neurotransmissão ocorrida nos animais para lançar suas bases de fundamentação. Com base neste fato, pode-se fazer uma analogia entre células nervosas vivas e o processo eletrônico, onde enfatizou-se o aprendizado dos sistemas como forma de

captação de conhecimento. Esta técnica vem sendo utilizada para o reconhecimento de padrões como fala e visão (Xerez, 1996).

Existe uma grande tendência de se utilizar algoritmos “inteligentes”. Esses algoritmos se baseiam na técnica de aprendizagem e processamento humano, devido a sua grande capacidade de processamento de informações, aos quais destacam-se as redes neurais artificiais (RNA) (Dalabrida, 1999).

Uma característica interessante das redes neurais é a capacidade de generalização, que torna possível a produção de uma saída correta mesmo que a entrada esteja parcialmente incorreta ou incompleta. Isto torna a rede útil para o reconhecimento de padrões e distinção de sinais (Loesch, 1995).

De acordo com Xerez (1996), a RNA é utilizada, basicamente, como reconhecedor de padrões com capacidade de reconhecer dados que até então não lhe foram apresentados, ou, de outra forma, reconhecer dados que não constituíram o conjunto de treinamento. Levando-se em consideração essas características, uma das principais áreas as quais se fazem necessárias à aplicação de RNA, é sem dúvida o reconhecimento de sinais.

3. REDES NEURAIS ARTIFICIAIS

3.1 INTRODUÇÃO

Tontini (1995) menciona que há muitos anos os pesquisadores vêm tentando entender como o cérebro humano funciona na esperança de que esse conhecimento possa ser usado na implementação de máquinas inteligentes. As RNA são o primeiro passo consistente nesta direção.

Segundo Betemps (1997), as RNA são um modo de tentar simular o sistema nervoso biológico, buscando emular os recursos desses sistemas orgânicos. As RNA são capazes de estabelecer relações complexas entre dados, relativamente precisas, não necessitando informar a elas qual a função que os relaciona. Hugo (1992) menciona que desde a década de 50, pesquisadores tentam confeccionar RNA que funcionem similarmente ao cérebro humano.

As RNA são “máquinas” concebidas para trabalharem segundo o mesmo processo de um sistema nervoso. Seus elementos de processamento são neurônios artificiais, interconectados, que efetuam a soma das entradas e geram uma saída através da aplicação dessa soma a uma função de transferência (Campestrini, 2000).

3.2 CONCEITOS

RNA são sistemas computacionais, de implementação em hardware ou software, que imitam as habilidades computacionais do sistema nervoso biológico, usando um grande número de simples neurônios artificiais interconectados (Loesch, 1996). Segundo Betemps (1997), as RNA são um modo de tentar simular o sistema nervoso biológico, buscando emular os recursos desses sistemas orgânicos. As RNA são capazes de estabelecer relações complexas entre dados, relativamente precisas, não necessitando informar a elas qual a função que os relaciona.

As RNA são também conhecidas por outros termos, como: ciência da neurocomputação, processamento paralelo maciço, neuro-computadores, computadores

biológicos e sistemas neumórficos. Todos esses termos são sinônimos e tem um mesmo objetivo: procurar através de modelos baseados nos neurônios de um sistema biológico, captar a essência do mecanismo operatório dos neurônios, abstraindo suas propriedades operatórias mais importantes, bem como descobrir as propriedades emergentes do comportamento destes elementos (Loesch, 1996).

As RNA tendem a trabalhar com os dados de uma forma inteiramente diferente dos sistemas baseados em algoritmos ou conjuntos de regras. As redes neurais processam dados com base em informações, enquanto que a computação programada, faz uso de algoritmos e regras (Betemps, 1997).

3.3 CARACTERÍSTICAS

Loesch (1996), descreve como sendo as três principais características das RNA, contribuindo para a sua habilidade funcional:

a) estrutura: onde cada RNA possui estruturas diferentes que acarretam em aplicações mais adequadas a determinados assuntos;
b) dinâmica: específica como a rede computa os valores;
c) aprendizado: possibilita a RNA aprender com padrões novos a ela apresentados para em uma ocasião semelhante que possa vir a acontecer, resolver novos problemas.

Segundo Tontini (1995), as redes neurais possuem inúmeras características que as tornam atrativas a diversas aplicações. Como exemplo, pode-se citar o reconhecimento de padrões. Estas características são:

a) capacidade de aprendizado: as RNAs tem a habilidade de aprender com base em dados apresentados para treinamento ou em experiências passadas;
b) auto-organização: uma RNA pode criar sua própria organização ou representação das informações que recebe durante o treinamento e/ou operação;
c) degeneração suave: dados parciais, contaminados com ruído, ou uma destruição parcial da rede causa uma degeneração gradativa, diminuindo seu desempenho, mas ainda guardando alguma capacidade de associação ou operação;

d) operação em tempo real: como o processamento é feito em paralelo, com cada nó da rede (em uma mesma camada) processando ao mesmo tempo as informações, os cálculos e operações são feitas muito mais rapidamente que em sistemas comuns (desde que a rede esteja implementada em hardware);
e) essencialmente não linear: como as funções de transferência em cada nó são não lineares, a rede tem a capacidade de modelar sistemas não lineares.

3.4 APRENDIZADO

A habilidade de aprender é uma característica fundamental da inteligência. Embora uma definição precisa sobre o que seria aprendizado seja difícil de formular, o processo de aprendizado no contexto de RNA pode ser visto como o problema de atualizar a arquitetura da rede e os pesos das conexões tal que a rede possa realizar eficientemente uma determinada tarefa (Heckmann, 1998).

3.5 TIPOS

Heckmann (1998), descreve a existência de três principais paradigmas (tipos, formas) de aprendizado:

a) supervisionado: a rede recebe a resposta correta (saída) para cada padrão de entrada. Os pesos são determinados de tal forma a permitir que a RNA produza respostas tão próximas quanto possíveis das respostas corretas conhecidas;
b) não-supervisionado: não requer uma resposta correta associada a cada padrão de entrada contido no conjunto de dados de treinamento. Explora a estrutura não aparente entre os dados, ou correlações entre padrões entre dados, e organiza os padrões em categorias a partir desta correlações. No aprendizado não supervisionado, a rede é ajustada em termos de regularidades estatísticas dos dados de entrada tal que forme categorias ou classes;
c) híbrido: combina os aprendizados supervisionado e não-supervisionado.

Os tipos de aprendizado empregam regras, que de acordo com Heckmann (1998) são quatro: de correção de erro, de Boltzmann, de Hebb e o aprendizado competitivo.

3.6 VANTAGENS X DESVANTAGENS

Como principais vantagens das RNAs Betemps (1997) menciona:

a) menor necessidade de se determinar a prioridade: fatores determinantes sobre o modelo que está sendo desenvolvido;
b) interferência de múltiplos fatores de entrada: permitindo um inter-relacionamento muito mais complexo entre elas;
c) alta tolerância a falhas: uma vez que é permitida a entrada de grande número de parâmetros;
d) modelagem direta do problema: sem necessidade de se seguir um modelo preestabelecido;
e) paralelismo inerente: cada sinapse na rede neural pode ser seu próprio processador.

Betemps (1997) ainda menciona que características como: tolerância a falhas, robustez e capacidade de implementar uma classe particular de transformações são garantidas por teoremas matemáticos, ou seja, podem ser empregados de forma útil e confiável, servindo como vantagens na escolha de uma técnica na resolução de um problema. Porém, esses teoremas não afirmam sobre como, em termos conceitualmente mais altos, a RNA aprende o conhecimento. Essa falta de embasamento teórico das RNA ainda é uma das mais sérias desvantagens desta técnica (RNA), por gerar alguma desconfiança por parte dos especialistas quanto a sua confiabilidade.

Loesch (1996) cita as principais vantagens e desvantagens das RNA são:

a) aprendizagem por exemplos: é desejado quando a complexidade computacional do problema a ser resolvido é alta, no sentido que o problema não possui todas as suas variáveis conhecidas;
b) independência do problema: uma RNA antes da aprendizagem não possui nenhum conhecimento sobre o problema que se pretende resolver. Assim,

uma mesma RNA pode ser aplicada a problemas diferentes sem necessidade de qualquer modificação na sua estrutura;

c) obtenção de resultados desejados: caso uma RNA não esteja fornecendo resultados desejados aceitáveis, a sua arquitetura pode ser modificada em busca de otimizações;
d) processamento distribuído, paralelo e local: a própria arquitetura da rede e a natureza dos neurônios propiciam um processamento distribuído, paralelo e local. O paralelismo se dá nos níveis de módulos de redes neurais, camadas, neurônios e conexões;
e) implementação realística: diversos modelos de RNA têm sido realizados em hardware, a fim de otimizar as implementações e aumentar a sua faixa de utilização. Novas arquiteturas de sistemas de computação têm surgido sem utilizar uma arquitetura Von Neumann.

Como desvantagens oferecidas pelas RNA Loesch (1996) cita:

a) falta e a dificuldade de um formalismo na especificação e na análise de modelos de RNA, uma vez que, para se compreender os mecanismos fundamentais das redes, é necessário realizar simulações que, na maioria dos casos, são tarefas árduas e distantes da realidade do modelo;
b) o estado da arte das implementações ainda limita certas aplicações. Certos problemas exigem poderes computacionais específicos das implementações que ainda não foram atingidos, como de computadores mais rápidos ou RNA a serem aperfeiçoadas, entre outros.

3.7 TÉCNICAS

Ao escolher um modelo de RNA para uma determinada aplicação a ser desenvolvida, devem ser levados alguns fatores em consideração, como por exemplo, as características da RNA, e a finalidade a qual a RNA foi desenvolvida ou melhor se aplica (Loesch, 1996).

As RNA têm sido aplicadas na solução de vários problemas computacionais, e os principais resultados são encontrados em problemas como: reconhecimento de padrões

(caracteres, voz, face, manuscritos), controle/robótica, processamento de sinais, otimizações e associação de padrões (Loesch, 1996).

A tabela 3.1 apresenta inúmeras técnicas de RNA, mencionando também a aplicação básica.

Tabela 3.1 – Técnicas de RNA

RNA	Ano	Aplicação principal
Adaline / Madaline	1960	Filtro adaptativo de sinal, equalização adaptativa
Adaptative Ressonance Theory (ART)	1983	Reconhecimento de padrões
Backpropagation	1974/1986	Reconhecimento de padrões, filtro de sinais, remoção de ruído, classificação, modelamento de sistemas
Recurrent	1987	Controle de robôs, reconhecimento de fala
Time-Delay	1987	Reconhecimento de voz
Rede de ligações funcionais	1988	Classificação e mapeamento
Rede de funções de base radial	1987/1988	Classificação e mapeamento
Backpropagation de função utilidade no tempo	1974	Neurocontrol
BAM- Memória Associativa Bidirecional	1987	Memória heteroassociativa de conteúdo endereçável
Máquinas de Boltzmann e Cauchy	1984/1986	Reconhecimento de padrões, otimização
Boundary Contour System	1985	Processamento de imagens de baixo nível
Brain-State-in-a-Box (BSB)	1977	Recuperação autoassociativa
Hopfield	1982	Recuperação autoassociativa, otimização
Quantização do Vetor de Aprendizagem	1981	Recuperação autoassociativa, compreensão de dados
Neocognitron	1975/1982	Reconhecimento de escrita manual
Mapas de preservação da topologia de auto-organização	1981	Mapeamento complexo, compreensão de dados, otimização

Fonte: Tontini (1995).

4. RNA RBF-FUZZY ARTMAP

4.1 INTRODUÇÃO

As aplicações da lógica difusa têm cada vez mais aplicações em problemas reais, sendo ela mais adequada que a lógica clássica na representação de problemas reais, onde informações são mal definidas e imprecisas (Tontini, 1995).

Como visto anteriormente, no capítulo 4, as RNA apresentam uma série de vantagens para aplicação na identificação de padrões, mais especificamente neste trabalho, no reconhecimento de fala. Tontini (1995), menciona que a introdução de lógica difusa dentro das RNAs permite que elas tratem problemas difusos de maneira mais natural.

No decorrer deste capítulo será demonstrado conceitos referentes à estas RNAs, e o funcionamento da RNA *RBF-Fuzzy Artmap*.

4.2 LÓGICA DIFUSA

Existe na comunicação quotidiana muitas palavras e sentenças com significado não preciso ou vago (Rabuske, 1995 apud Heckmann 1998). Isto acontece porque, tanto quem fala como quem ouve, não necessita de informações mais precisas e está acostumado a lidar com tais tipos de imprecisão (Heckmann, 1995). Pode-se citar como exemplo: uma determinada pessoa em uma lanchonete pede para beber uma xícara de café, esta pessoa não está preocupada com a temperatura real do café, mas somente que o café esteja acima do que ela considera morno. Analisando verifica-se que ninguém é capaz de determinar o ponto preciso em que o café passa de morno para quente.

Interessado em representar tais imprecisões, Zadeh desenvolveu a teoria dos conjuntos difusos, também chamados conjuntos nebulosos (Marks, 1994 apud Heckmann, 1998). De acordo com essa teoria, um conjunto não apresenta limites bem definidos, podendo um elemento pertencer parcialmente a ele, ou pertencer a dois conjuntos ao mesmo tempo. O que vai caracterizá-lo será o grau de pertinência, que é uma medida que quantifica o grau de quanto este elemento pertence a um determinado conjunto.

Conforme exemplo anterior, e recorrendo a forma de pensar de Zadeh, o café não tem apenas dois estados, ou é quente ou não é, mas tem um grau de pertinência a quente. A determinação do grau de pertinência é, por vezes, bastante subjetiva.

O conceito de Lógica Difusa (LD) foi concebido por Lotfi Zadeh (1973), como uma forma de processar dados por permitir pertinência parcial a um conjunto ao invés de pertinência binária (Ross, 1995 apud Heckmann, 1998). A LD provê um modo simples de chegar a uma conclusão, findada numa informação de entrada que seja vaga, ambígua, imprecisa ou ruidosa, isto é, uma conclusão imprecisa é deduzida a partir de uma coleção de premissas imprecisas. A LD é uma extensão da lógica modal, com processos de inferência e operadores derivados da Teoria dos Conjuntos Difusos (Heckmann, 1998).

Na modelagem de sistemas são muito comuns as técnicas de análise quantitativa, baseadas na modelagem matemática que, por meio do uso de símbolos, equações e outras sentenças matemáticas, representam uma certa realidade. No entanto, existem fenômenos complexos ou mal definidos, onde as técnicas de análise convencionais, baseadas na manipulação precisa e rigorosa dos dados, não são adequados (Ross, 1995 apud Heckmann, 1998). Pheckmann (1998) ainda menciona que para ser capaz de fazer inferências significativas a respeito do comportamento de tais sistemas, é necessário abandonar o alto rigor e precisão de nossa análise matemática e ser mais tolerantes, pois tal comportamento é aproximado por natureza.

4.3 REDE DE FUNÇÕES COM BASE RADIAL (RBF)

Segundo (Haykin, 1994 apud Heckmann, 1998), Broomhead e Lowe em 1988 foram os primeiros a explorar o uso da rede de Funções com Base Radial (RBF) no projeto de RNA.

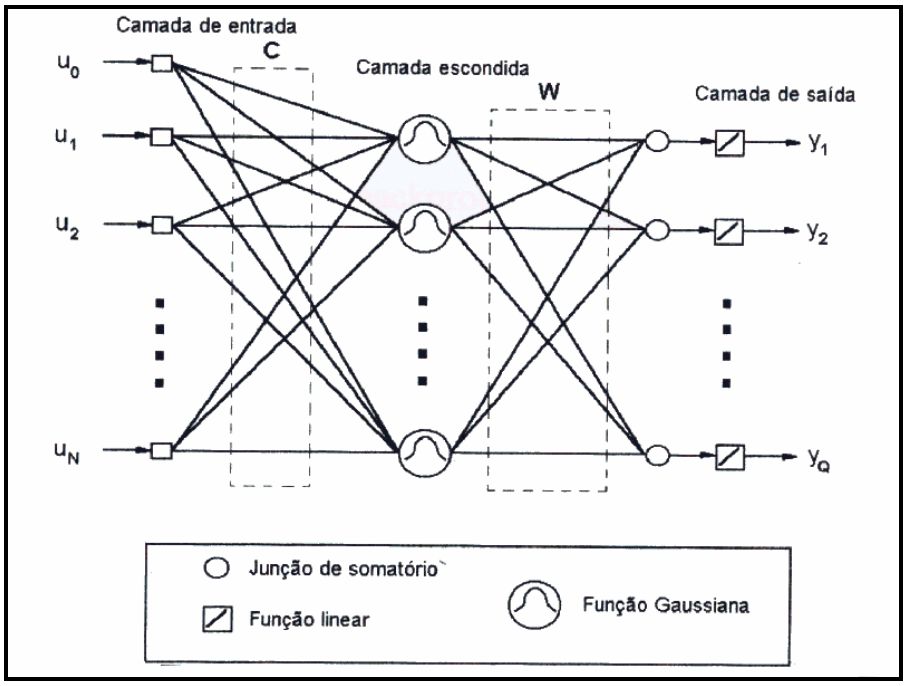
A RNA RBF é funcionalmente equivalente a um sistema com lógica difusa, segundo (Jang, 1993 apud Tontini, 1995). A RBF é uma alternativa para a dificuldade de determinação do número de camadas intermediárias e do número de neurônios nessas camadas (encontradas nas RNA como a de backpropagation) e o tempo de treinamento da rede é geralmente menor (Tontini, 1995).

A RNA RBF deste trabalho terá sua camada intermediária treinada pelo método de aproximação sucessiva e a camada da saída pelo método *gradient descent* conforme proposta por Tontini (1995).

4.3.1 FUNCIONALIDADE

A RBF possui sua meso-estrutura dividida em três camadas: camada de entrada, camada intermediária (também conhecida como camada escondida, proveniente do termo “hidden layer”) e camada de saída como demonstrado na figura 4.1.

Figura 4.1 – Estrutura básica da RBF



Fonte: Heckmann (1998)

A camada de entrada distribui os padrões de entrada para os diversos nós da camada intermediária. A saída da camada intermediária é calculada pela função gaussiana contida no quadro 4.1.

Quadro 4.1 – Função para cálculo das saídas da camada intermediária RBF

$$\phi_h(x_i) = e\left(\frac{-\sum(x_i - c_i)^2}{\delta_h^2}\right)$$

Para h = 1, 2, ..., H; e i = 1, 2, ... N,

onde, $\underline{x_i}$ é o valor do padrão de entrada na dimensão “i”,

c é o centro da função gaussiana na dimensão “i”,

δ_h é um parâmetro que define a “largura” da função gaussiana para o nó “h”,

H é o número de nós na camada intermediária,

N é a dimensão do padrão de entrada.

Os valores na camada de saída são calculados pela equação contida no quadro 4.2.

Quadro 4.2 – Função para cálculo da camada de saída da RBF

$$T_j = \sum_{h=1}^{H} \phi_{hj} \cdot \phi_h$$

onde ϕ_{hj} são os pesos da camada de saída, para j = 1, 2, ..., J e “J’ = número de saídas.

4.3.2 TREINAMENTO

Segundo Tontini (1995), o treinamento da rede RBF é realizado em duas etapas:

a) inicialmente a camada intermediária é treinada por um método como o “k-means clustering” ou por outro algoritmo como o de aproximação sucessiva;
b) após o treinamento da camada intermediária, a camada de saída é treinada por um método linear, como mínimos quadrados, ou não linear, como *gradient descent*.

Tontini (1995) menciona que o desempenho da RNA RBF está diretamente relacionado com a eficiência no treinamento de sua camada intermediária, nominalmente, com a localização dos centros e a largura das funções gaussianas.

Segundo Heckmann (1998), a mais básica forma de treinamento para uma RNA RBF, emprega uma estratégia de dois passos, também utilizada por Tontini (1995), onde Heckmann denominou-a de aprendizado híbrido. Esta estima posições de centro e seu tamanho usando um algoritmo de agrupamento não supervisionado, seguido por um algoritmo de mínimos quadrados supervisionado para determinar os pesos das conexões entre a camada escondida e a de saída. Pelo fato das unidades de saída serem lineares, um algoritmo não iterativo pode ser utilizado. Depois desta solução inicial ser obtida, um algoritmo de gradiente descendente pode ser utilizado para refinar os parâmetros da rede, como também citado por Tontini (1995).

Heckmann (1998) ainda menciona que este algoritmo de aprendizado híbrido para treinamento da RBF converge muito mais rápido do que o algoritmo de backpropagation para o treinamento de perceptrons multicamadas. Entretanto, para muitos problemas, a rede RBF frequentemente envolve um número grande de unidades escondidas.

4.3.2.1 APROXIMAÇÃO SUCESSIVA

No método de aproximação sucessiva, apresentado por Linkens & Nie, os padrões são apresentados para treinamento apenas uma vez, com a camada intermediária se atualizando totalmente a cada apresentação. Este método aumenta a velocidade de treinamento da rede, pois a necessidade de interação será apenas para o treinamento da camada de saída (se treinada por “gradient descent”), ou pode ser realizado em apenas duas apresentações dos padrões de treinamento (se treinada por mínimos quadrados) (Tontini, 1995).

Neste método, quando um padrão é apresentado para treinamento, são realizados os seguintes procedimentos:

a) é calculado o valor de saída para todos os nós ϕ_h da camada intermediária;

b) é encontrado o valor ϕ_h que tenha uma resposta máxima ($\phi_{hmax} = \max(\phi_h)$, para h = 1, ..., H);

c) o nó vencedor é comparado a um valor de corte $0 < \phi_h < 1$. Se $\phi_{hmax} >= \phi_{h,}$ ϕ_{hmax} é o nó vencedor; ou, se $\phi_{hmax} < \phi_{h,}$ abre um novo nó;

d) se o nó ϕ_{hmax} é vencedor, modifica os parâmetros do nó segundo as equações contidas no quadro 4.3;

Quadro 4.3 – Equações para modificar parâmetros dos nós

$$N^{hmax} = n^{hmax} + 1;$$

$$\alpha^{hmax} = 1 / n^{hmax};$$

$$c_i^{hmax} = c_i^{hmax} + \alpha^{hmax} . [x_i - c_i^{hmax}];$$

Onde "n" é o número de vezes que o nó hmax foi treinado;

e) se um novo nó é criado, então H = H + 1, nH+1 = 1, ci = xi, para i = 1, ..., N.

O método de aproximação sucessiva necessita que a largura dos nós(ϕ), seja definida antes do início do treinamento. Esta definição não é muito crítica, pois o treinamento compensará diferenças na largura com um diferente número de nós e pelo treinamento da camada de saída (Tontini, 1995).

A adoção do método de aproximação sucessiva permite acelerar mais ainda o aprendizado da RNA RBF, pois o treinamento da camada intermediária é efetuado com apenas uma passada (Tontini, 1995).

4.3.3 VANTAGENS

Tontini (1995) cita como vantagens da utilização da RNA RBF:

a) a velocidade de treinamento é em geral uma ordem de grandeza mais rápida (Moody, 1989 apud Tontini, 1995);
b) possui três camadas, não tendo o problema da rede de *backpropagation* para a determinação do número de camadas intermediárias;
c) menos sensível a determinação de parâmetros da rede;
d) é funcionalmente equivalente a um sistema com lógica difusa, podendo ser aplicada com mais naturalidade aos problemas reais.

4.3.4 DESVANTAGENS

Tontini (1995) cita como desvantagens da RNA RBF:

a) sua aplicação em casos reais não está ainda bem estudada, principalmente no que diz respeito a problemas da qualidade;
b) apesar da rapidez de aprendizado, mesmo com o método de aproximação sucessiva, a rede de RBF necessita de um aprendizado off-line. Se um novo problema ou tipo de padrão tiver que ser aprendido pela rede após seu treinamento, é necessário repassar todos os exemplos de treinamento junto com os novos padrões para evitar com segurança que os padrões anteriores não sejam esquecidos.

4.4 FUZZY ARTMAP

A RNA Fuzzy Artmap surgiu da adaptação de uma RNA Artmap para a utilização de padrões analógicos tanto na entrada como na saída (Tontini, 1995). Esta RNA Artmap, por sua vez, originou-se do desenvolvimento da RNA Fuzzy-Art. Estes fatos serão melhor discutidos a seguir.

4.4.1 ART

A Teoria da Adaptação Ressonante (ART - *Adaptive Ressonance Theory*), foi inicialmente proposta por Stephen Grossberg, em 1976, e em publicações posteriores em parceria com Gail Carpenter (1986/87). É apropriada para reconhecimento de padrões (radar/sonar, reconhecimento de fala), e indicada para classificação de padrões (Loesch, 1996).

Tontini (1995) menciona que a ART, foi introduzida como uma teoria sobre o processamento cognitivo de informações no cérebro humano, e que, essa teoria levou ao desenvolvimento de uma série de modelos de RNA capazes de um aprendizado não supervisionado para classificação de padrões em tempo real.

De acordo com Loesch (1996), ART é um sistema que auto-organiza padrões de entrada em categorias de reconhecimento, mantendo um equilíbrio entre as propriedades de plasticidade (discriminação) e de estabilidade (generalização).

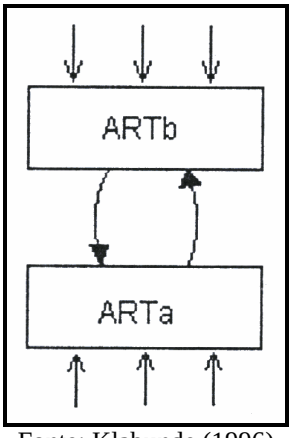
Conforme Loesch (1996) e Tontini (1995) citam, foram desenvolvidos três modelos de redes ARTs:

a) ART1: é capaz de aprender a categorizar padrões de entrada binários apresentados em ordem arbitrária;
b) ART2: pode aprender a categorizar padrões de entrada analógicos ou binários;
c) ART3: pode realizar uma busca paralela, ou teste de hipóteses, em códigos com reconhecimento distribuído.

Loesch (1996) afirma que os modelos ART são os mais adaptáveis para aprender e responder em tempo real, para um mundo não estacionário, com um número limitado de entradas, até que ele utilize toda a sua capacidade de memória.

Segundo (Vermuri, 1992 apud Klabunde, 1996), é aconselhável a utilização da RNA ART para o reconhecimento de padrões de entrada, devido a sua capacidade de implementação em forma de par (ARTa e ARTb) como pode ser visto na figura 4.2. Isto possibilita a RNA de atualizar e auto-estabilizar os pesos da rede e assim facilitar na compreensão de novos dados de saída.

Figura 4.2 – RNA ART



Fonte: Klabunde (1996)

4.4.2 ARTMAP

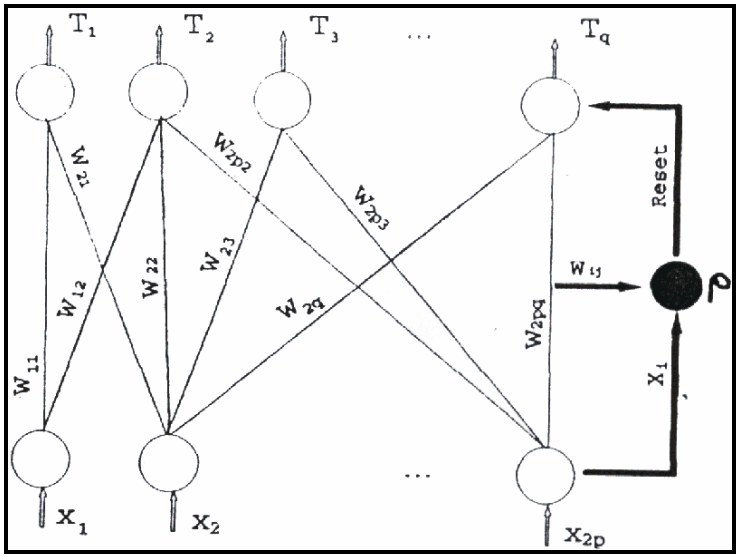
Tontini (1995) menciona que a ART-1, foi usada no projeto de uma rede com arquitetura hierárquica chamada Artmap, que possui aprendizado instantâneo, estabelecendo uma associação entre padrões binários de entrada com padrões binários de saída.

4.4.3 FUZZY-ART

A RNA Fuzzy-ART exemplificada na figura 4.3, é formada por duas camadas:

a) camada de entrada (x1, x2, ..., x2p): com número de nós igual ao dobro da dimensão dos padrões de entrada;
b) camada de saída (T1, T2, ..., Tq): formada de tantos nós “j”, quantos necessários para classificar os padrões de entrada.

Figura 4.3 – RNA Fuzzy-Art



Fonte: Tontini (1995)

A camada de saída inicia com um número “q” de nós não comprometidos (não treinados) (Tontini, 1995).

A ligação entre as camadas de entrada e saída é realizada por um conjunto de pesos W_{ij} que tem o valor 1 quando o nó “j” não está comprometido. Então assim, inicialmente, tempo para $W_{ij} = 1$ para i = 1,2, ..., 2p e j = 1 , 2, ..., q.

4.4.3.1 FUNCIONAMENTO

Segundo Fischer (1999), quando um padrão de entrada ‘X” é apresentado à camada de entrada, os valores dos nós da camada de saída são calculados pela equação contida no quadro 4.4.

Quadro 4.4 – Equação da camada de saída da Fuzzy-ART

$$T_j = \frac{\sum_{i-1}^{2p} \min(x_i W_{ij})}{\alpha + \sum_{i-1}^{2p} W_{ij}}$$

Onde:

a) x_i, i = 1, 2, ..., 2p são as dimensões dos padrões de entrada normalizados por uma técnica chamada codificação complementar, com O <= x_i <= 1;
b) w_{ij} são os pesos conectando o no “i” da camada de entrada com o nó “1” da camada de saída;
c) α é um valor pequeno > O;
d) T_j é o valor dos nós “j” na camada de saída, com j = 1, 2, ..., N;
e) “N” é o número de nós comprometidos na camada de saída.

A rede escolhe o nó “Tj” com o maior valor de saída e aplica o critério de comparação através da equação demonstrada no quadro 4.5.

Quadro 4.5 – Equação de comparação do nó de saída da Fuzzy-ART

$$A = \frac{\sum_{i-1}^{2p} mn(x_i, W_{ij})}{\alpha + \sum_{i-1}^{2p} x_i} \geq \rho$$

Onde “p” é o nível de vigilância da rede, fixado pelo usuário (O <= p <= 1).

Tontini (1995) menciona que se o critério de comparação não for válido (A < p), a rede desativa o nó vencedor “j” e escolhe outro nó. O processo se repete até que um nó satisfaça o critério de comparação. Se nenhum nó comprometido satisfizer o critério de comparação, a rede escolherá um nó não comprometido e abrirá uma outra classe na saída, fazendo n = n+1.

O nível de vigilância (p) determina quão discriminatória é a rede. Pequenos valores de “p” permitirão que um grande número de padrões sejam assimilados pelo mesmo nó de saída. Um valor de “p” grande tornará a rede mais discriminatória, aumentando também o número de nós (Klabunde, 1996).

Após um nó “j” adequado tenha sido escolhido, os pesos Wij são alterados de acordo com a equação descrita no quadro 4.6.

Quadro 4.6 – Equação de atualização dos pesos da Fuzzy-ART

$$W_{ij} = \beta . \text{Min}(x_i, W_{ij}) + (1 - \beta) W_{ij}$$

4.4.4 FUZZY-ARTMAP

A RNA Fuzzy-Artmap é uma generalização da rede binária Artmap (Tontini, 1995). Ela é capaz de um aprendizado supervisionado incremental, atualizando-se durante a operação sem “esquecer” o que já aprendeu anteriormente. A RNA Fuzzy-Artmap pode ser empregada para classificação e/ou associação de padrões binários de entrada e saída com dimensão arbitrária (Tontini, 1995).

Devido ao modelo de RNA ART1 ser utilizado no desenvolvimento do modelo de RNA ARTMAP e no desenvolvimento da rede Fuzzy-ART, permitiu a adaptação da rede ARTMAP para a utilização de padrões analógicos tanto na entrada como na saída. Esta nova versão é chamada Fuzzy-Artmap (Klabunde, 1996).

4.4.4.1 ESTRUTURA

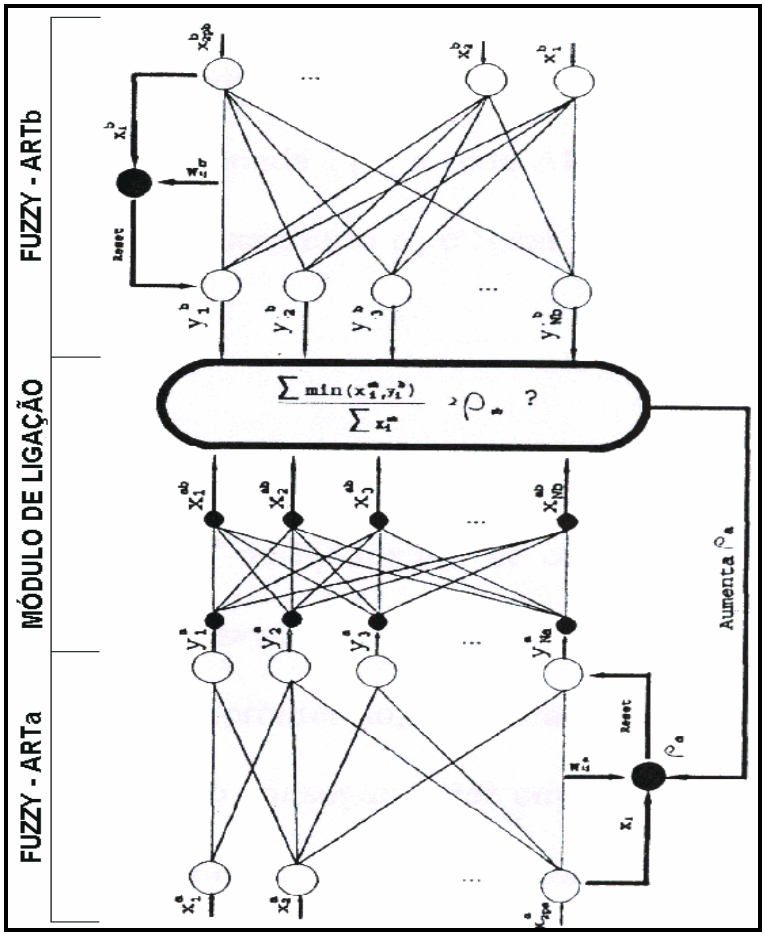
A RNA Fuzzy-Artmap incorpora dois módulos Fuzzy-ART: ARTa e ARTb (já mencionados no item 4.4.1), conectados através de um módulo inter-ART, chamado de módulo de ligação como pode ser visto na figura 4.4.

Os módulos ARTa e ARTb funcionam da mesma maneira que a rede Fuzzy-ART, exceto pela interação com o módulo de ligação. O módulo de ligação é usado para fazer associações preditivas entre categorias nas redes ARTa e ARTb e para executar a pesquisa de similaridade, onde o nível de vigilância do módulo ARTa aumenta em resposta a um erro de previsão no módulo ARTb. A pesquisa de similaridade reconhece a estrutura de uma categoria de tal maneira que o erro de previsão não será repetido em apresentações subsequentes da mesma entrada (Tontini, 1995).

O funcionamento da RNA Fuzzy-Artmap (item 4.3.5.2) é baseado nas definições a seguir, e são relatados neste item, por fazerem parte da estrutura da RNA:

a) As entradas das RNA ARTa e ARTb são normalizadas pelo método de classificação complementar;
b) $x^a = (x_1^a, ..., x^a_{2Pa})$ representa o vetor das entradas na camada F_1^a;
c) $y^a = (y_1^a, ..., y_{Na}^a)$ representa o vetor das saídas da camada F_2^a;
d) $w_j^a = (w_{j1}^a, w_{j2}^a, ..., w^a_{j2Pa)}$ j-ésimo vetor de pesos da rede ARTa;
e) $x^b = (x_1^b, ..., x^b_{2Pb})$ representa o vetor das entradas na camada F_1^b;
f) $y^b = (y_1^b, ..., y_{Nb}^b)$ representa o vetor das entradas na camada F_2^b;
g) $w_k^b = (w_{k1}^b, w_{k2}^b, ..., w^b_{k2Pb}{}^b)$ representa o k-ésimo vetor de pesos da rede ARTb;
h) $x^{ab} = (x_1^{ab}, ..., y_{Nb}^b)$ representa as saídas do módulo de ligação para rede ARTb;
i) $w_j^{ab} = (w_{j1}^{ab}, w_{j2}^{ab}, ..., w^{ab}_{j2Nb})$ representa o vetor de pesos ligando o j-ésimo nó em F_2^a à camada F_2^b;
j) x^a, y^a, x^b, y^b, x^{ab}, são igualados a zero no intervalo entre duas apresentações.

Figura 4.4 – RNA Fuzzy-Artmap



Fonte: Tontini (1995)

4.4.4.2 FUNCIONAMENTO

As camadas de saída da ARTa e ARTb (F_2^a e F_2^b) estão ativadas quando pelo menos um de seus nós está ativo devido à apresentação de uma entrada. O módulo de ligação F^{ab} é ativado (isto é, tem saída $x^{ab} \neq 0$) quando ao menos uma das camadas de ARTa e ARTb (F_2^a e F_2^b) estiver ativa. A saída x^{ab} do módulo de ligação é dada pela seguinte regra:

a) $x_i^{ab} = \min(y_i^b, w_{ji}^{ab})$ para i = 1, ..., Nb, se o J-ésimo nó da camada F_2^a estiver ativo e F_2^b também estiver ativa;

b) $x_i^{ab} = w_{ji}^{ab}$, para i = 1, ..., Nb, se o J-ésimo nó da camada F_2^a estiver ativo e F_2^b estiver inativa;

c) $x_i^{ab} = y_i^b$, para i = 1, ..., Nb, se a camada F_2^a está inativa e F_2^b está ativa;

d) $x_i^{ab} = 0$, se tanto a camada F_2^a quanto F_2^b estiverem inativas.

Tontini (1995) cita que, como nas camadas F_2^a e F_2^b só haverá um nó ativo de cada vez, x^{ab} será igual a 0 se a previsão executada por w_j^{ab} não for confirmada por y^b.

Quando um padrão é apresentado a camada F_1^a , a RNA ARTa seleciona um nó vencedor "J" que melhor represente aquele padrão. A esta altura p_a é igual à um valor base p_a.

4.4.4.2.1 ESTÁGIOS DO FUNCIONAMENTO

Klabunde (1996) menciona que a RNA Fuzzy-Artmap funciona em dois estágios, um de execução e outro de treinamento:

4.4.4.2.1.1 EXECUÇÃO

Caso a rede esteja em execução, a camada F_2^b estará inativa e a saída do módulo de ligação será dado por $x^{ab} = w_j^{ab}$. Se o nó "J" de F_2^a for um nó comprometido, w_j^{ab} terá apenas um valor $w_{jK}^{ab} \neq 0$, e a previsão da RNA Fuzzy-Artmap a matriz $w_k = 1$. Se o nó "j" de F_2^a for um nó não comprometido, w_j^{ab} terá todos os valores $w_{jK}^{ab} = 1$, significando que a rede não consegue fazer uma previsão com os casos apresentados para treinamento. Neste caso, a rede pode ser treinada imediatamente através da apresentação do padrão de saída correto na rede ARTb, ativando a camada F_2^b.

4.4.4.2.1.2 TREINAMENTO

Caso a RNA esteja sendo treinada, e a camada F_2^b esteja ativa, a saída do módulo de ligação será dado por $x_i^{ab} = \min(y_i^b, W_{ji}^{ab})$ para i = 1, ..., Nb.

O módulo de ligação faz a pesquisa de similaridade, comparando-se através da função contida no quadro 4.7.

Quadro 4.7 – Equação de comparação do nó de saída da RNA Fuzzy-Artmap.

$$A_{ab} = \frac{\sum_{i=1}^{Nb} \min(x_i^{ab}, y_i^b)}{\sum_{i=1}^{p} x_i^{ab}} \geq \rho_{ab}$$

Onde p_{ab} é o nível de vigilância do módulo de ligação.

Se $A_{ab} < p_{ab}$, o nível de vigilância da rede ARTa (P_a) é aumentado para um valor um pouco maior que a equação demostrada no quadro 4.8.

Quadro 4.8 – Equação do nível de vigilância da rede Fuzzy-Artmap.

$$\rho_a > \frac{\sum_{i=1}^{2p} \min(x_i^a , w_{iJ}^a)}{\sum_{i=1}^{2p} x_i^a}$$

Onde "J" é o índice do nó ativo em F_2^a.

Quando isto ocorre, a rede ARTa desativará o no "J" que tinha escolhido anteriormente, procurando por outro nó que satisfaça o seu critério de similaridade. Este novo nó é novamente testado pelo módulo de ligação. Este processo se repete até que um nó que tenha uma previsão correta seja escolhido ou, se este nó não existir, à RNA abrirá um novo nó que aprenderá a fazer a previsão correta. Este aprendizado é efetuado através do treinamento do módulo de ligação (Tontini, 1995).

Inicialmente os pesos do módulo de ligação $W_{jk}^{ab} = 1$. Durante o tempo em que o novo nó da rede ARTa está ativo, o vetor w_j^{ab} é igualado ao vetor x^{ab} . Uma vez que o nó "J" aprende a predizer a categoria "k" da rede ARTb, esta associação é permanente, isto é, $W_{jk}^{ab} = 1$ para sempre (Tontini, 1995).

O valor de vigilância p_a é retomado para $p_a = p_a$ no intervalo entre a apresentação de dois padrões consecutivos (Tontini, 1995).

Se $A_{ab} \geq P_{ab}$, os pesos da rede ARTa são atualizados pela equação demonstrada no quadro 4.9.

Quadro 4.9 – Equação de atualização dos pesos da RNA ARTa

$$\mathbf{W_{ij} = \beta \min (x_i, w_{ij}) + (1-\beta) w_{ij}}$$

Onde $\beta = 1$ para o caso de aprendizado rápido.

4.4.5 VANTAGENS

Tontini (1995), apresenta como vantagens da RNA Fuzzy-Artmap:

a) pode aprender instantaneamente durante sua operação *on-llne;*

b) tem capacidade de se especializar, determinando as dimensões das entradas que mais representam um tipo de padrão.

4.4.6 DESVANTAGENS

Tontini (1995), apresenta como desvantagens da RNA Fuzzy-Artmap:

a) apresenta problema de sensibilidade à ordem de apresentação dos padrões de treinamento e a outliers com valores pequenos;

b) ao contrário da RNA RBF (item 4.3), a RNA Fuzzy-Artmap possui uma função de pertinência gradual apenas para valores menores que o valor do nó. Valores maiores são tratados pela abertura de outros nós. Isto faz com que o processo de inferência não siga uma transição gradual entre as classes que se quer identificar.

4.5 RBF-FUZZY ARTMAP

Para suprir as necessidades e aproveitar as vantagens dos diferentes modelos de RNAs, Vermuri (1992, apud Klabunde) considera a junção de modelos diferentes de RNA como de grande valia para o aperfeiçoamento das RNAs e desenvolvimento das pesquisas.

Tontini (1995) apresentou a elaboração de uma nova proposta de RNA, conhecida por *RBF-Fuzzy Artmap*, que é composta pelas RNAs: RBF (item 4.3) e Fuzzy Artmap (item 4.4), portanto, uma RNA de meso-estrutura híbrida.

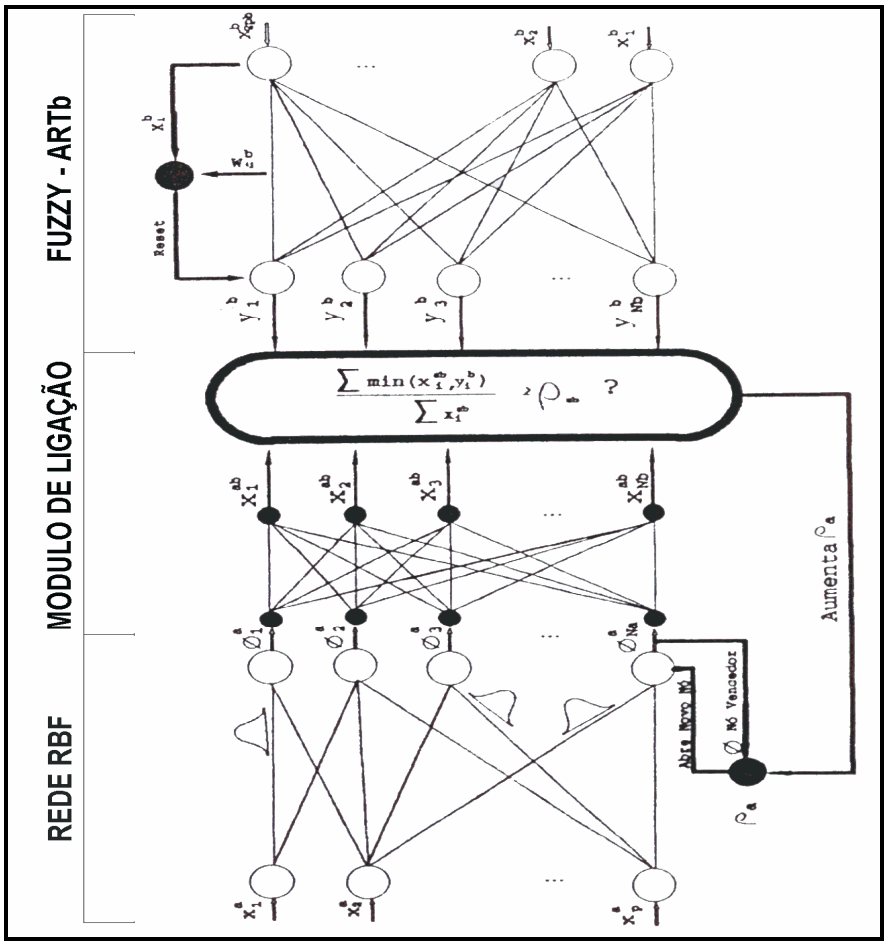
A inclusão de funções de base radial(RBF) na rede ARTa da RNA Fuzzy-Artmap, utilizando o método de treinamento por aproximação sucessiva, elimina o problema de *outliers* e da sensibilidade à ordem de apresentação dos padrões de treinamento, além de aumentar a capacidade de generalização da rede (Tontini, 1995).

Tontini (1995) menciona que a RNA RBF tem a vantagem de não ser muito sensível a ordem de apresentação dos padrões de treinamento. Porém, ela necessita de uma fase de aprendizado separada da fase de operação (como também já visto no item 4.3). Na aplicação desta RNA no reconhecimento de fala há a necessidade de um aprimoramento contínuo do sistema, para identificação de novos tipos de padrões.

A capacidade de aprendizado instantâneo e incremental, apresentada pela rede Fuzzy-Artmap (item 4.4), a torna uma forte candidata à aplicação em sistemas de reconhecimento de padrões (Exemplo : reconhecimento de fala). Porém, sua sensibilidade à ordem de apresentação dos padrões de treinamento é uma limitação séria, que deve ser superada para torná-la útil em aplicações reais (Tontini, 1995).

A RNA proposta por Tontini (1995), RBF Fuzzy Artmap, elimina as limitações apresentadas pelas RNA RBF e Fuzzy Artmap. Ela utiliza as funções de base radial e o mecanismo de treinamento por aproximação succesiva, da camada intermediária da rede RBF, em substituição à rede ARTa da rede Fuzzy-Artmap. A figura 4.5 mostra a arquitetura da nova rede.

Figura 4.5 – RNA RBF-Fuzzy Artmap



Fonte: Tontini (1995).

4.5.1 FUNCIONAMENTO

Tontini (1995) explica o funcionamento da RNA *RBF-Fuzzy Artmap* através do funcionamento das RNA que a compõem, já descritos neste trabalho de pesquisa nos items 4.3 e 4.4.

Quando um padrão é apresentado a RNA *RBF-Fuzzy Artmap*:

a) as saídas ϕ_h (h = 1,2, ..., Na) são calculadas da mesma maneira que a rede RBF;
b) acha o valor ϕ_h que tenha resposta máxima;
c) compara a saída do nó vencedor ao nível de vigilância p_a, se $\phi_{hmax} \geq p_a$, ϕ_{hmax} é o nó vencedor, ou ainda, se $\phi_{hmax} < p_a$, abre um outro nó com o centro nas coordenadas do padrão de entrada.

Em seguida a rede calcula a saída do módulo de ligação x^{ab} pela regra:

a) $x_i^{ab} = \min(y_i^b, w_{hmaxi}^{ab})$ para i = 1, ..., Nb, se as camadas RBF e F_2^b estiverem ativas;
b) $x_i^{ab} = w_{hmax}^{ab}$, para i = 1, ..., Nb, se a camada RBF estiver ativa e F_2^b estiver inativa;
c) $x_i^{ab} = y_i^b$, para i = 1, ..., Nb, se a camada RBF estiver inativa e F_2^b estiver inativa;
d) $x_i^{ab} = 0$, se tanto a camada RBF quanto F_2^b estiverem inativas.

Como nas camadas RBF e F_2^b só haverá um nó ativo de cada vez, x^{ab} será igual a 0 se a previsão executada por w_{hmax}^{ab} não for confirmada por y^b.

4.5.1.1 ESTÁGIOS

4.5.1.1.1 EXECUÇÃO

Caso a RNA esteja em execução, a camada F_2^b estará inativa e a saída do módulo de ligação será dado por $x^{ab} = w_{hmax}^{ab}$. Se o nó ”hmax” não for um nó novo, w_{hmax}^{ab} terá apenas um valor $w_{hmaxK}^{ab} \neq 0$, e a previsão da RNA Fuzzy-Artmap será igual a matriz $w_k^b = (w_{k1}^b, w_{k2}^b, ..., w_{k2P}^b)$. Se o nó “hmax” for um nó novo, w^{ab} terá todos os valores

$w_{hmaxK}^{ab} = 1$ (para k = 1, ..., Nb), significando que a RNA não consegue fazer uma previsão com os casos apresentados para treinamento. Neste caso, a RNA pode ser treinada imediatamente através da apresentação do padrão de saída correto na rede ARTb, ativando a camada F_2^b.

4.5.1.1.2 TREINAMENTO

Caso a rede esteja sendo treinada, e a camada F_2^b esteja ativa, a saída do módulo de ligação será dado por $x_i^{ab} = \min(y_i^b, w_{hmaxi}^{ab})$ para i = 1, ..., Nb.

O módulo de ligação faz a pesquisa de similaridade, através da equação demonstrado no quadro 4.10.

Quadro 4.10 – Equação para pesquisa de similaridade

$$A_{ab} = \frac{\sum_{i=1}^{Nb} \min(x_i^{ab}, y_i^b)}{\sum_{i=1}^{p} x_i^{ab}} \geq \rho_{ab}$$

Onde, p_{ab} nível de vigilância do módulo de ligação.

Se $A_{ab} < p_{ab}$, o nível de vigilância da rede ARTa (p_a) é aumentado a um valor um pouco maior que $p_a > \phi_{hmax}$, sendo "hmax" o índice do nó vencedor da RNA RBF.

Quando isto ocorre, um novo nó é criado com as coordenadas do padrão de entrada. Este novo nó aprenderá a fazer a previsão correta através do treinamento do módulo de ligação.

Inicialmente os pesos do módulo de ligação w_{hk}^{ab} são iguais a 1. Durante o tempo em que o novo nó da RB estiver ativo, o vetor w_{hmaxi}^{ab} é igualado ao vetor x^{ab}. Uma vez que o nó "hmax" aprende a predizer a categoria "K" da rede ARTb, aquela associação torna-se permanente, isto é $w_{hmaxi}^{ab} = 1$ para sempre. O valor de vigilância p_a. é retornado para $p_a = p_a$ no intervalo entre apresentação de dois padrões consecutivos.

Se $A_{ab} \geq p_{ab}$, a camada RBF é treinada pelo método de aproximação sucessiva descrito anteriormente no item 4.3.2.1.

4.5.2 VANTAGENS

Tontini (1995) apresenta como vantagens da RNA RBF-Fuzzy-Artmap a sua arquitetura, que permite a incorporação das qualidades da rede RBF na rede Fuzzy Artmap, diminuindo a sensibilidade a ordem dos padrões de treinamento, ao mesmo tempo que mantém a capacidade de aprendizado instantâneo.

4.5.3 DESVANTAGENS

Tontini (1995) apresenta como desvantagens da RNA RBF-Fuzzy-Artmap a necessidade de um número maior de nós, na camada RBF, do que a rede Fuzzy-Artmap necessita na rede ARTa.

5. PRÉ-PROCESSAMENTO DE SINAIS

5.1 INTRODUÇÃO

Pré-processamento de sinais é uma forma de pré-organizar ou pré-ajustar os sinais com os quais será feito o processamento, de modo a otimizar esse processamento, procurando também garantir um maior sucesso na operacão (Fischer, 1999).

Segundo Hugo (1995), o pré-processamento de sinais é uma forma de organizar ou ajustar os sinais captados procurando torná-los passíveis de processamento na fase posterior do processo de reconhecimento de fala. Ele é responsável por gerar o vetor característico do padrão a ser analisado.

O uso do computador para a execução de processamento de sinal de som é quase fundamental. O computador não apenas processa informações matemáticas relativas ao sinal, mas também, pode capturar o sinal sonoro transformando-o em informação digital (Malvino, 1985 apud Tafner, 1996).

Som quantificado, e por assim dizer, registrado, possui características “cruas” que não são apropriadas para constituir um conjunto de treinamento para uma RNA. Dessa maneira, o processo de análise é importante pois faz parte de um mecanismo de refinamento da informação. Da mesma forma que não se pode colocar petróleo crú no tanque de combustível de um automóvel (movido a gasolina) na esperança de que o carro ande, não pode-se simplesmente jogar dados do mundo real em um conjunto de treinamento de uma rede neural artificial, e esperar que a rede os classifique corretamente (Lawrence, 1992 apud Tafner, 1996).

Dessa maneira, é necessário que seja feita uma análise do som como forma de extrair do seu sinal apenas a informação necessária para efetuar o treinamento da RNA (Tafner, 1996).

5.2 PROCESSAMENTO DO SINAL UTILIZANDO MÉDIA

O pré-processamento utilizando médias já foi estudado por diversas obras, e neste trabalho de pesquisa, tomou-se como fontes bibliográficas: Tafner (1996) e Hugo (1995), sendo utilizada a técnica de pré-processamento desenvolvida por Tafner (1996), descrita a seguir.

5.2.1 ETAPAS

Tafner (1996) menciona que, basicamente, o sinal amostrado está apenas digitalizado, e, portanto, precisa passar por algumas etapas que compõem o refinamento do sinal. Esse refinamento é necessário por pelo menos 3 motivos:

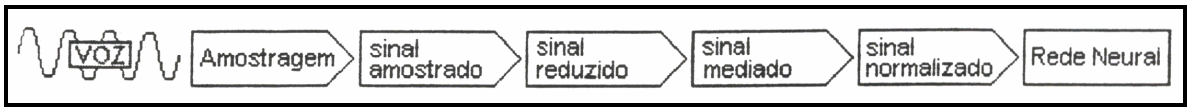
a) identificar os sinais mais significativos (envoltória);
b) diminuir a quantidade de sinais para compor a entrada da RNA;
c) melhorar a representatividade do sinal em relação ao sinal original.

As 3 propostas do refinamento acontecem em meio a 4 etapas criadas por Tafner (1996) para o processamento do sinal, sendo elas:

a) eliminação do ciclo negativo do sinal amostrado;
b) redução do sinal amostrado detectando a forma de onda (envoltória);
c) mediação do sinal reduzido;
d) normalização do sinal mediado.

Quando o sinal é convertido em sinal digital, é chamado de sinal amostrado. Quando o sinal amostrado é reduzido, é chamado de sinal reduzido, e quando mediado é chamado de sinal mediado. Depois da normalização o sinal é chamado de normalizado. Esses nomes foram atribuídos por Tafner (1996) como forma de se conhecer em que fase do processamento o sinal original se encontra, e, dessa forma, deduzir o que ainda falta fazer para que o sinal fique pronto para compor o conjunto de treinamento ou testar a RNA. Essas etapas são representadas na figura 5.1.

Figura 5.1 – Fluxo do sinal amostrado



Fonte: Tafner (1996)

5.2.1.1 ELIMINAÇÃO DO CICLO NEGATIVO

Tafner (1996) afirma que nesta etapa é determinado que todo o sinal negativo da onda deverá ser eliminado. Essa eliminação consiste em atribuir 128 para os sinais abaixo da linha de silêncio. Isto deve ser feito sem perturbar a ordem original dos sinais escolhidos, ou, ainda, eliminar a posição em que o sinal foi silenciado.

5.2.1.2 REDUÇÃO DO SINAL AMOSTRADO

Tafner (1996) menciona que a necessidade da redução do sinal é simples. A quantidade de valores gerados por uma leitura de uma palavra falada é incrivelmente alta. A uma taxa de 8000 Hz com 256 níveis de quantização (8 bits de precisão), 1 segundo de fala gera 8000 valores de 0 a 255 cada um, ou 8000 bytes. Levando-se em conta que se precisa apenas da forma de onda para identificar o sinal, a quantidade de 8000 bytes torna-se abundante.

Imaginando-se que a quantidade de entradas para a RNA seja de 100 entradas, a redução deverá diminuir o número de sinais amostrados para 100 bytes. Essa diminuição deve ser feita detectando a forma da onda (envoltória), ou seja, considera-se as amplitudes significativas (Tafner, 1996). Quanto à redução, se o sinal amostrado possue 6000 bytes, 1 em cada 60 deverá ser escolhido para compor o sinal processado. Essa taxa de redução é obtida a partir do cálculo demonstrado no quadro 5.1.

Quadro 5.1 – Cálculo para taxa de redução

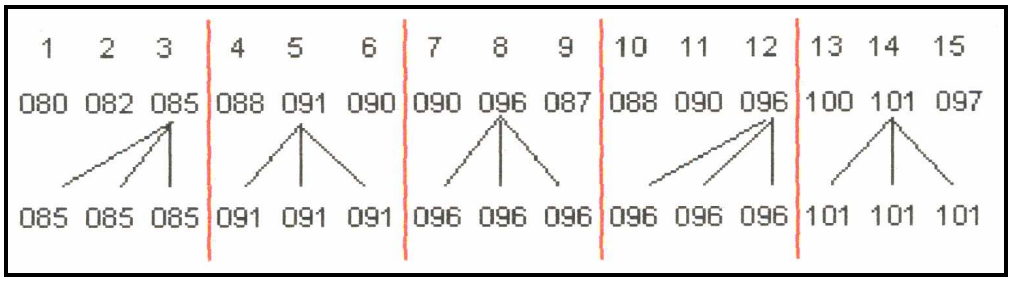
$$\textbf{Taxa de redução} = \frac{\textbf{quantidade de sinais amostrados}}{\textbf{100}}$$

Tafner (1996) menciona que é neste momento em que a forma de onda deve ser detectada, ou seja, o sinal escolhido entre os 60 sinais deve ser o mais significativo possível. Isto quer dizer que o valor de maior amplitude no intervalo é o valor que deve ser escolhido para compor o sinal amostrado.

5.2.1.3 MEDIAÇÃO DO SINAL REDUZIDO

Tafner (1996) descreve que a etapa de mediação do sinal reduzido, consiste em uma espécie de segunda redução do sinal, porém, sem perda de posição, ou seja, mantendo-se as 100 posições originais. Esse método, trata, basicamente, de igualar 3 sinais consecutivos pelo valor mais alto dos 3. Isso deve ser feito ao longo de todas as 100 posições, e pode-se observar uma representação desta etapa na figura 5.2.

Figura 5.2 – Mediação de 3 sinais em um sinal amostrado

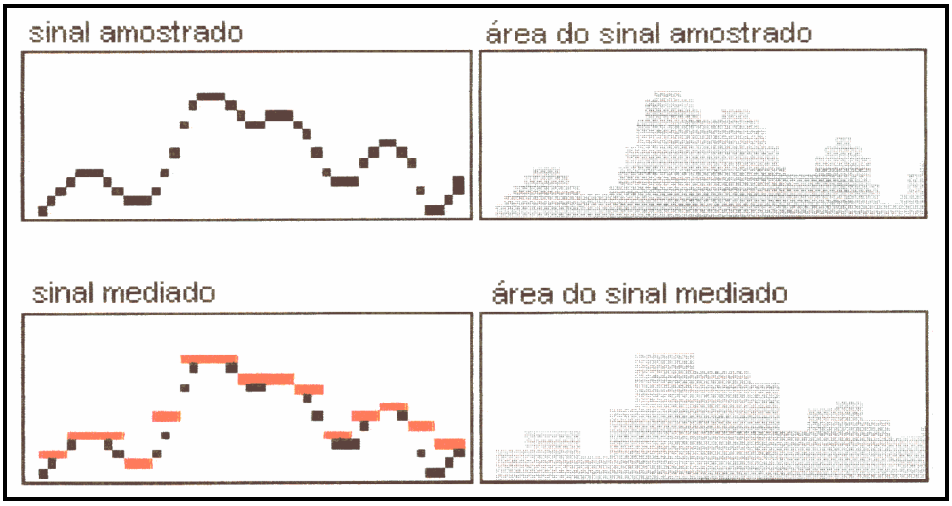


Fonte: Tafner (1996).

Tafner (1996) afirma que no processo da mediação é certo que ocorre uma redução do sinal de 100 para 33 bytes, porém, não há redução das posições, e isto faz com que o sinal fique ainda mais intenso quando for apresentado à RNA.

Conforme a figura 5.3, nota-se que a área do sinal mediado possui características bastante fortes que a identificam com a área do sinal amostrado. O mérito desta técnica é que apesar de reduzir ainda mais a quantidade de sinais, mantém, ainda, a identidade do sinal de forma bastante significativa (Tafner, 1996).

Figura 5.3 – Representação gráfica do sinal antes e depois da mediação



Fonte: Tafner (1996)

5.2.1.4 NORMALIZAÇÃO DO SINAL MEDIADO

Esta etapa procura anular a diferença da intensidade do sinal como um todo, ou seja, falar duas vezes a mesma palavra, mas com alturas diferentes, produzirá, consequentemente, dois sinais com intensidades diferentes, apesar da mesma forma de onda. Esse desalinhamento de sinais pode ser causado por diversos motivos, o mais comum e fácil de ocorrer é a distância da boca em relação ao microfone. Quanto mais perto a boca estiver do microfone maior será a intensidade do sinal, e vice-versa (Tafner, 1996).

Tafner (1996) menciona que o método utilizado para suprimir essa deficiência é um método matemático conhecido como normalização. O método consiste em ajustar os valores de uma série qualquer em relação a um valor especifico. No nosso caso, basicamente, consiste em igualar todos os valores do sinal mediado à intensidade de 150[26], eliminando, assim, diferenças de intensidade entre os outros sinais. O valor x_i do sinal mediado x deve ser multiplicado pelo valor de 150 e, depois, dividido pelo mais alto valor encontrado em todo o sinal x, conforme cálculo contido no quadro 5.2.

Quadro 5.2 – Cálculo para obter valor normalizado

$$\text{Valor normalizado} = \frac{\text{valor mediado} * 150}{\text{maior valor do sinal}}$$

Tafner (1996) menciona que dessa forma, o valor de mais alta intensidade do sinal será igualado a 150, e os demais valores igualados de forma proporcional ao valor de 150.

5.2.2 CONSIDERAÇÕES FINAIS

Essas etapas foram implementadas no protótipo conforme proposto por Tafner (1996), obedecendo a especificação presente no item 7.2. Caso restarem dúvidas quanto ao pré-processamento utilizado neste trabalho de pesquisa, elas podem ser respondidas consultando-se Tafner (1996).

6. OLE AUTOMATION

6.1 INTRODUÇÃO

Neste capítulo que trata sobre *OLE Automation*, *Automation OLE*, ou também chamado de *Automation* (segundo recentes documentações da Microsoft Corporation), serão mencionados conceitos, formas de utilização e inovações sobre á tecnologia COM, fornecida pela empresa desenvolvedora do Sistema Operacional (SO) Windows, a Microsoft Corporation. Abordar-se-á também, o uso desta tecnologia de alto nível, baseada em COM, no ambiente de programação Delphi 5, utilizando *Object Pascal* como linguagem de programação.

6.2 EVOLUÇÃO

O Microsoft Visual Basic foi o primeiro ambiente para desenvolvimento a introduzir a idéia do fornecimento de componentes de software para o mercado de massa, utilizando-se do conceito de componentes de software reutilizáveis, provindo das teorias de programação orientada á objetos (POO) (Cantú, 2000).

O primeiro padrão técnico promovido pelo Visual Basic foi o VBX, uma especificação de 16 bits que posteriormente, com a evolução para 32 bits, passou a ser chamado de OLE Controls ou ActiveX (Cantú, 2000).

Como próximo passo, foi construir o *shell* do Windows 95 usando a tecnologia OLE e naturalmente, interfaces. Através de estratégias de mercado, passou a chamar os controles OLE para controles ActiveX (Cantú, 2000).

A medida que a tecnologia foi ampliada e tornou-se cada vez mais importante para a plataforma Windows, a Microsoft mudou o nome para OLE e depois para COM, e em sua última versão do SO Windows, o Windows 2000, chamasse COM+.

6.3 TECNOLOGIA COM

Criado em 1993, o *Component Object Model* (COM) pode ser considerado uma tecnologia de empacotamento, um grupo de convenções de bibliotecas de suporte que possibilita a interação entre diferentes partes de software de uma maneira consistente e orientada a objetos (Honorato Junior, 2000). Cantú (2000) cita que COM é uma tecnologia que define um modo padronizado para um módulo-cliente e um módulo-servidor se comunicarem através de uma interface específica, sendo “módulo”, um aplicativo ou biblioteca e interface (DLL). As interfaces são implementadas por objetos servidor e um objeto servidor pode implementar mais de uma interface, sendo que todos os objetos servidor possuem algo em comum, a interface *Iunknown*.

O objetivo das interfaces COM é se comunicar entre dois módulos de software, dois arquivos executáveis ou um arquivo executável e uma biblioteca (DLL) (Cantú, 2000).

6.4 AUTOMATION OLE

Cantú (2000) menciona que *Automation OLE* é uma das estratégias utilizadas para que aplicativos comuniquem entre si, no SO Windows. Menciona ainda, que no SO Windows, os aplicativos não são isolados, e frequentemente os usuários querem que eles interajam. A área de transferência e o DDE oferecem um mundo muito simples para aplicativos interagirem, pois possibilitam a troca de dados entre aplicativos. Por sua vez, cresce a demanda que programas ofereçam uma interface *Automation OLE* para permitir que outros programas os conduzam.

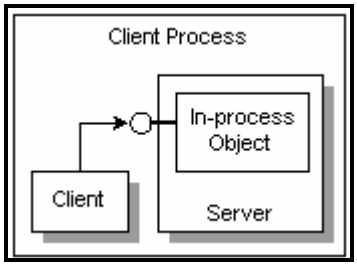
Borland (1999) define um servidor *Automation OLE* como sendo aplicações que expõem suas funcionalidades para serem chamadas por aplicações clientes (*Automation Controllers*). As aplicações clientes podem ser qualquer aplicação que suporte *Automation OLE*, podendo ser escritas, por exemplo, em ambiente Delphi, Visual Basic, ou C++ Builder. Um objeto *Automation* OLE pode ser uma aplicação ou uma biblioteca (Borland, 1999).

6.4.1 TIPOS DE AUTOMATION OLE

Borland (1999) afirma que na tecnologia COM, um cliente não precisa saber onde um objeto reside, faz simplesmente uma ligação à interface de um objeto. COM executa os passos necessários para fazer a ligação. Estes passos diferem dependendo se o objeto reside no mesmo processo como o cliente (*in-process*), em um processo diferente na máquina do cliente (*out-of-process*), ou em uma máquina diferente pela rede (*remote server*). Os tipos diferentes de servidores conhecidos são:

a) servidor no mesmo processo (*in-process*): uma biblioteca (DLL) executando no mesmo espaço de processo do cliente, mostrado na figura 6.1. Por exemplo, um controle de ActiveX embutiu em uma página de Web vista debaixo do Internet Explorer ou Netscape. Neste caso o controle ActiveX é carregado na máquina do cliente e foi chamado no mesmo processo do browser cliente. O cliente comunica com o servidor *in-process* usando as chamadas diretas da interface de COM;

Figura 6.1 – Exemplo de servidor in-process



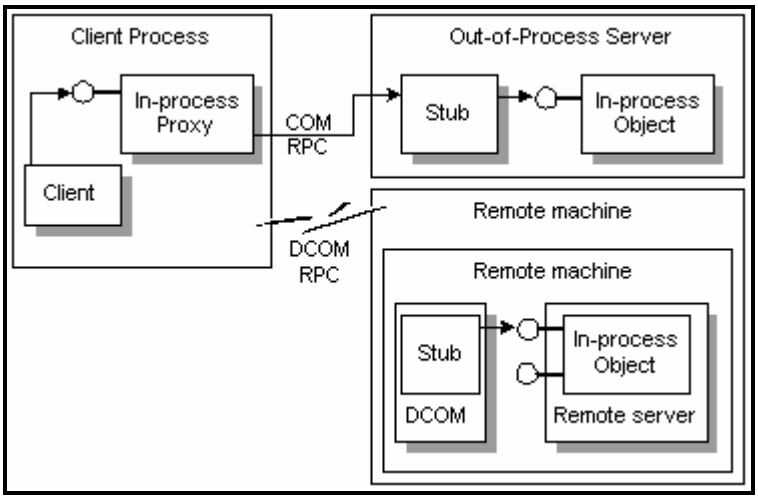
Fonte: Borland (1999)

b) servidor fora do processo (*out-of-process*): seria uma outra aplicação (.EXE) executando em um espaço de processo diferente, mas na mesma máquina do cliente, mostrado na figura 6.2. Por exemplo, uma planilha do Microsoft Excel inserida em um documento no Microsoft Word, onde tem-se duas aplicações separadas que executam na mesma máquina. O servidor local usa COM para comunicar com o cliente;

c) servidor remoto (*remote server*): uma DLL ou uma aplicação que executam em uma máquina diferente da do cliente, mostrado na figura 6.2. Por exemplo, uma aplicação de banco de dados em *Object Pascal* (Delphi) conectada a um servidor de aplicação em uma outra máquina na rede.

Servidores remotos utilizam *Distributed Component Object Model* (DCOM) interfaces para comunicar com o servidor de aplicação.

Figura 6.2 – Exemplo de servidor fora de processo e servidor remoto



Fonte: Borland (1999)

Servidores *Automation OLE* só podem ser registrados nos tipos *in-process* e *out-of-process* no ambiente Delphi (Borland, 1999).

6.5 AUTOMATION OLE E DELPHI

Servidores são de fácil criação no ambiente Delphi, devido ao trabalho extensivo que a *Visual Component Library* (VCL) e o compilador realizam para isolar os desenvolvedores de seus detalhes complexos (Cantú, 2000). Para o Delphi oferecer suporte à *Automation OLE*, ele fornece assistentes de desenvolvimento de servidor e um editor de *Type Librarys* (item 6.5.1.2), além de aceitar interfaces duais.

6.5.1 MODO DE ACESSO

Segundo Cantú (1999) existem duas maneiras pelo qual um cliente pode chamar os métodos expostos por um servidor:

a) *Variant*: solicita-se a execução de um método, passando seu nome em uma *string* (cadeia de caracteres), de maneira semelhante à chamada dinâmica de uma DLL. Cantú (1999), cita que esta técnica é de fácil implementação, mas não tem uma performance satisfatória em termos de velocidade e verificação do compilador;

b) *Interface*: pode-se importar a definição de uma interface Delphi do objeto no servidor e chamar seus métodos de maneira mais direta (executando simplesmente um número). Cantú (1999) cita que utilizar esta técnica baseada em interfaces, permite que o compilador verifique os tipos dos parâmetros e produza código mais rápido, mas ela exige um pouco mais de trabalho do desenvolvedor, fazendo com que o desenvolvedor vincule o seu aplicativo cliente a uma versão específica do servidor.

6.5.1.1 ACESSO ATRAVÉS DE VARIANT

Conforme item 6.5.1 um cliente pode acessar um servidor, utilizando o tipo de variável *variant* do Delphi, não tendo com isso nenhuma informação de tipo sobre o servidor que está usando. Internamente, toda a chamada de função tem que ser executada no servidor usando o método *Invoke* de *Idispatch*, passando o nome da função como um parâmetro de string e esperando que o nome corresponda a uma função existente do servidor (Cantú, 2000). Um exemplo deste tipo de acesso através de *variant* é demonstrado no quadro 6.1.

Quadro 6.1 – Exemplo de acesso através de Variant

```
var
    Varw: Variant;
Begin
    Varw := CreateOleObject('PrototipoRF.Create');
    Varw.Inicializa;
    Varw.Reconhece;
End;
```

O tipo de dados *varian*t (*type variant*), pode assumir diferentes tipos de dados, incluindo um objeto COM que ofereça suporte à interface *Idispatch*. As *variants* têm seu tipo verificado em tempo de execução. Desta forma que o compilador pode compilar o código mesmo que não saiba a respeito dos métodos do servidor *Automation OLE* em Delphi (Cantú, 2000).

6.5.1.2 ACESSO ATRAVÉS DE INTERFACE

A principal diferença deste tipo de acesso ao servidor, em relação ao acesso *variant*, é que este tipo de acesso requer uma *Type Library*, uma das bases das tecnologias OLE e COM (Cantú, 2000).

Type Library é basicamente um conjunto de informações de tipo. Esse conjunto de informações geralmente descreve todos os elementos (objetos, interfaces e outras informações de tipo) disponibilizados pelo servidor. As *Type Librarys* são independentes de linguagem. Os elementos de tipos são definidos pelo OLE como um subconjunto dos elementos padrão das linguagens de programação, e qualquer ferramenta de desenvolvimento pode usá-los (Cantú, 2000).

Cantú (2000) cita que exportar a descrição de suas interfaces e objetos usando um *Type Library* é mais seguro, devido ao fato que está *Type Library* pode ser convertida por uma ferramenta específica, no caso Delphi, em definições escritas na linguagem que o desenvolvedor utiliza para escrever a aplicação cliente, no caso *Object Pascal*. Com este fato, é possível para o compilador verificar se o código está correto.

Tendo o compilador feito suas verificações, ele pode usar uma das duas técnicas diferentes para enviar a solicitação ao servidor, conforme Cantú descreve, que são:

a) elemento *Vtable*: uma entrada em uma declaração de tipo de interface, ou, chamada direta no elemento *Vtable*;
b) elemento *dispatch* interface: um modo de mapear cada entrada de uma interface em um número, podendo ser executadas as entradas do servidor pelo número.

A capacidade de conectar a um servidor de duas maneiras diferentes, usando uma estratégia mais dinâmica ou estática, chama-se interfaces duais (Cantú, 2000). Isto significa que, ao desenvolver um servidor *Automation OLE*, pode-se acessar os métodos de um servidor de duas maneiras: pode usar ligação tardia e o mecanismo fornecido pelo elemento *dispinterface* ou pode usar vinculação prematura e o mecanismo baseado nos elementos *Vtables*, os tipos *inferface* (Cantú, 2000).

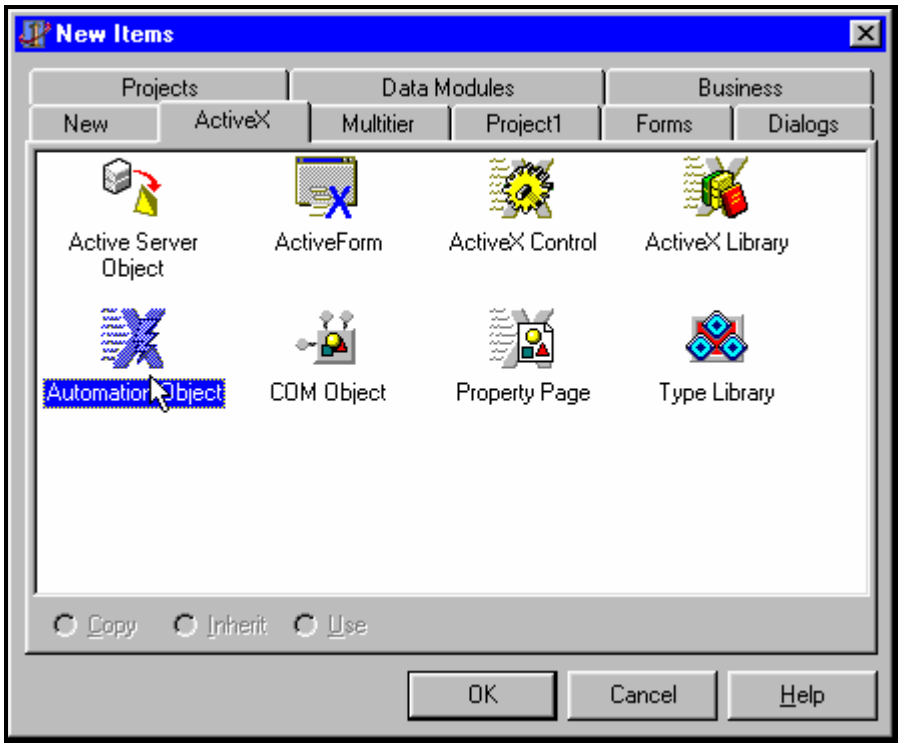
Cantú (2000) ainda cita que, dependendo da técnica a se utilizar, resulta em uma execução mais rápida e mais lenta. Procurar uma função pelo nome, e realizar a verificação de tipo em tempo de execução é a estratégia mais lenta (*dispinterface*); ou usar a chamada direta à *Vtable*, *que* é a com melhor performance.

6.5.2 CRIANDO UM SERVIDOR AUTOMATION OLE

Quando há a necessidade da criação de um servidor *Automation OLE*, usa-se o assistente *Automation object* do Delphi. Os passos iniciais para criação de um servidor *Automation OLE* são descritos abaixo:

a) Inicia-se primeiramente uma nova aplicação no ambiente Delphi, selecionando-se o menu File, e posteriormente a opção *New*. Abrirá a janela com o título "*New items*", demonstrada na figura 6.3;

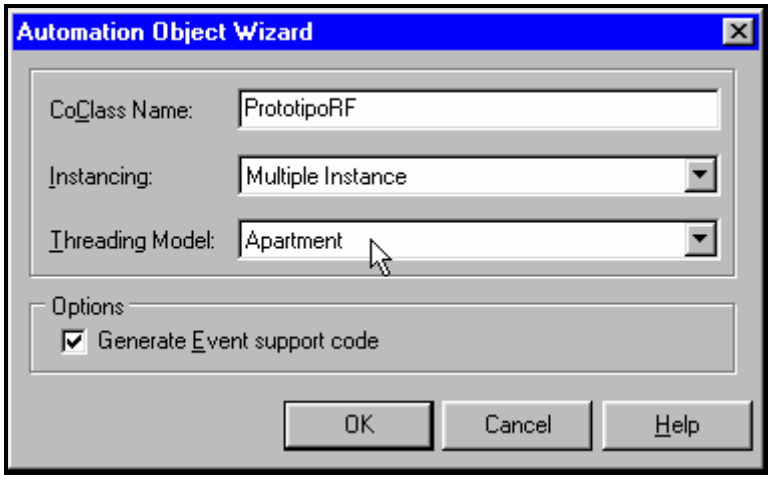
Figura 6.3 – Janela "New Items" do ambiente Delphi



Fonte: Borland (1999)

b) Selecionar a página "ActiveX", e o ícone "*Automation Object*". Abrirá a janela com o título "*Automation Object Wizard*", demonstrada na figura 6.4;

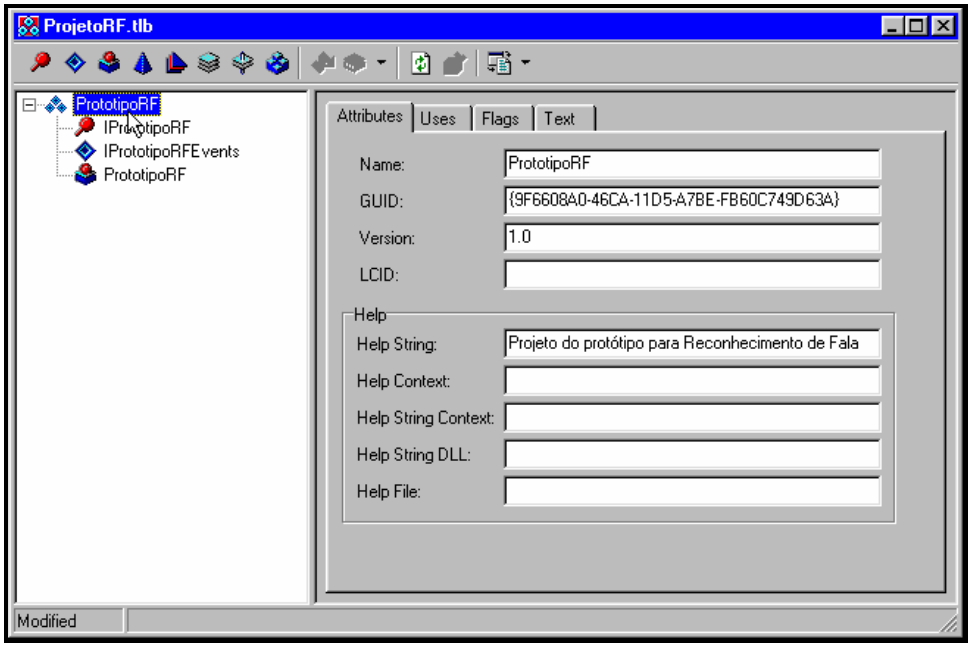
Figura 6.4 – Janela "Automation Object Wizard" do ambiente Delphi



Fonte: Borland (1999)

c) Dever-se-á entrar com as informações referentes ao objeto *Automation OLE* que se deseja criar, e posteriormente clica-se no botão "OK". Abrirá a janela com o título "ProjetoRF.tlb", demonstrada na figura 6.5, que é na verdade o editor de *Type Librarys* do Delphi. Nele deverão ser definidas as propriedades e métodos do objeto *Automation OLE* que se deseja criar.

Figura 6.5 – Editor de Type Librarys do Delphi



Fonte: Borland (1999)

Neste trabalho de pesquisa, não pretende-se descrever todos os passos do desenvolvimento até a conclusão de um objeto *Automation OLE*, porque seria descrito no ambiente Delphi e há outras linguagens que o suportam, além do fato da possibilidade de

configurar diferentes formas de desenvolvimento, acesso e utilização e por estas informações poderem ser obtidas em Cantú (2000) e Borland (1999).

6.5.3 REGISTRO DE UM SERVIDOR AUTOMATION OLE

A unidade que contém o objeto servidor possui uma instrução incluída pelo Delphi na seção *initialization*, demonstrada no quadro 6.2.

Quadro 6.2 – Seção initialization de um servidor

```
initialization
  TautoObjectFactory.Create(ComServer, TprototipoRF, Class_PrototipoRF,
  ciMultiInstance);
end;
```

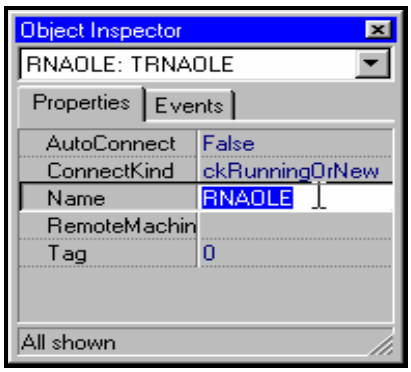
O desenvolvedor pode registrar um servidor de duas maneiras, como já mencionado no item 6.3.1:

a) no processo: no próprio ambiente do Delphi, escolhe-se a opção *Run* e por sua vez a opção *Register ActiveX Server*. Para retirar o registro, na mesma opção *Run*, selecione a opção *Unregister ActiveX Server*;
b) fora do processo: executa-se o servidor com o parâmetro */regserver* na linha de comando e para retirar o registro */unregserver* na linha de comando.

6.5.4 PROPRIEDADES PADRÕES CLASSE AUTOMATION OLE

Conforme Borland (1999), as classes *Automation OLE* possuem algumas propriedades padrões. Essas propriedades padrões podem ser acessadas em tempo de execução, ou seja, quando o projeto está sendo executado, ou em tempo de desenvolvimento, no ambiente Delphi, através do “Object Inspector” (figura 6.6).

Figura 6.6 – Object Inspector das Classes Automation OLE no Delphi



A propriedade “AutoConnect” indica quando inicializar o componente servidor em tempo de projeto. Como alternativa, a classe *Automation OLE* pode ser iniciada na primeira vez em que um de seus métodos é chamado em execução.

Outra propriedade padrão das classes *Automation OLE* é “ConnectKind”, que indica como estabelecer a conexão com o servidor. Esta propriedade pode assumir cinco valores para estabelecer a conexão com o servidor, que são:

a) “ckNewInstance”: sempre cria uma nova instância de servidor;
b) “ckRunningInstance”: utiliza a instância do servidor que está em execução no momento, caso não haja nenhuma em execução, causa uma violação de acesso (*Access Violation*);
c) “ckRunningOrNew”: seleciona a instância do servidor em execução, ou cria uma nova no caso de não haver uma sendo executada;
d) “ckRemote”: utiliza um servidor remoto que deverá ser descrito na propriedade “RemoteMachineName”;
e) “ckAttachToInterface”: não liga-se a nenhuma instância do servidor. Ao invés, a aplicação provê uma interface que usa o método “ConnectTo” que é introduzido em classes descendentes. Lembra-se que esta opção não pode ser utilizada com a propriedade “AutoConnect”.

Borland (1999) menciona que a propriedade “Name” especifica o nome pelo qual o objeto será referenciado no código fonte, e que a propriedade “Tag” armazena um valor inteiro, e não tem função predefinida. “Tag” deve ser utilizada conforme necessidade pelos desenvolvedores.

Dúvidas referentes as propriedades padrões de objetos e tipos de dados da linguagem Object Pascal podem ser esclarecidas em Borland (1999).

6.5.5 VANTAGENS

Cantú (2000) afirma que além da vantagem óbvia da automação programada, comparada às operações manuais do usuário, essas interfaces *Automation OLE* são completamente neutras com relação à linguagem, mesma vantagem citada por Steve (2000). Cantú (2000) cita ainda, como exemplo, que linguagens de programação como

Object Pascal (Delphi), C++ e Visual Basic podem ativar um servidor *Automation OLE*, independente da linguagem de programação utilizada para escrevê-lo.

6.5.6 DESVANTAGENS

Tecnologias OLE e COM não oferecem suporte a todos os tipos disponíveis no ambiente Delphi. Este é um fato que deve ser considerado quando optar-se pelo uso desta tecnologia, pois o cliente e o servidor são freqüentemente executados em diferentes espaços de endereço, e o sistema precisa mover os dados de um lado para outro, não esquecendo que *interfaces* OLE devem ser acessíveis por programas escritos em qualquer linguagem (Cantú, 2000).

7. PROTÓTIPO

7.1 INTRODUÇÃO

Este trabalho foi desenvolvido com o objetivo de proporcionar ao desenvolvedor de aplicativos a interface com reconhecimento de fala através de RNA, utilizando o método de RNA *RBF-Fuzzy Artmap*. O protótipo deste trabalho de pesquisa foi dividido em três etapas:

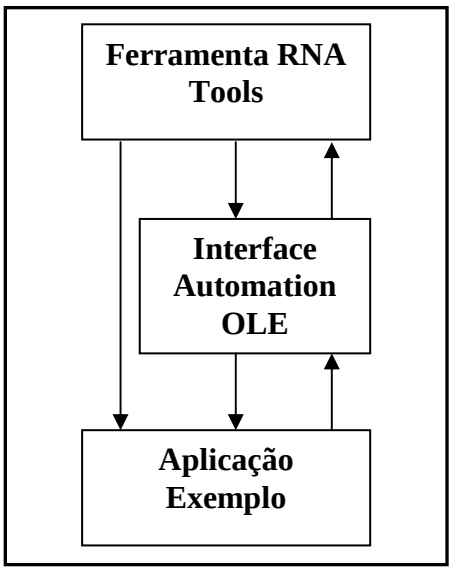
a) ferramenta de treinamento: local onde é criado um novo projeto de reconhecimento de fala, e nele efetua-se a aquisição de sinal (palavras faladas por um locutor), o pré-processamento (média ponderada), treina-se a RNA *RBF-Fuzzy Artmap* e salva-se como fonte de conhecimento para ser utilizada pela interface *Automation OLE*. Denomina-se "*RNA Tools*";

b) interface *Automation OLE*: objetos que farão a ligação entre o aplicativo do desenvolvedor e as classes criadas para reconhecimento de fala, independente de linguagem de programação. Será adicionada no software do desenvolvedor, e fará a ligação entre o projeto criado anteriormente na ferramenta de treinamento com a aplicação;

c) aplicação exemplo: foi desenvolvido um aplicativo simulando uma aplicação corriqueira de um desenvolvedor de software, que utilizará as classes resultantes deste trabalho, possibilitando ao seu usuário interagir com o aplicativo por comandos pronunciados, criando uma interface de reconhecimento de fala.

Na figura 7.1, pode ser verificada a relação entre ferramenta, interface e aplicação, onde a ferramenta gerará o arquivo contendo as informações referentes a RNA, que será relacionado nas classes pela aplicação exemplo.

Figura 7.1 – Relação entre: ferramenta, interface e aplicação.



7.2 ESPECIFICAÇÃO

Será demonstrada a modelagem do protótipo através da linguagem UML (Furlan, 1998). Foram utilizados os diagramas de casos de uso, de classes e de seqüência, desenvolvidos na ferramenta de modelagem Rational Rose, da empresa de software Rational.

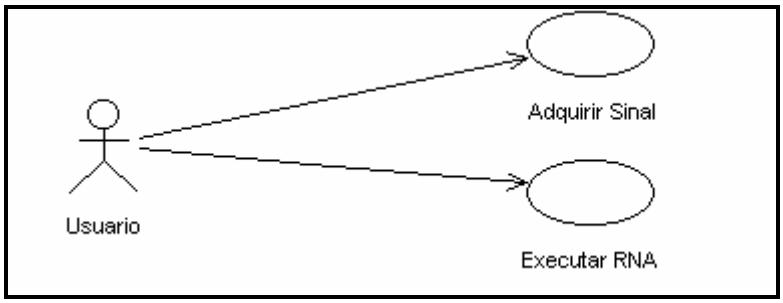
7.2.1 INTERFACE AUTOMATION OLE

As classes *Automation OLE* criadas para efetuar a interface entre a aplicação exemplo e a RNA para reconhecimento de fala são: TRNAOLE e TSINALOLE, que serão melhor vistas observando-se os subitens seguintes.

7.2.1.1 DIAGRAMA DE CASO DE USO

Pode-se observar dois casos de uso das classes *Automation OLE* na figura 7.2.

Figura 7.2 – Casos de uso das classes Automation OLE



Para as classes *Automation OLE* foram observados dois casos de uso:

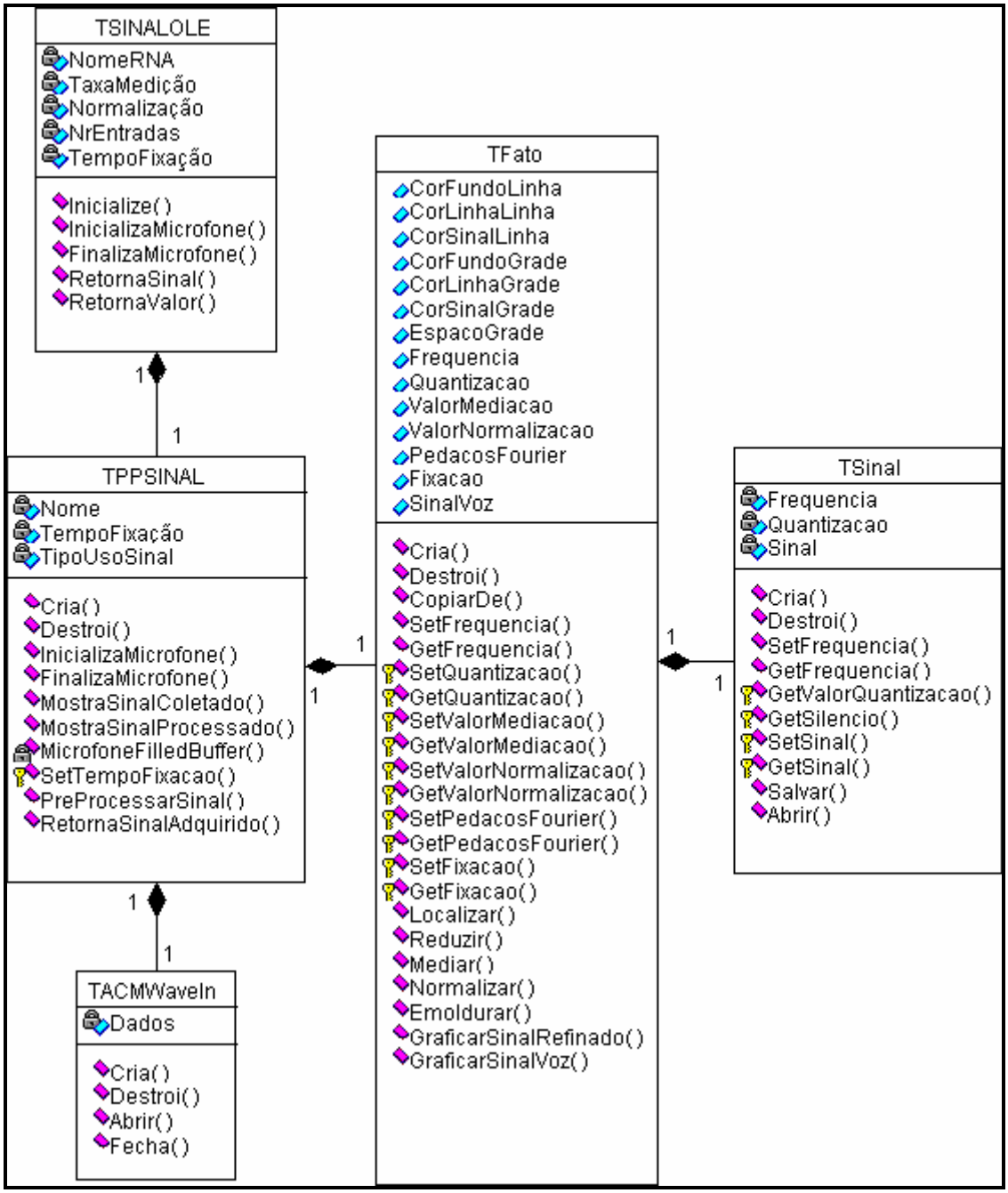
a) “Adquirir Sinal” (TSinalOLE): o usuário deverá fazer uma locução ao microfone, captando o sinal, e após isto, efetuar o pré-processamento na RNA;
b) “Executar RNA” (TRnaOLE): o usuário deverá passar os valores adquiridos e já pré-processados, como entradas para a RNA, e logo após efetuar a sua execução para verificar se a palavra foi ou não identificada.

7.2.1.2 DIAGRAMA DE CLASSES

7.2.1.2.1 DIAGRAMA DE CLASSES TSINALOLE

O diagrama de classes da classe TSinalOLE é demonstrado na figura 7.3.

Figura 7.3 – Diagrama de classes da classe Automation OLE TSinalOLE



Relacionadas à classe TSinalOLE, que é a classe para interface na aplicação, que possibilita a aquisição de sinal e pré-processamento, estão:

a) TPPSinal: é a classe responsável pela aquisição de sinal e pré-processamento;

b) TACMWaveIn: esta classe tem a finalidade de adquirir os sinais vindos da placa de som, através de APIs do Windows;

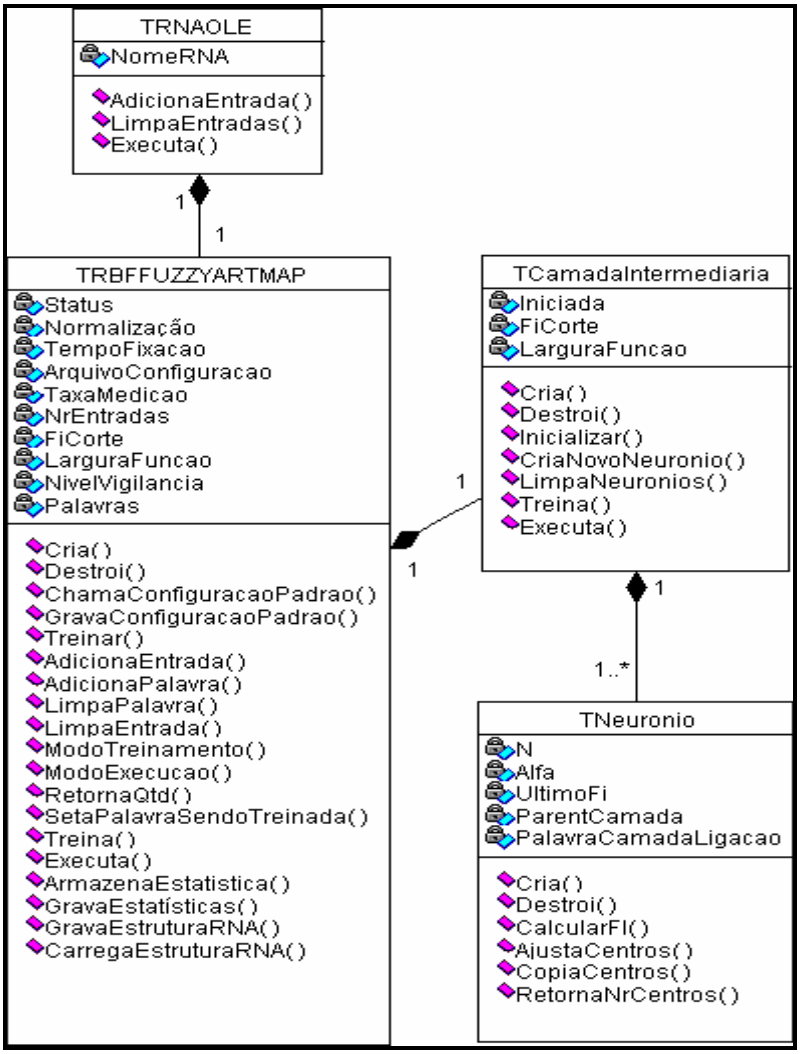
c) TFato: classe para tratar informações referentes ao sinal, pré-processamento e fornecer informações para efetuar a demonstração do sinal;

d) TSinal: a classe TSinal tem a função de estabelecer início e fim do sinal adquirido, identificando zonas de silêncio e mecanismos para possibilitar salvar e abrir um *buffer* de som, previamente capturado.

7.2.1.2.2 DIAGRAMA DE CLASSES TRNAOLE

O diagrama de classes, da classe TRnaOLE, é demonstrado na figura 7.4.

Figura 7.4 – Diagrama de classes da classe Automation OLE TRNAOLE



Relacionadas à classe TRnaOLE, que é a classe para interface na aplicação, que cria o mecanismo de ligação entre a RNA criada e treinada na ferramenta à aplicação final do usuário, estão:

a) TRBFFuzzyArtmap: é a classe implementada conforme o modelo de RNA *RBF-Fuzzy Artmap*, citada no capítulo 6. É responsável pelos atributos da RNA, treinamento e execução, gerenciando todas as ações relevantes à RNA;

b) TCamadaIntermediaria: tem a função de controlar todos os neurônios da(s) camada(s) RBF;

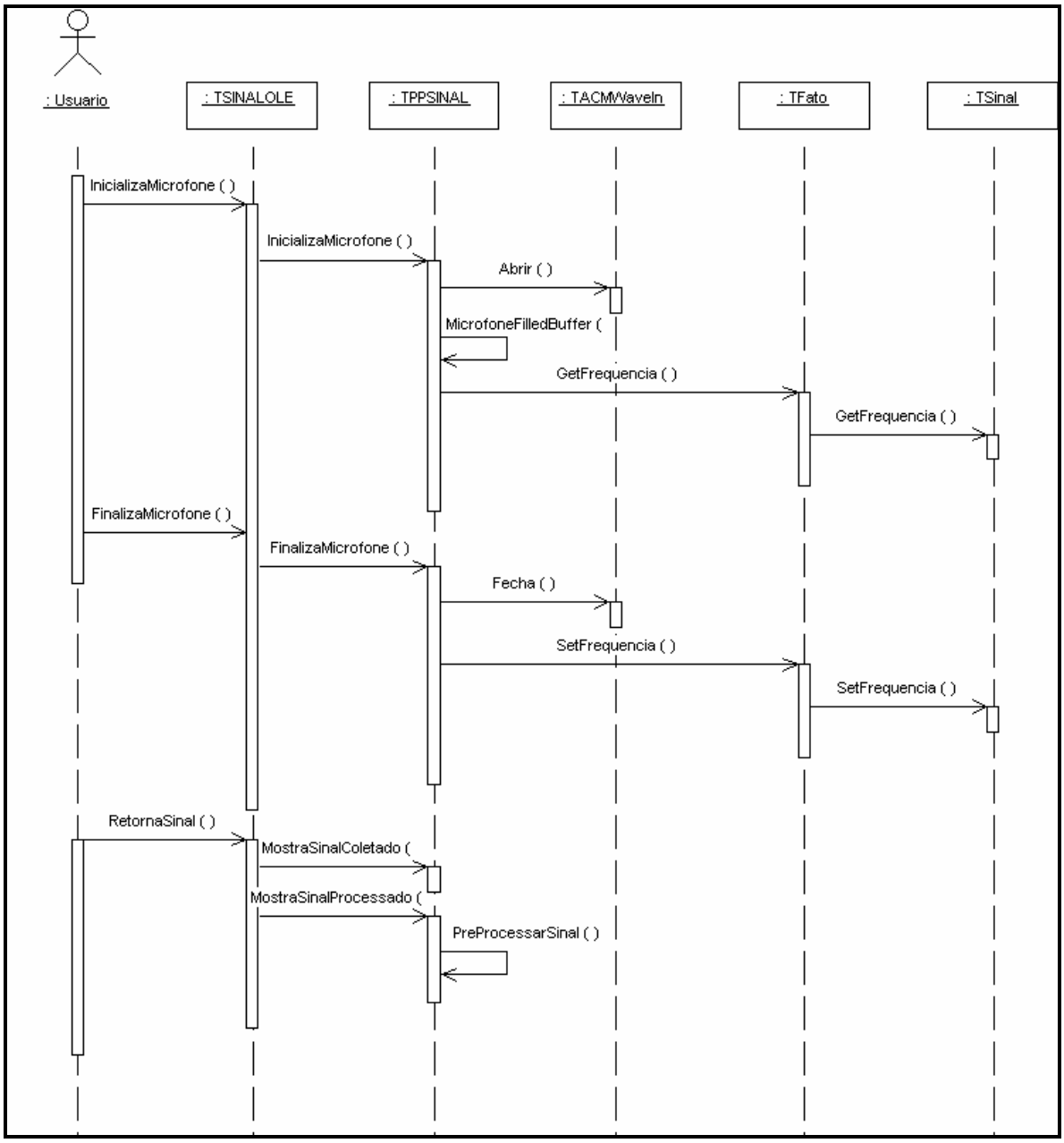
c) TNeuronio: classe para armazenar informações relevantes ao treinamento, execução e reconhecimento de padrões, possuindo capacidade de ajuste e ativação.

7.2.1.3 DIAGRAMA DE SEQÜENCIA

Segundo Colling (2000) os diagramas de seqüência representam, como o próprio nome indica, a seqüência em que as ações ocorrem dentro do sistema. Demonstram como é feita a troca de mensagens entre as classes. Como para cada caso de uso há um diagrama de seqüência, criou-se dois diagramas de seqüência, sendo possível observar o primeiro deles na figura 7.5.

7.2.1.3.1 DIAGRAMA DE SEQÜÊNCIA TSINALOLE

Figura 7.5 – Diagrama de seqüência da classe Automation OLE TSINALOLE



No diagrama de seqüência mostrado na figura 7.5, provindo do caso de uso “Adquirir Sinal”, citado no item 7.2.1.1, pode-se observar a presença de três métodos:

a) “InicializaMicrofone”: tem a finalidade de iniciar a captação do sinal;

b) “FinalizaMicrofone”: possui a finalidade de finalizar a captação do sinal; e

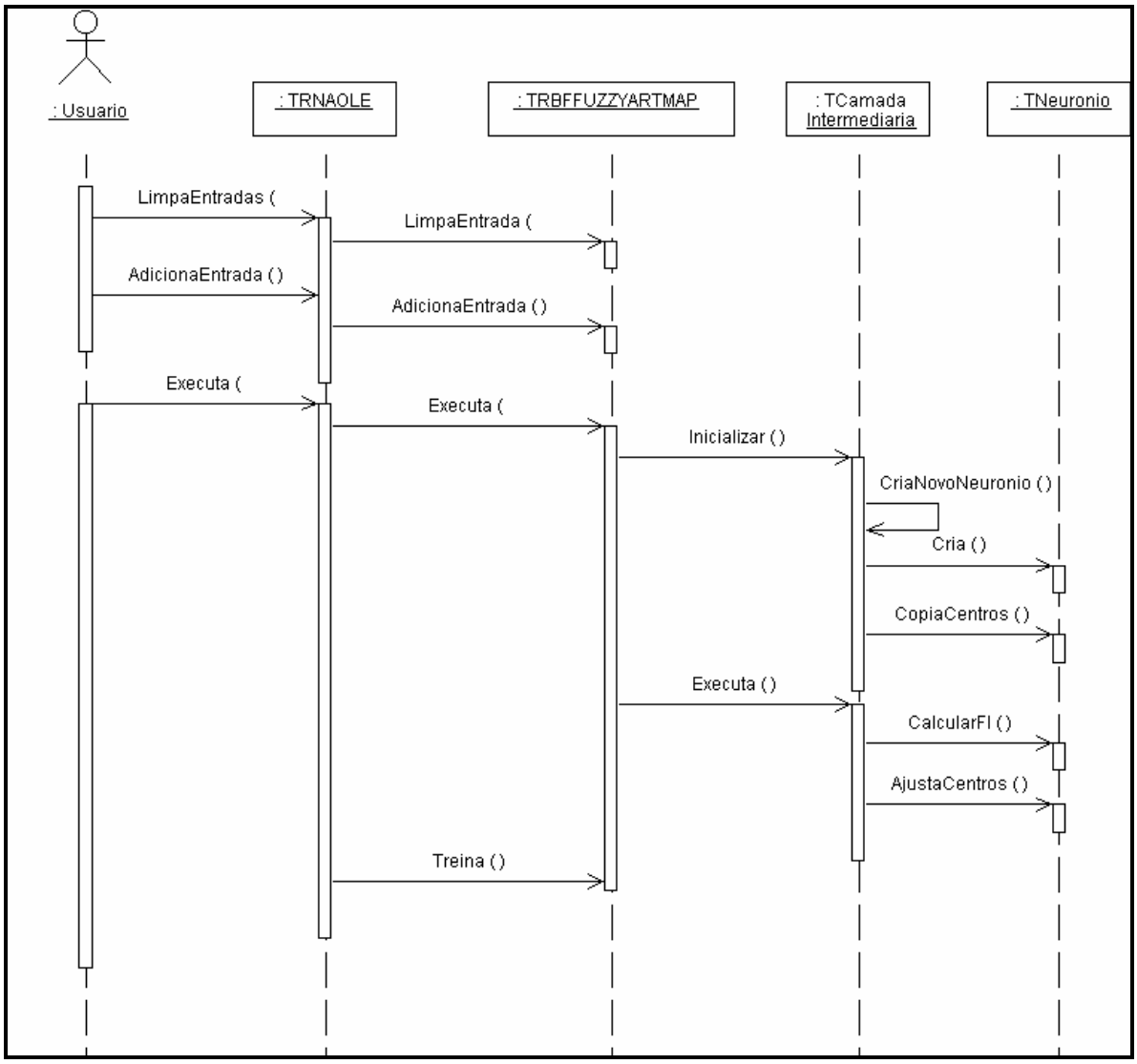
c) “RetornaSinal”: tem a função de retornar o sinal captado.

Para cada um deles, são executadas novas ações que são passadas para as classes subsequentes também através de métodos, conforme demonstrado na figura 7.5.

7.2.1.3.2 DIAGRAMA DE SEQÜÊNCIA TRNAOLE

O segundo diagrama de seqüência, que originou-se do caso de uso “Executar RNA” pode ser observado na figura 7.6.

Figura 7.6 – Diagrama de seqüência da classe Automation OLE TRNAOLE



O caso de uso “Executar RNA”, mencionado no item 7.2.1.1, originou o diagrama de seqüência mostrado na figura 7.6, e pode-se observar a presença de três métodos:

a) “LimpaEntradas”: inicializa novamente o vetor de entradas da RNA, com valores iguais à 0 (zero);
b) “AdicionaEntradas”: alimenta o vetor de entradas da RNA, com os valores já pré-processados;

c) “Executa”: tem a função de realizar o reconhecimento do padrão de entrada na RNA, observando-se que para a classe TRnaOLE, apenas o modo de execução está habilitado, o que para a ferramenta que utiliza a classe TRbfFuzzyArtmap, não é verdadeiro, pois faz-se necessário a opção de treinamento. Diferem-se no ponto em que uma (execução), não faz a atualização de centros e também, não cria novos neurônios e a outra (treinamento), faz.

Para cada um deles, são executadas novas ações que são passadas para as classes subsequentes também através de métodos, conforme demonstrado na figura 7.6.

7.3 DEFINIÇÕES

7.3.1 AQUISIÇÃO DE SINAL

A aquisição de sinal foi feita através da utilização de APIs do Windows presentes em classes desenvolvidas em ambiente Delphi, por Tafner (1996). São também disponibilizadas através de uma interface *Automation OLE*, juntamente com o pré-processamento.

7.3.2 PRÉ-PROCESSAMENTO

Baseando-se nas conclusões obtidas por Tafner (1996), onde conclui-se que a técnica de redução e detecção da envoltória (registro de intensidade) mostrou-se correta como elemento de percepção do som emitido, e as técnicas de mediação e normalização executaram de forma apropriada e generalizada os sinais, representando-os de forma mais concisa e discreta. Estas 4 técnicas adotadas por Tafner (1996) foram intituladas de pré-processamento por média, e foi mencionado neste trabalho de pesquisa no item 5.2, sendo utilizada a técnica de pré-processamento de sinais por média neste trabalho de pesquina na fase de pré-processamento de sinais do reconhecimento de fala.

7.3.3 RNA

Neste trabalho de pesquisa utiliza-se o modelo de RNA, proposto por Tontini (1995), *RBF-Fuzzy Artmap*, que é composto por duas RNAs, que são: rede de Funções com

Base Radial (RBF) e rede Fuzzy-Artmap, treinando-a para o reconhecimento de fala. Este modelo de RNA, também já foi utilizado em Klabunde (1996) com aplicação no reconhecimento de fala e em Fischer (1999), que propôs um novo pré-processamento de sinal para reconhecimento de fala utilizando este modelo.

7.4 REQUISITOS DE SOFTWARE E HARDWARE

Loesch (1996) menciona que a tarefa de desenvolver um software para realizar experimentos com RNAs representa uma parte significativa da parcela de tempo na pesquisa.

Como SO foi utilizado Windows 98 SE (*Second Edition*) e quanto ao ambiente de desenvolvimento, utilizou-se Borland Delphi 5 com sua linguagem de programação, Object Pascal. Neste ambiente foram desenvolvidas as três etapas mencionadas no item 7.1 (ferramenta, interface e aplicação exemplo).

Para o desenvolvimento do protótipo, foi necessário hardware que possibilitasse a aquisição de sinais sonoros e a conversão dos mesmos para sinais digitais.

Quanto a plataforma de hardware, foi utilizado um computador padrão IBM-PC, com arquitetura Atlon e clock de 900 Mhz, com um microfone conectado a placa de som Sound Blaster 128 bits, padrão Windows para aquisição de sinal.

7.5 APRESENTAÇÃO DO PROTÓTIPO

Conforme o objetivo já citado no item 7.1, além das classes *Automation OLE* (segunda etapa), foi ainda desenvolvido uma ferramenta de treinamento de RNA (primeira etapa), e uma aplicação exemplo, demonstrando a utilização e benefícios do reconhecimento de fala (terceira etapa). Estas duas etapas serão melhor descritas no transcorrer deste item.

7.5.1 FERRAMENTA RNA TOOLS

As classes *Automation OLE* foram criadas para efetuar a interface entre a aplicação exemplo e a RNA, possibilitando a interface por reconhecimento de fala, mas

necessitam de informações para efetuar a aquisição de sinal, o pré-processamento e o processamento da RNA, que resultará no reconhecimento ou não de uma palavra. Estas informações (banco de conhecimento) são geradas na ferramenta de treinamento RNATools.

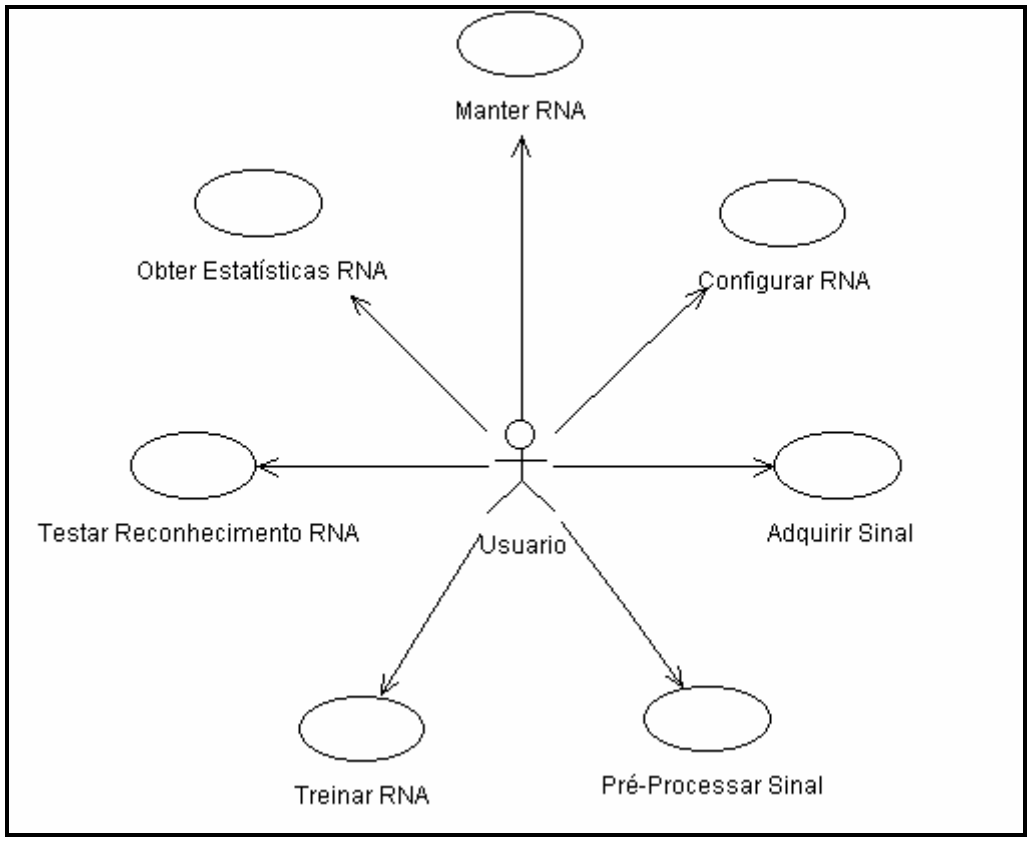
7.5.1.1 ESPECIFICAÇÃO

Para especificação será adotada a mesma metodologia descrita no item 7.2, e já utilizada na especificação das classes *Automation OLE*, ou seja, casos de uso, diagrama de classes e diagramas de seqüência.

7.5.1.1.1 DIAGRAMA DE CASO DE USO

A figura 7.7 ilustra o diagrama de caso de uso da ferramenta RNATools.

Figura 7.7 – Caso de uso da ferramenta RNATools



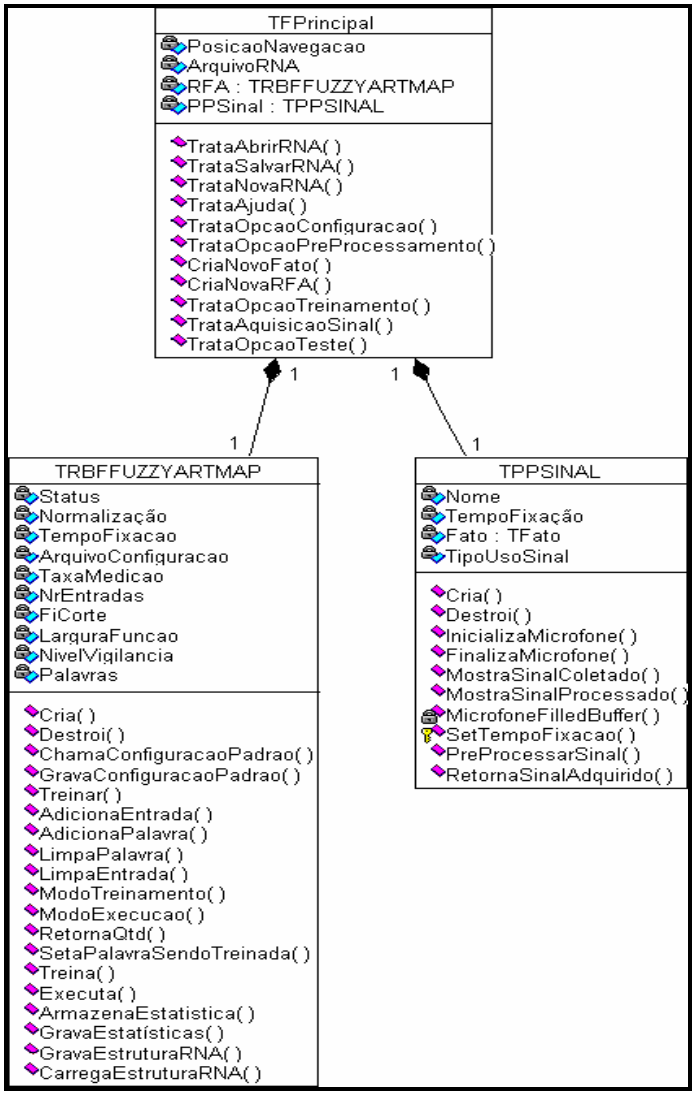
Analisou-se para a ferramenta RNATools sete casos de usos, que serão descritos a seguir:

a) “Manter RNA”: o usuário pode criar um novo projeto de RNA, salvar o projeto de RNA corrente ou abrir um projeto de RNA previamente criado;
b) “Configurar RNA”: o usuário deverá configurar os parâmetros da RNA, como: nome, valor de corte, nível de vigilância, número de entradas, tempo de fixação do sinal, taxa de medição, nível de normalização, etc. Nesta opção ainda, deve entrar com as palavras que deverão ser reconhecidas pela RNA. A ordem incluída será a ordem de representação final, por exemplo, se for incluída a palavra "UM" por primeiro, e "ZERO" após, serão consideradas pela RNA como sendo as palavras de índice 0 e 1 respectivamente, cabendo ao usuário fazer esta relação em sua aplicação;
c) “Adquirir Sinal”: o usuário irá informar a palavra que está sendo coletada, se será utilizada para teste ou treinamento, sendo ambos uma opção também aceita. Após esta ação executada, deverá fazer a locução através do microfone, e por meio da placa de som e APIs do Windows contidas nas classes de captação de sinal, o sinal será captado pela ferramenta;
d) “Pré-Processar Sinal”: o usuário deverá selecionar os sinais que deseja pré-processar e utilizar no treinamento da RNA, e através da execução desta etapa, obterá os valores de entrada para a RNA;
e) “Treinar RNA”: o usuário deverá selecionar os sinais que deseja treinar na RNA, os quais definiu como sinais de treinamento ou ambos na aquisição do sinal, sendo possíveis somente os pré-processados. Com os sinais já pré-processados será feito o treinamento da RNA;
f) “Testar Reconhecimento RNA”: nesta etapa do processo, o usuário pode efetuar testes de reconhecimento, adquirindo sinal, pré-processando-o e executando-o na RNA. Após estas etapas concluídas, pode ainda verificar se o reconhecimento foi ou não efetuado. Caso não tenha reconhecido o sinal, o usuário pode ainda treinar a RNA, com a finalidade de torná-la mais eficiente;
g) “Obter Estatísticas RNA”: o usuário pode obter estatísticas do número de apresentações das palavras na RNA, e de quantas vezes foram reconhecidas.

7.5.1.1.2 DIAGRAMA DE CLASSES

A classe TFPrincipal é a classe do formulário principal da ferramenta RNATools, responsável por disponibilizar a interface citada no item 7.5.1.2, e também manter o controle sobre as classes referentes a RNA RBF Fuzzy Artmap (TRBFFuzzyArtmap) e aquisição de sinal (TPPSinal). Sua principal função é a geração da base de conhecimento, que "fisicamente" é um arquivo contendo as informações de configuração (RNA e aquisição de sinal), palavras (lista de palavras a serem reconhecidas) e estrutura da RNA (camadas e neurônios). O diagrama de classes da ferramenta RNATools é demonstrado na figura 7.8, tendo as classes TRbfFuzzyArtmap e TPPSinal a estrutura de classes já demonstradas na figura 7.3 e 7.4 respectivamente, não necessitando a sua reapresentação neste diagrama de classes da figura 7.7.

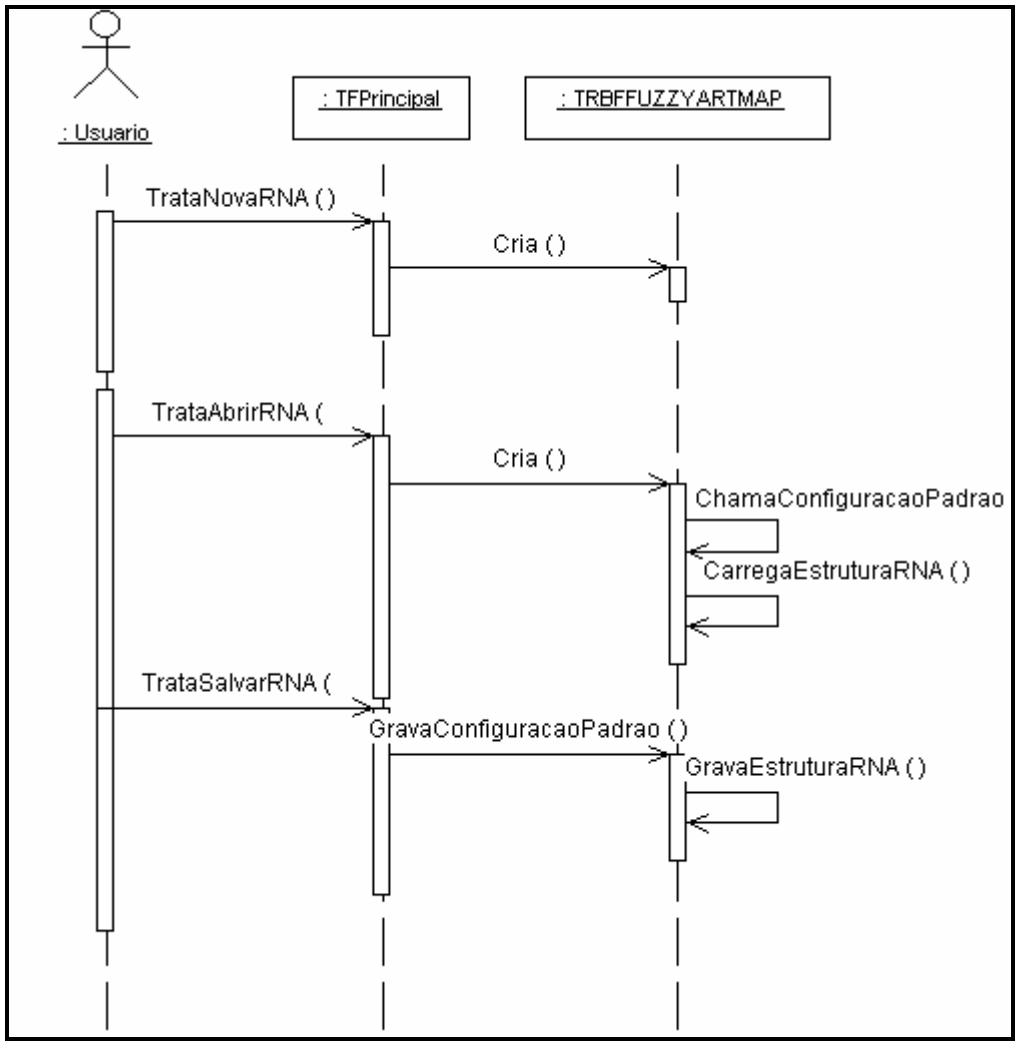
Figura 7.8 – Diagrama de classes da ferramenta RNATools



7.5.1.1.3 DIAGRAMAS DE SEQÜÊNCIA

7.5.1.1.3.1 DIAGRAMA DE SEQÜÊNCIA “MANTER RNA”

Figura 7.9 – Diagrama de seqüência “Manter RNA” da ferramenta RNATools



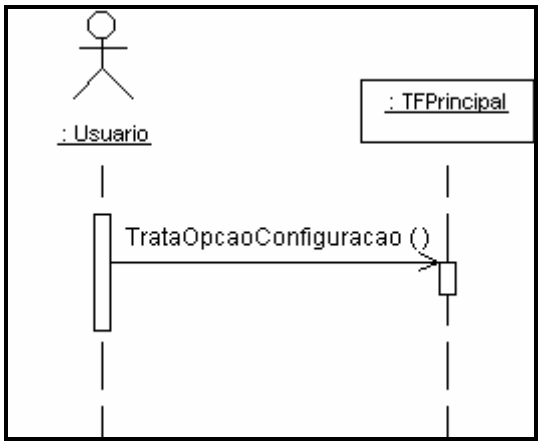
O caso de uso “Manter RNA” originou o diagrama de seqüência mostrado na figura 7.9, e pode-se observar a presença de três métodos:

a) “TrataNovaRNA”: possibilita ao usuário criar um novo projeto de RNA;
b) “TrataAbrirRNA”: o usuário poderá abrir um projeto já existente, que poderá já conter a estrutura de um treinamento efetuado anteriormente;
c) “TrataSalvarRNA”: para que não seja perdida informação a cada treinamento, e possibilite ao usuário efetuar vários treinamentos sem perder

os resultados anteriores, o usuário pode efetuar o salvamento do projeto atualmente em utilização na ferramenta.

7.5.1.1.3.2 DIAGRAMA DE SEQÜÊNCIA “CONFIGURAR RNA”

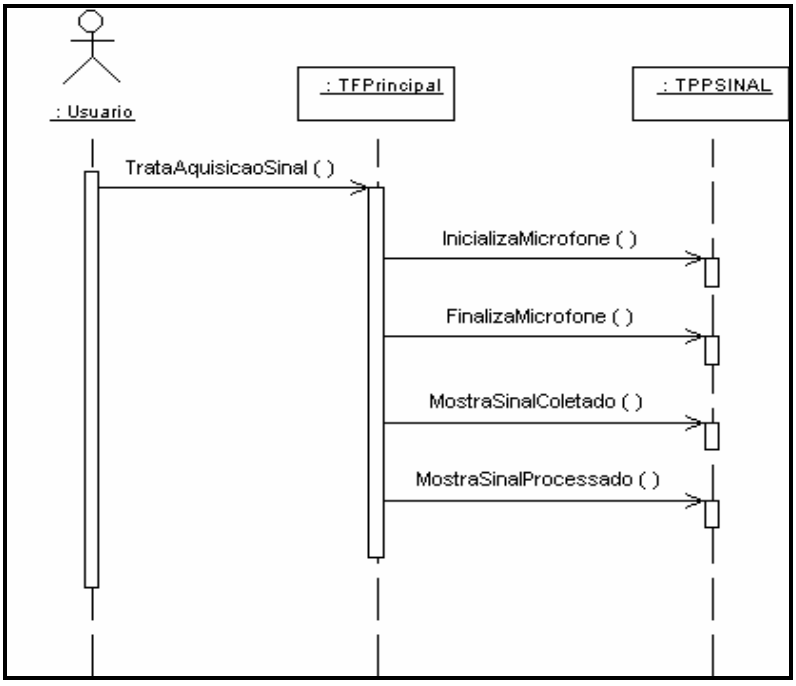
Figura 7.10 – Diagrama de seqüência “Configurar RNA” da ferramenta RNATools



O caso de uso “Configurar RNA” possui o diagrama de seqüência mostrado na figura 7.10, e nele observa-se a presença de apenas um método, “TrataOpcaoConfiguracao”, que possui a finalidade de possibilitar ao usuário configurar os parâmetros fixos para a RNA, e também palavras a serem reconhecidas.

7.5.1.1.3.3 DIAGRAMA DE SEQÜÊNCIA “ADQUIRIR SINAL”

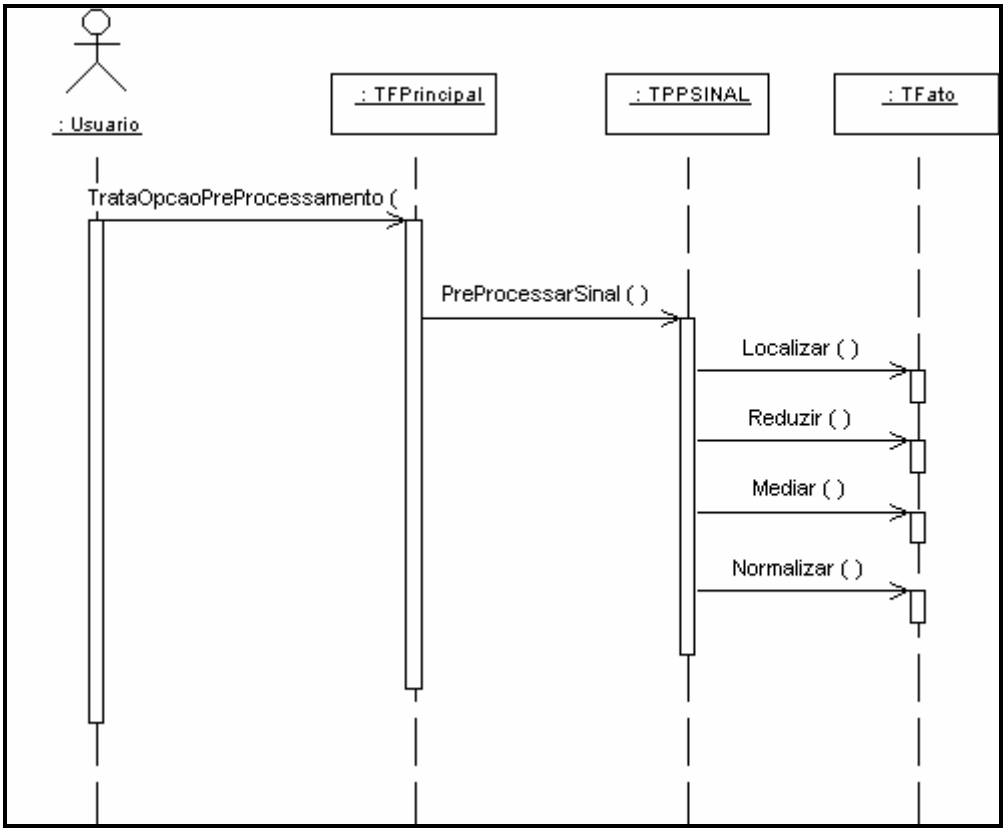
Figura 7.11 – Diagrama de seqüência “Adquirir Sinal” da ferramenta RNATools



Do caso de uso “Adquirir Sinal”, resultou o diagrama de seqüência mostrado na figura 7.11. Pode-se observar a existência do método “TrataAquisicaoSinal”, que aciona a classe TPPSinal (descrita no item 7.2.1.2.2) efetuando assim a aquisição de sinal pelo usuário.

7.5.1.1.3.4 DIAGRAMA DE SEQÜÊNCIA “PRÉ-PROCESSAR SINAL”

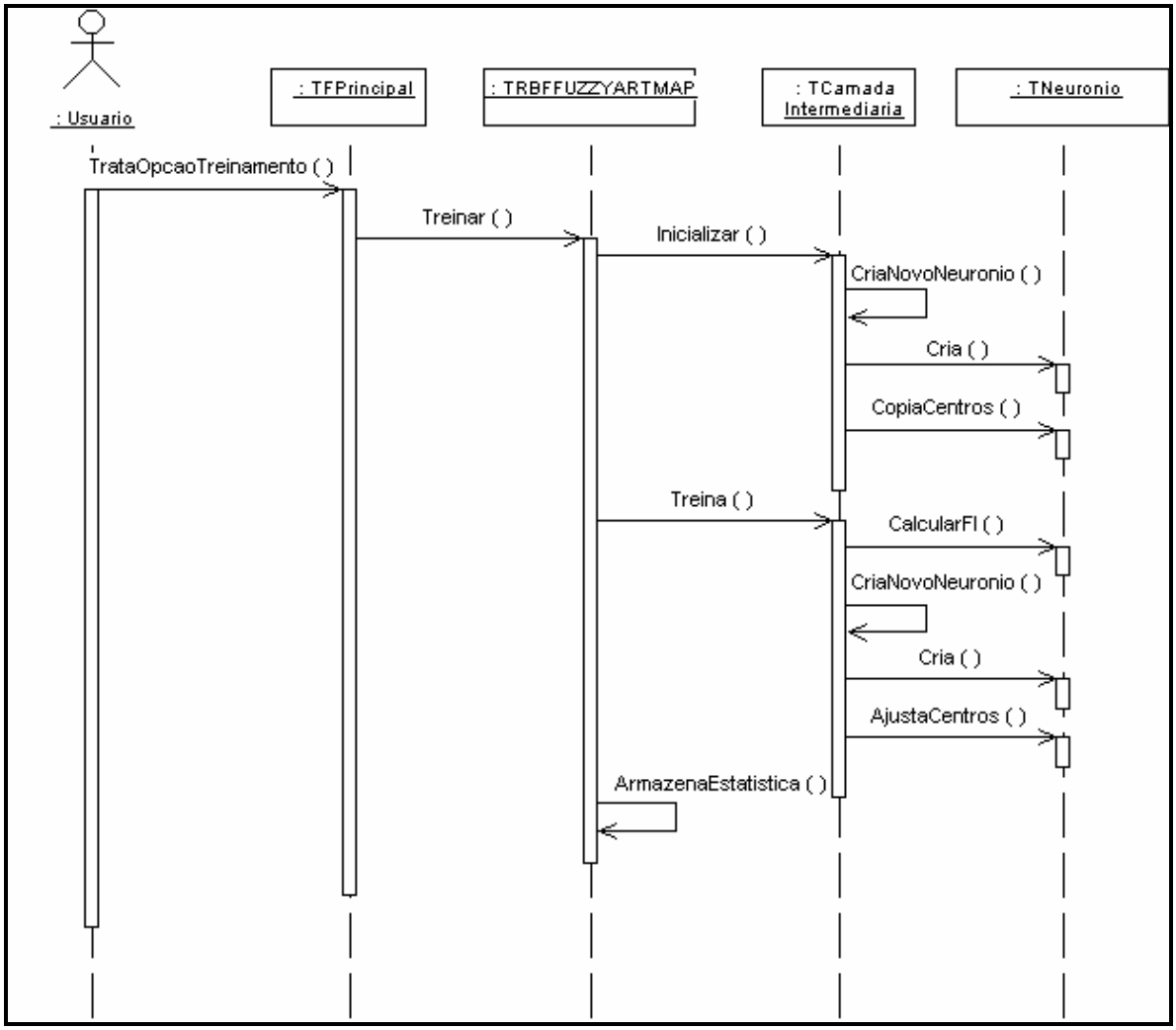
Figura 7.12 – Diagrama de seqüência “Pré-processar Sinal” da ferramenta RNATools



O caso de uso “Pré-processar Sinal” possui o diagrama de seqüência mostrado na figura 7.12. Pode-se observar a existência do método “TrataOpcaoPreProcessamento”, que aciona a classe TPPSinal (descrita no item 7.2.1.2.2) através do método “PreProcessarSinal”, e este efetua o pré-processamento seguindo as etapas citadas no item 5.2.1., utilizando-se também da classe TFato mencionada no diagrama de classes da figura 7.4.

7.5.1.1.3.5 DIAGRAMA DE SEQÜÊNCIA “TREINAR RNA”

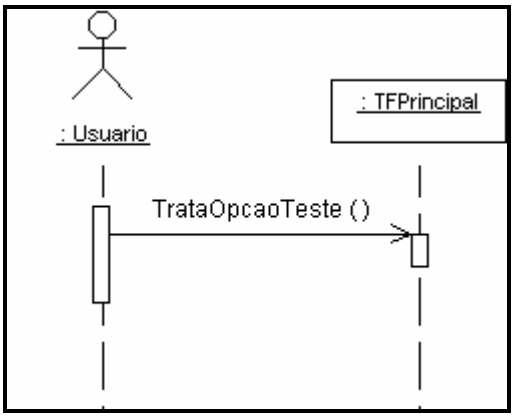
Figura 7.13 – Diagrama de seqüência “Treinar RNA” da ferramenta RNATools



O caso de uso “Treinar RNA” possui o diagrama de seqüência mostrado na figura 7.13, o qual pode-se observar a existência do método “TrataOpcaoTreinamento”, que aciona a classe TRBFFuzzyArtmap (descrita no item 7.2.1.2.1) através do método “Treinar”. O método “Treinar” da classe TRBFFuzzyArtmap verifica se a camada intermediária possui neurônios; caso não haja a presença de nenhum, inicializa a camada intermediária através do método “Inicializar”, caso contrário, efetua o treinamento através do método “Treina”.

7.5.1.1.3.6 DIAGRAMA DE SEQÜÊNCIA “TESTAR REC. RNA”

Figura 7.14 – Diagrama de seqüência “Testar Reconhecimento RNA” da RNATools



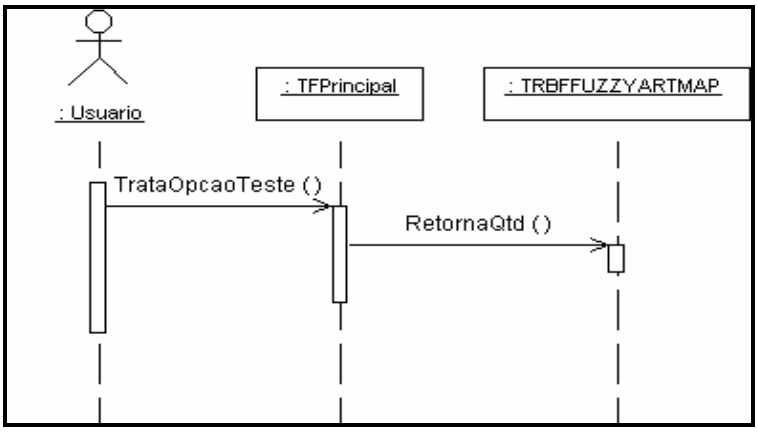
O caso de uso “Testar Reconhecimento RNA” possui o diagrama de seqüência mostrado na figura 7.14, e nele observa-se a presença de apenas um método, “TrataOpcaoTeste”, que possibilita ao usuário efetuar aquisição de sinal, ou utilizar um sinal previamente captado (mencionado pelo diagrama de seqüência 7.5.1.1.3.3), e efetuar a execução da RNA para verificação se a palavra é reconhecida ou não.

O usuário possui ainda a possibilidade de efetuar treinamento caso a palavra não tenha sido reconhecida, informando somente o padrão correto a ser reconhecido para treinamento. Este processo está descrito no item 7.2.1.3.2, onde em modo de treinamento, será feita a atualização dos centros caso o neurônio vencedor reconheça o padrão apresentado, ou, a criação de um novo neurônio para o reconhecimento deste padrão.

Neste diagrama de seqüência não se fez a descrição de todas as chamadas às classes em seqüência da apresentada, por se tratar de todo o processo que está sendo descrito neste capítulo.

7.5.1.1.3.7 DIAGRAMA SEQÜÊNCIA “OBTER ESTATÍSTICA RNA”

Figura 7.15 – Diagrama de seqüência “Obter Estatísticas RNA” da RNATools



O caso de uso “Obter Estatísticas RNA” resultou o diagrama de seqüência mostrado na figura 7.15, e nele pode-se observar a presença do método “TrataOpcaoTeste”, que aciona a classe TRBFFuzzyArtmap, que fornece através do método “RetornaQtd” informações referentes ao número de treinamentos, número de apresentações, e os acertos no reconhecimento das apresentações de padrões realizadas na RNA.

7.5.1.2 APRESENTAÇÃO

Para apresentar uma maneira simples e de fácil operação ao usuário, optou-se pela utilização de apenas uma janela onde a disposição das opções representam a seqüência das etapas para reconhecimento de fala citada no item 2.7. A tela principal da ferramenta de treinamento, intitulada de “RNATOOLS”, é demonstrada na figura 7.16.

Figura 7.16 – Tela principal da ferramenta de treinamento RNATOOLS

Inicialmente o usuário, caso já não tenha um projeto criado, opta por selecionar o menu “Arquivo”, e sua sub-opção “Nova RNA”. Um novo projeto será criado, após informar o local para armazenar o banco de conhecimento (citado no item 7.5.1).

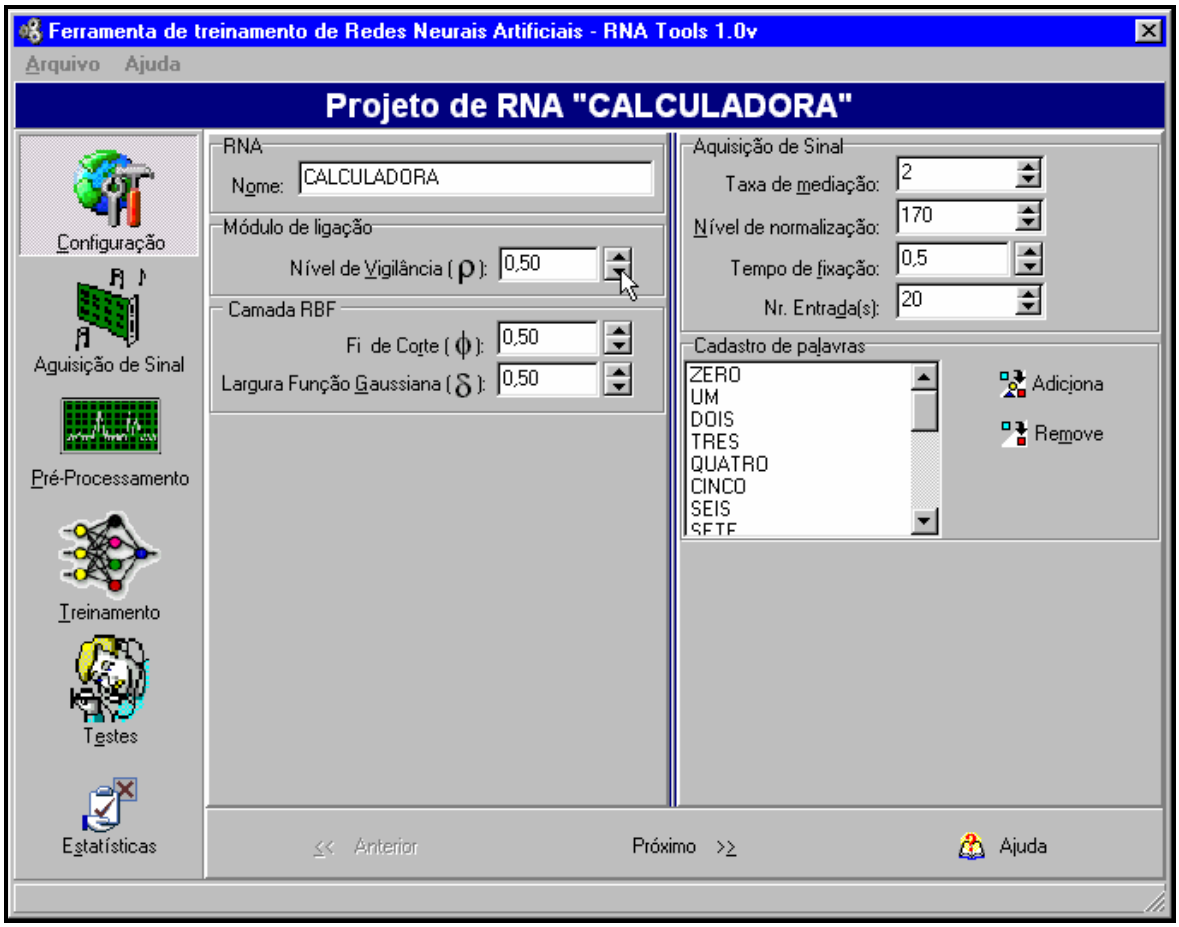
Caso já o possua, deverá localizá-lo onde o armazenou, e a RNA contida neste projeto será carregada para a ferramenta, através do menu “Arquivo”, sub-opção “Abrir RNA”. Restando a opção “Salvar RNA” deste mesmo menu, para efetuar o salvamento do projeto corrente da ferramenta.

Após criar um novo projeto de RNA, ou abrir um projeto existente, a interface possibilita ao usuário seguir as etapas do projeto da RNA através dos botões de navegação “Anterior” e “Próximo” criando assim uma interface do tipo *Wizard*, ou também conhecida por “Passo-a-passo”. Outro método de utilização é clicar sobre os botões com a opção desejada.

7.5.1.2.1 RNATOOLS – CONFIGURAÇÃO

A ferramenta de treinamento disponibiliza ao usuário a opção de parametrizar a RNA conforme as suas necessidades. Para efetuar esta entrada de dados, foi desenvolvida a janela ilustrada na figura 7.17.

Figura 7.17 – Opção de configuração da ferramenta RNATOOLS



Os parâmetros que podem ser alterados na RNA como nível de vigilância, fi de corte e largura da função gaussiana, já foram mencionados no capítulo 6 deste trabalho de pesquisa, sobre a RNA *RBF-Fuzzy Artmap*. O parâmetro “Nr. Entradas”, é utilizado tanto para a aquisição de sinal, pré-processamento e RNA, pois é a quantidade de elmentos do vetor de entrada para a RNA, proveniente do pré-processamento. Os demais parâmetros são referentes à aquisição de sinal e são:

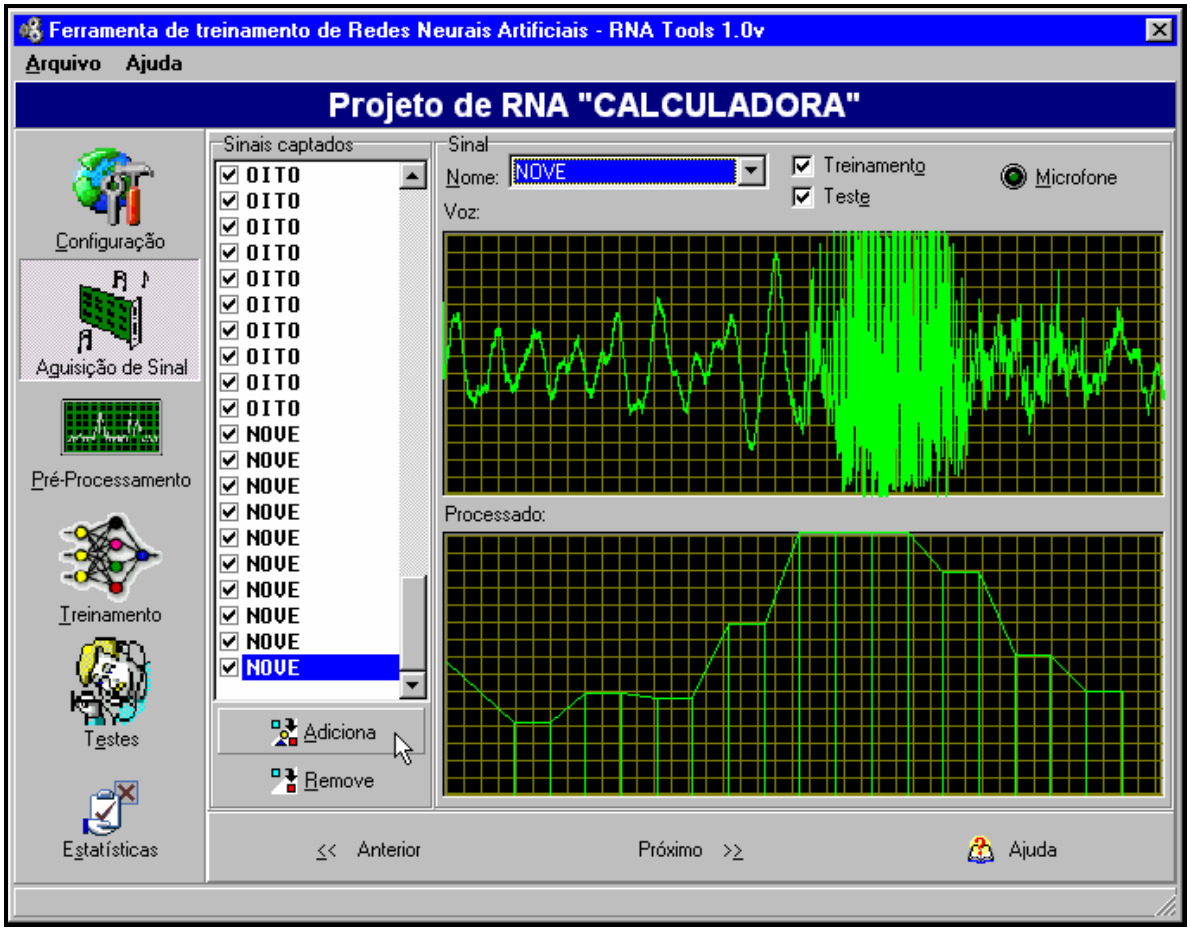
a) Taxa de mediação: estabelece a quantidade de sinais que serão agrupados prevalecendo o sinal de maior valor;
b) Nível de normalização: sabendo-se que o sinal adquirido possue um intervalo de 0 à 255, pode-se ampliar o sinal adquirido transferindo-se o limite superior para o valor informado, transformando-se assim os sinais maiores que o limite informado iguais ao limite;
c) Tempo de fixação: é o tempo em segundos para detecção do sinal e zona de silêncio.

Nesta opção de configuração, ainda é feito o cadastro das palavras que a RNA deverá ser treinada para reconhecer.

7.5.1.2.2 RNATOOLS – AQUISIÇÃO DE SINAL

A ferramenta de treinamento possibilita ao usuário adquirir o sinal, não tendo que recorrer a outros softwares para isso. Este recurso é disponibilizado através da opção “Adquirir Sinal” e esta opção está ilustrada na figura 7.18.

Figura 7.18 – Opção de aquisição de sinal da ferramenta RNATOOLS



Sua funcionalidade é feita através da execução dos seguintes passos:

a) inicializa-se o microfone através do botão “Microfone”, e após isto, o *led* contido neste botão acende-se, representando o microfone apto a adquirir sinal para a ferramenta;

b) o usuário efetua a locução do som, que representa a palavra a ser reconhecida;

c) finaliza-se o microfone através do mesmo botão “Microfone”, e após isto, o *led* contido neste botão apaga-se, representando a situação oposta do passo anterior. O sinal é mostrado nas caixas “Voz” e “Processado”, sendo “Voz” a caixa que contém o sinal bruto e “Processado” a caixa que contém o sinal já pré-processado;

d) seleciona-se na caixa “Nome” o sinal que foi adquirido e nas caixas de seleção “Treinamento” e “Teste” a finalidade deste sinal;

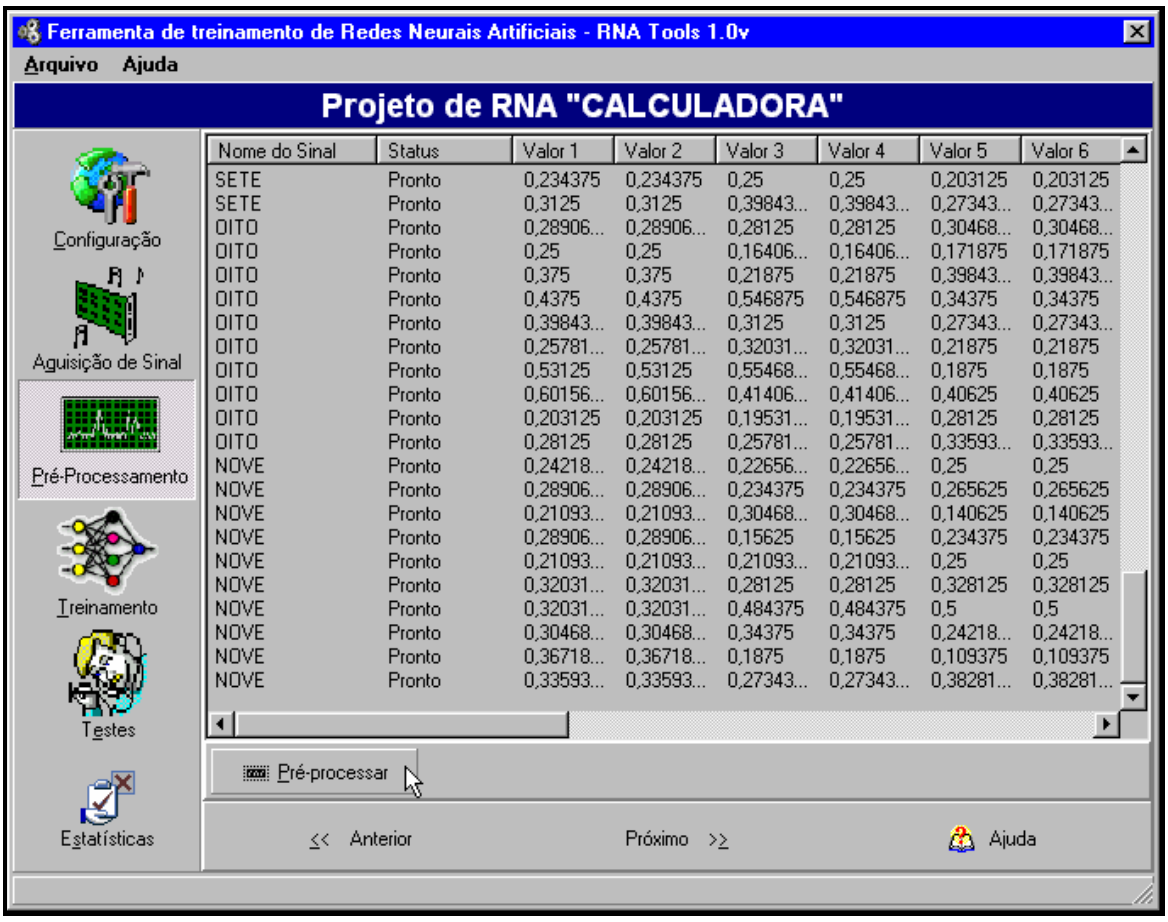
e) adiciona-se o sinal na lista, possibilitando antes ao usuário avaliar o sinal e verificar a presença de anomalias decorrentes de fatores externos ao processo de aquisição, tendo também a opção de remover um sinal já existente.

Após estes passos serem efetuados, o sinal estará disponível para pré-processamento, teste ou ambos. Este sinal também estará presente em outra ocasião que for aberto o projeto corrente caso o usuário salve o projeto. Lembra-se que os sinais captados não necessitam acompanhar a aplicação final e portanto foram armazenados em arquivo binário diferente da base de conhecimento.

7.5.1.2.3 RNATOOLS – PRÉ-PROCESSAMENTO

A opção "Pré-Processamento" possibilita ao usuário executar o pré-processamento da RNA. Somente após este passo estar realizado, os sinais poderão ser treinados na RNA. O pré-processamento é executado na ferramenta através do botão "Pré-processar" na janela ilustrada na figura 7.19.

Figura 7.19 – Opção de pré-processamento da ferramenta RNATOOLS

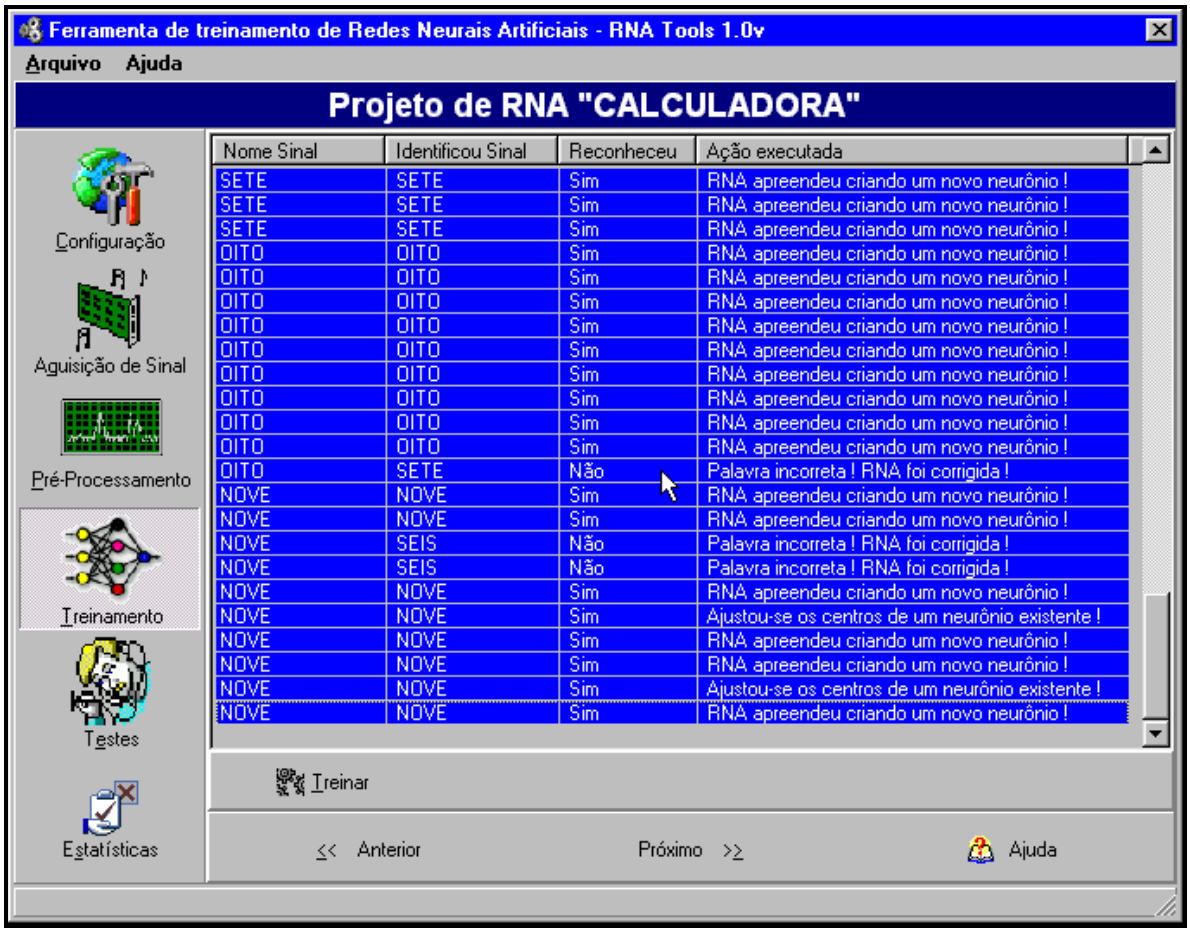


Somente os sinais captados com a finalidade de treinamento ou ambos, poderão fazer parte desta etapa, sendo ainda selecionados somente os que forem setados na caixa intitulada “Sinais captados” na opção descrita no item 7.5.1.2.2.

7.5.1.2.4 RNATOOLS – TREINAMENTO

A janela demonstrada na figura 7.20 possibilita ao usuário efetuar o treinamento da RNA.

Figura 7.20 – Opção de treinamento da ferramenta RNATOOLS



Na opção de treinamento, o usuário selecionará os sinais disponíveis para treinamento, que são os captados para esta finalidade e selecionados na lista de sinais. Após isto feito, clicará o botão “Treinar”, e a ferramenta fará o treinamento da RNA mostrando nesta mesma janela as ações executadas internamente pela RNA, se foi possível efetuar o reconhecimento e a identificação do sinal reconhecido.

Em modo de treinamento e na apresentação do primeiro padrão à RNA *RBF-Fuzzy Artmap*, será criado o primeiro neurônio associando a palavra correspondente ao padrão apresentado. A associação do padrão de entrada à palavra o qual representa já é feita na aquisição do sinal e pode ser verificada no item 7.5.1.2.2.

Apresentando-se os demais padrões de entrada para a RNA *RBF-Fuzzy Artmap*, as seguintes ações podem ser tomadas pela RNA *RBF-Fuzzy Artmap* implementada na ferramenta no modo de treinamento:

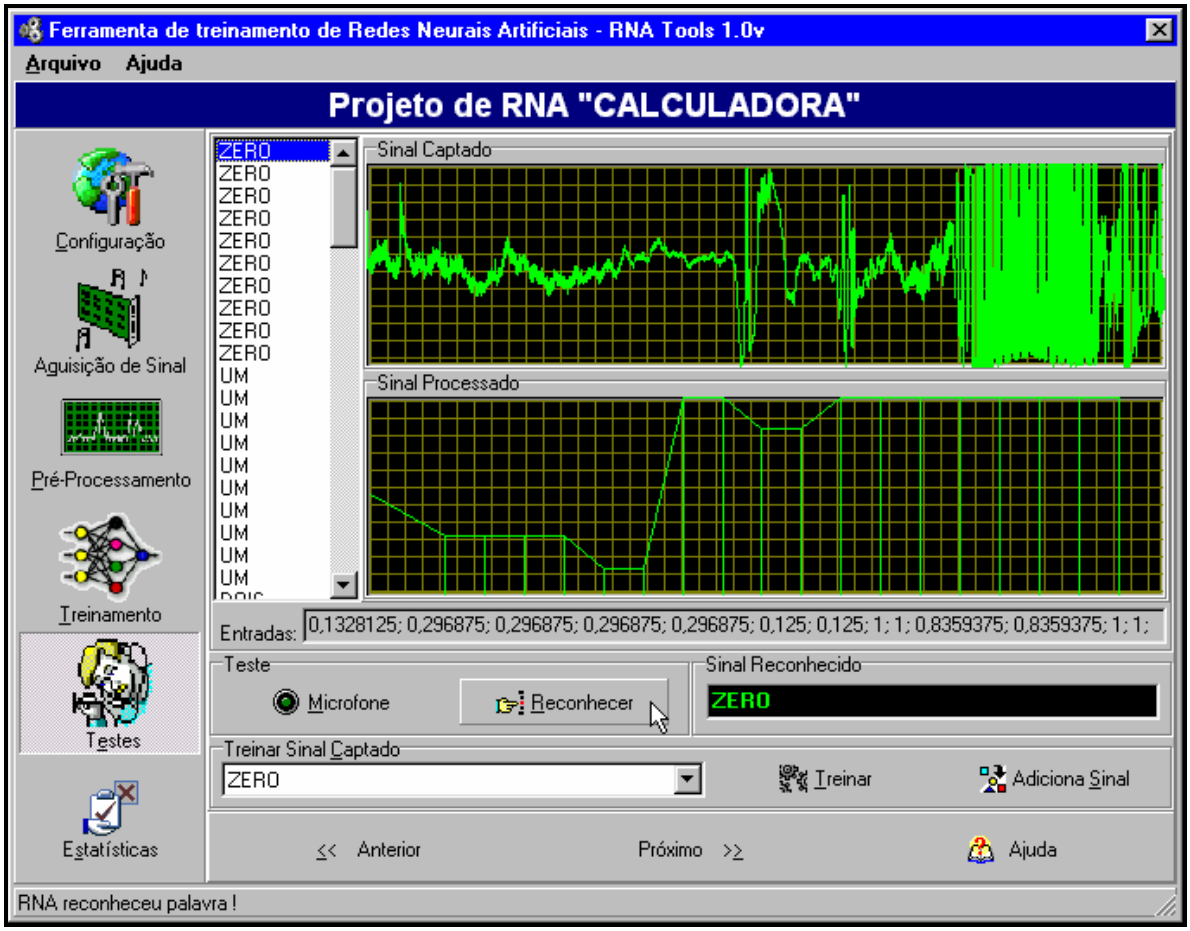
a) "RNA aprendeu criando um novo neurônio !": ocorre quando ao se analisar os índices de saída de cada neurônio já presente nas camadas da RNA, nenhum ultrapassou o nível de vigilância e associou corretamente o padrão apresentado a palavra. É então criado um novo neurônio para identificar o padrão apresentado;
b) "Palavra incorreta ! RNA foi corrigida !": ocorre quando ao se analisar os índices de saída dos neurônios já existentes, um deles que possuía o maior índice de saída e ultrapassou o nível de vigilância foi escolhido como vencedor, mas não representava a palavra apresentada. Neste caso, a ferramenta faz a correção automática da RNA, criando um novo neurônio para identificar aquele padrão;
c) "Ajustou-se os centros de um neurônio já existente": esta ação é executada quando ao analisar os índices de saída dos neurônios existentes, um deles obteve o maior índice de saída, ultrapassou o nível de vigilância e representava o sinal apresentado;
d) "Problemas ao processar informações. Padrão de entrada inválido !": ação proveniente de valores incorretos (menores que –1 e maiores que 1) estarem sendo treinados na RNA.

Estas ações podem ser observadas também na figura 7.20, na coluna "Ação executada".

7.5.1.2.5 RNATOOLS – TESTES

O recurso de testar o treinamento atual da RNA é disponibilizado ao usuário através da janela apresentada na figura 7.21.

Figura 7.21 – Opção de teste de reconhecimento da ferramenta RNATOOLS



O usuário tem a possibilidade de utilizar palavras já captadas na etapa de aquisição de sinal, contida na lista de sinais, ou utilizar sinais novos clicando no botão “Microfone”, que segue a mesma funcionalidade do item 7.5.1.2.2. O sinal escolhido é apresentado nas caixas “Sinal Captado” e “Sinal Processado”, e os valores do pré-processamento podem ser visualizados na caixa de texto “Entradas”.

Após a identificação do sinal que deseja utilizar, o usuário deverá informar qual o sinal que deseja reconhecer no quadro “Treinar Sinal Captado”, não tendo esta informação nenhuma relação com o resultado obtido com a execução da RNA. O usuário deverá entrar com esta informação, para que a RNA registre as estatísticas no reconhecimento do padrão e o número de vezes apresentado, que será visto no item 7.5.1.2.6.

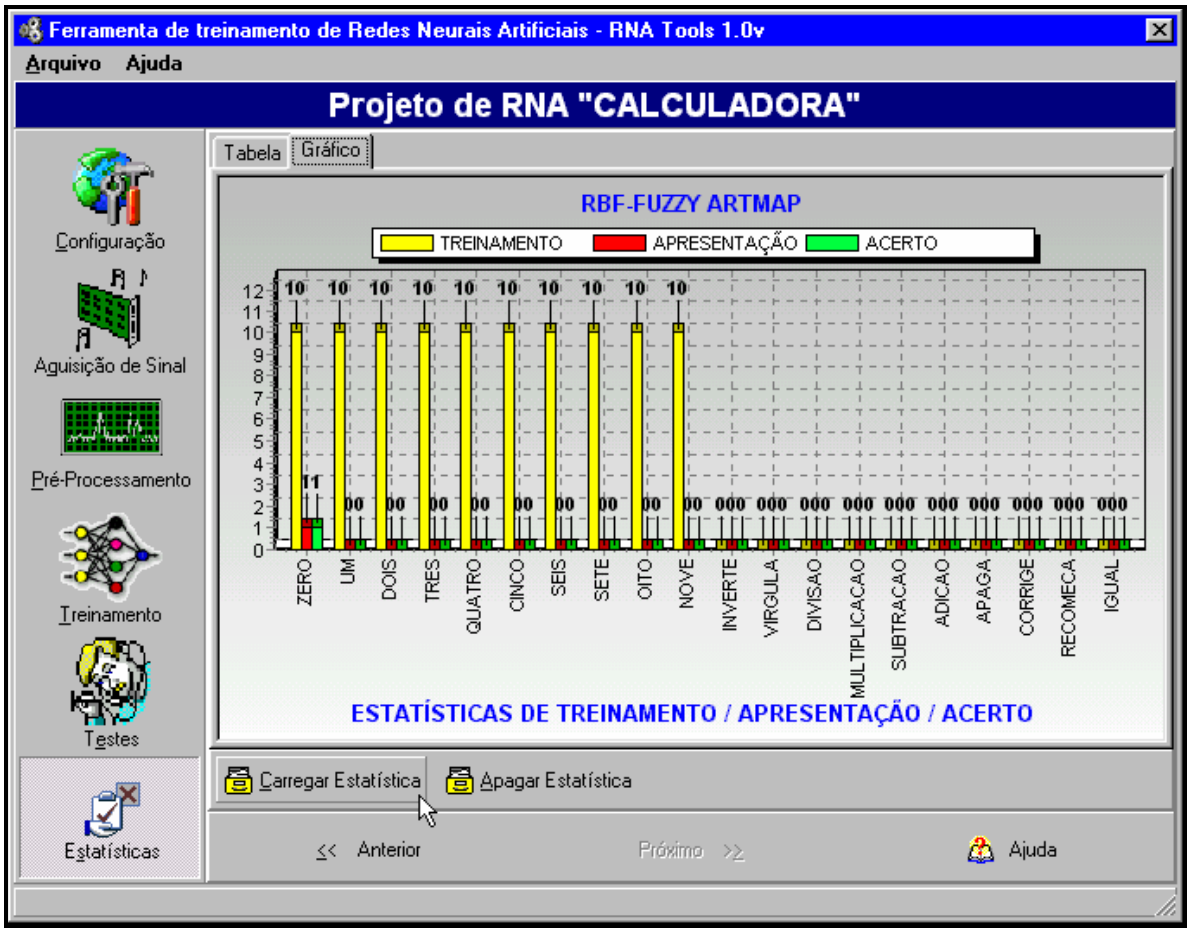
Tendo feito o procedimento anterior, o usuário deve clicar no botão “Reconhece” e a RNA entrará em modo de execução, verificando o reconhecimento deste

padrão. Caso reconheça, resultará na caixa “Sinal Reconhecido”, a palavra reconhecida; caso não, resultará nesta mesma caixa “Palavra Desconhecida !”. Em caso negativo, o usuário poderá treinar a RNA para reconhecer este padrão, no botão “Treinar” e também, se desejar, adicioná-lo na lista de sinais para futuros treinamentos, testes ou ambos.

7.5.1.2.6 RNATOOLS – ESTATÍSTICAS

Através da experiência adquirida no desenvolvimento do aplicativo citado no item 7.5.2 e de uma das etapas do reconhecimento de fala ser a observação dos resultados (item 2.7), observou-se a necessidade de obter estatísticas do treinamento e reconhecimento das palavras a serem reconhecidas. Por meio da opção “Estatísticas” presente na ferramenta o usuário pode verificar estes resultados à medida que vai efetuando o treinamento e teste de seu projeto de RNA. Esta opção está ilustrada na figura 7.22.

Figura 7.22 – Opção de estatísticas do reconhecimento da ferramenta RNATOOLS



Para visualizar as estatísticas atualmente armazenadas, o usuário deverá clicar no botão “Carregar Estatísticas”, e será apresentado os resultados obtidos através de uma

lista, ou também, graficamente. Tem ainda a opção de eliminar os valores armazenados para iniciar novamente o registro de resultados.

7.5.2 APLICAÇÃO EXEMPLO

Conforme citado no item 7.1, foi desenvolvido uma aplicação exemplo, que tem a finalidade de simular uma situação corriqueira de um desenvolvedor de software. Neste aplicativo são utilizadas as classes *Automation OLE* para possibilitar a interface por reconhecimento de fala.

Tendo este trabalho de pesquisa a proposta de reconhecer os números de 0 (zero) a 9 (nove), optou-se por desenvolver um aplicativo que já contivesse ações referentes a estes números, uma calculadora.

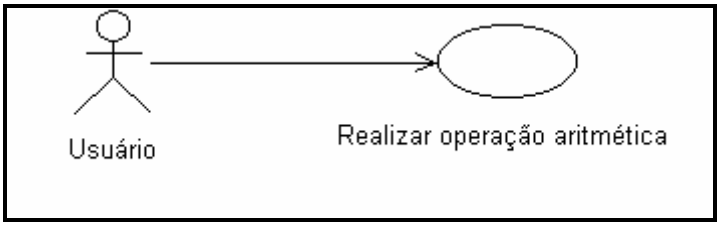
7.5.2.1 ESPECIFICAÇÃO

A especificação da aplicação exemplo segue a já adotada nos item anteriores (7.2 e 7.5.1.1).

7.5.2.1.1 DIAGRAMA DE CASO DE USO

Pode-se observar o caso de uso identificado para a aplicação exemplo na figura 7.23.

Figura 7.23 – Caso de Uso da aplicação exemplo



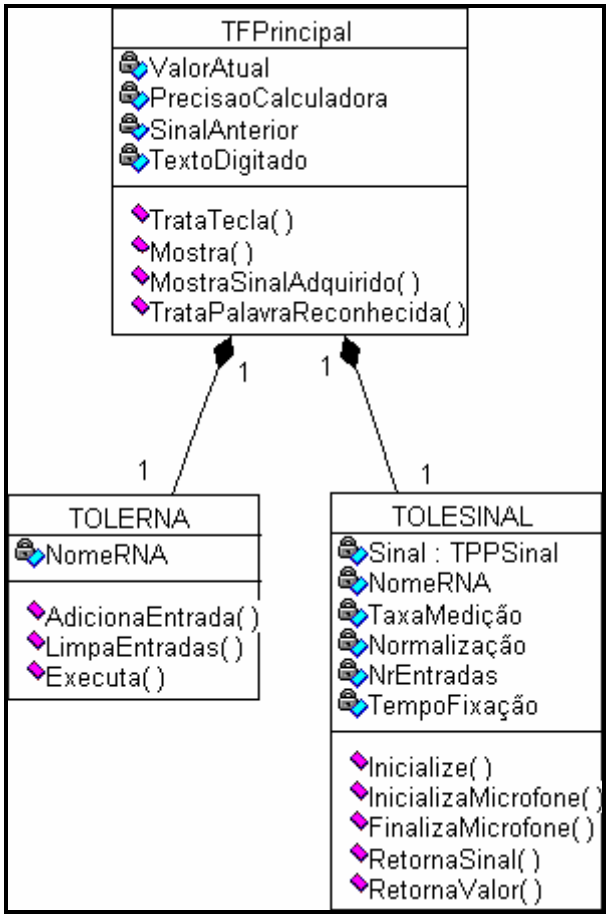
Para a aplicação exemplo, observou-se apenas um caso de uso onde usuário efetuará a entrada do primeiro valor, da operação desejada e do segundo valor, sendo mostrado o resultado da operação ao pressionar a próxima operação, ou sinal de igual.

Esta entrada pode ocorrer através do clique do mouse, através do teclado, através do teclado numérico ou por comandos definidos de fala.

7.5.2.1.2 DIAGRAMA DE CLASSES

O diagrama de classes da aplicação exemplo é demonstrado na figura 7.24.

Figura 7.24 – Diagrama de classes da aplicação exemplo



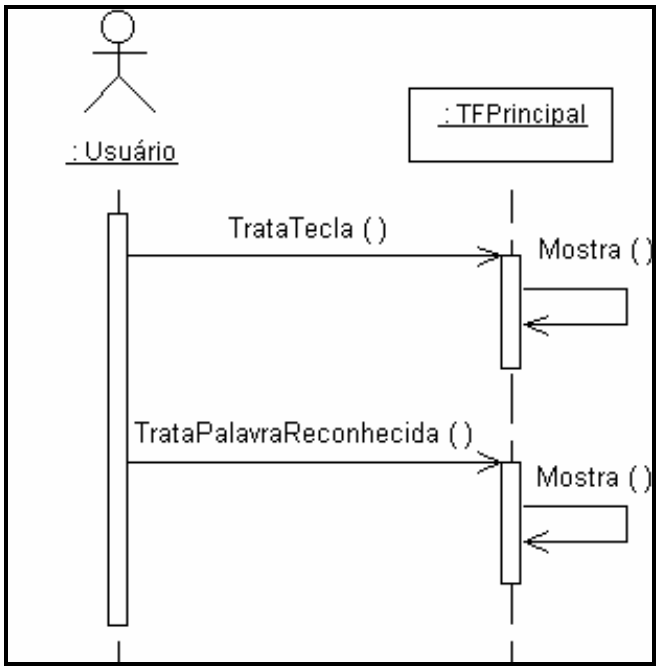
No diagrama de classes da aplicação exemplo nota-se a presença de três classes que são:

a) TFPrincipal: é a classe do formulário principal e único da aplicação exemplo, contendo métodos para tratamento das entradas do usuário pelo clique do mouse, através do teclado, através do teclado numérico ou por comandos definidos de fala;
b) TRNAOLE: mencionada no item 7.2.1.2.2;
c) TSINALOLE: mencionada no item 7.2.1.2.1.

7.5.2.1.3 DIAGRAMA DE SEQÜENCIA

O diagrama de seqüência resultante do caso de uso “Realizar operação aritmética”, que possui dois métodos, sendo o primeiro “TrataTecla” responsável pelas ações provindas de comandos via teclado, teclado numérico e mouse; e o segundo “TrataPalavraReconhecida”, para comandos de fala. Ele é demonstrado na figura 7.25.

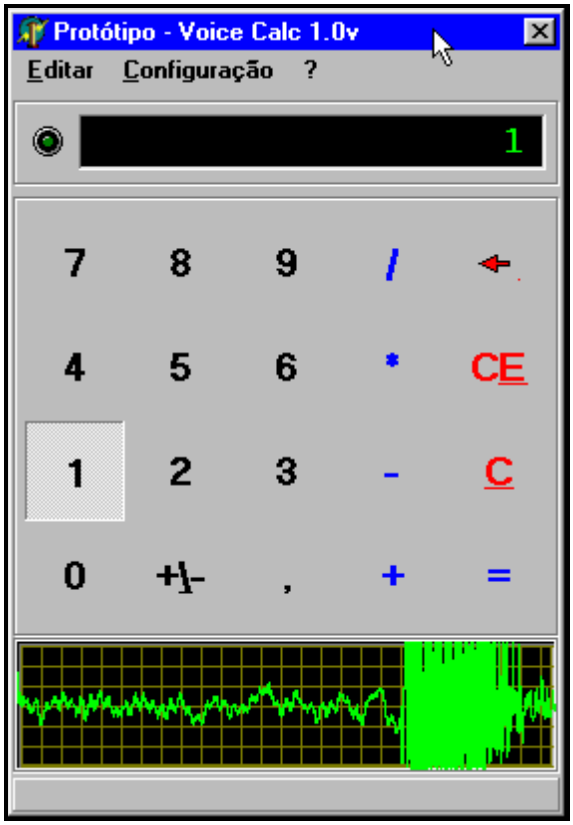
Figura 7.25 – Diagrama de seqüência da aplicação exemplo



7.5.2.2 APRESENTAÇÃO

A tela da aplicação exemplo é demonstrada na figura 7.26.

Figura 7.26 – Tela da aplicação exemplo



A funcionalidade da calculadora restringe-se ao usuário efetuar a entrada de um valor, a operação desejada e o segundo valor, sendo mostrado o resultado da operação ao pressionar a próxima operação, ou sinal de igual. Pode optar ainda por mais operações, corrigir a entrada de valor atual e iniciar o processo novamente.

Além dos dígitos de 0 a 9 a calculadora possui ainda outras operações que são acionadas pelo pressionamento das teclas no teclado e teclado numérico, cliques de mouse sobre os botões e pela pronúncia das seguintes palavras:

a) “inverte”: operação para mudar o sinal do valor digitado atualmente. No protótipo pode ser acionada pela tecla “\” ou pelo clique no botão “+/-”;
b) “vírgula”: operação para adicionar valores decimais ao valor atualmente sendo digitado. Pode ser acionada no protótipo pela tecla “,” ou pelo clique no botão “,”;
c) “divisão”: operação que selecionará a operação aritmética da divisão. Pode ser acionada no protótipo pela tecla “/” ou pelo clique no botão “/”;

d) “multiplicação”: operação que selecionará a operação aritmética da multiplicação. Pode ser acionada no protótipo pela tecla “*” ou pelo clique no botão “*”;
e) “subtração”: operação que selecionará a operação aritmética da subtração. Pode ser acionada no protótipo pela tecla “-” ou pelo clique no botão “-”;
f) “adição”: operação que selecionará a operação aritmética da adição. Pode ser acionada no protótipo pela tecla “+” ou pelo clique no botão “+”;
g) “apaga”: operação que irá apagar o último dígito digitado. Pode ser acionada no protótipo pela tecla “Backspace” ou pelo clique no botão “”;
h) “corrige”: operação que irá possibilitar realizar novamente a entrada do último valor informado. Pode ser acionada no protótipo pela tecla “E” ou pelo clique no botão “CE”;
i) “recomeça”: operação que irá iniciar novamente a utilização da calculadora, eliminando qualquer entrada ou operação feita. Pode ser acionada no protótipo pela tecla “C” ou pelo clique no botão “C”;
j) “igual”: retornará o valor das operações já realizadas, caso não havendo, não executará nenhuma função. Acionada no protótipo pela tecla “=” ou pelo clique no botão “=”.

O usuário tem as opções de menu “Editar” e “Configuração”, onde a primeira possibilita a troca de informações com outros aplicativos, através do recurso presente no Windows, o clipboard, também conhecido por área de transferência.

7.5.3 CLASSES AUTOMATION OLE

As classes Automation OLE desenvolvidas neste trabalho de pesquisa não possuem interface e foram criadas como sendo servidores *in-process* (item 7.4.1), lembrando ainda, que são disponibilizadas como bibliotecas com extensão DLL.

Nos itens seguintes será demonstrada a utilização das classes *Automation OLE* no ambiente de desenvolvimento Delphi. Lembra-se que nesta demonstração será apresentado o uso das classes *Automation OLE* como se o usuário necessitasse utilizá-las não possuindo o código fonte das mesmas, porque caso o tenha, é possível eliminar várias

etapas do processo demonstrado a seguir. Como são duas classes *Automation OLE*, basta repetir o processo para a classe *Automation OLE* TSinalOLE.

7.5.3.1 UTILIZANDO CLASSES AUTOMATION OLE EM DELPHI

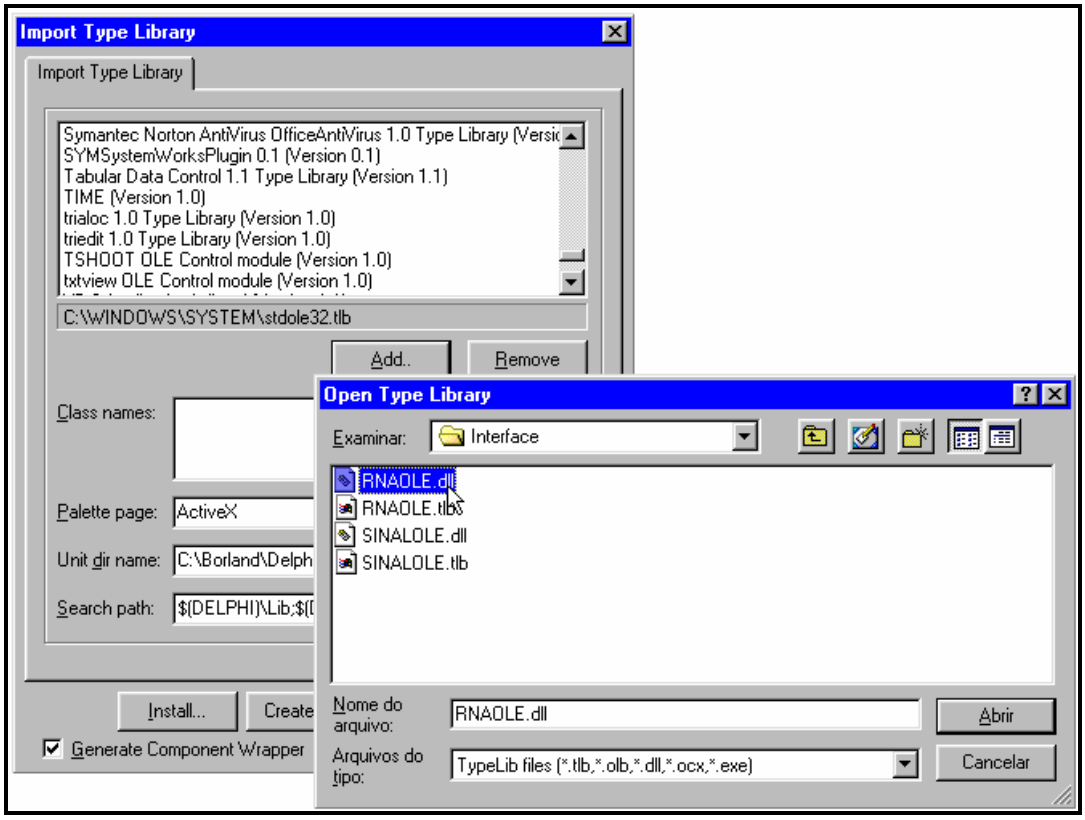
As classes *Automation OLE* em Delphi recebem um tratamento especial segundo Cantú (2000). Elas podem ser "empacotadas" criando um componente em torno da classe *Automation OLE*. O quadro 7.1 demonstra o código fonte do objeto TRnaOLE gerado da importação classe *Automation OLE* TRnaOLE.

Quadro 7.1 - Declaração da TRnaOLE como componente

```
TRNAOLE = class(TOleServer)
 private
      FIntf:       IRNAOLE;
{$IFDEF LIVE_SERVER_AT_DESIGN_TIME}
      FProps:      TRNAOLEProperties;
      function     GetServerProperties: TRNAOLEProperties;
{$ENDIF}
   function     GetDefaultInterface: IRNAOLE;
 protected
      procedure InitServerData; override;
      procedure InvokeEvent(DispID: TDispID; var Params: TVariantArray); override;
      function  Get_RNA: WideString;
      procedure Set_RNA(const Value: WideString);
      function  Get_Executa: Integer;
 public
      constructor Create(AOwner: TComponent); override;
      destructor  Destroy; override;
      procedure Connect; override;
      procedure ConnectTo(svrIntf: IRNAOLE);
      procedure Disconnect; override;
      procedure AdicionaEntrada(Valor: Double);
      procedure LimpaEntrada;
      property  DefaultInterface: IRNAOLE read GetDefaultInterface;
      property Executa: Integer read Get_Executa;
      property RNA: WideString read Get_RNA write Set_RNA;
 published
{$IFDEF LIVE_SERVER_AT_DESIGN_TIME}
      property Server: TRNAOLEProperties read GetServerProperties;
{$ENDIF}
end;
```

A importação da classe *Automation OLE* ocorre através do menu "Project", sub-opção "Import Type Library" demonstrado na figura 7.27, que cria o código da classe Automation OLE como um componente derivado da classe TOleServer (Borland, 1999), como visto no quadro 7.1, com uma função *Register* demonstrada no quadro 7.2.

Figura 7.27 – Opção Import Type Library do Delphi

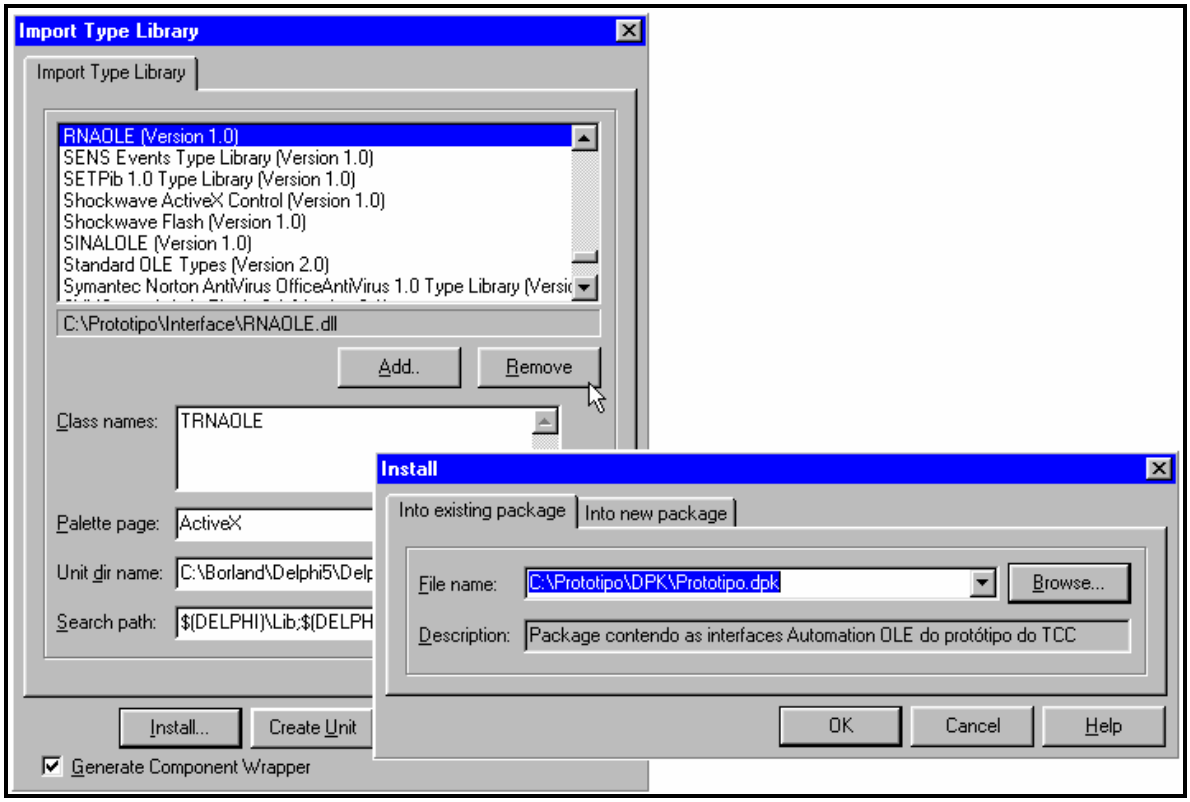


Quadro 7.2 – Código da função "Register" gerado pelo Delphi

```
procedure Register;
begin
  RegisterComponents('TCC',[TRNAOLE]);
end;
```

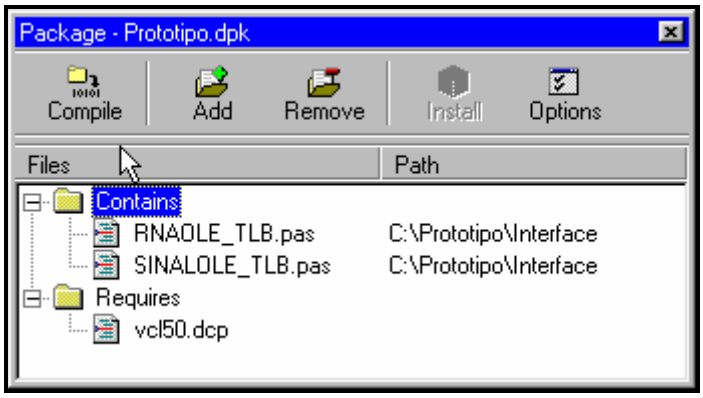
Após selecionar o arquivo com extensão DLL da biblioteca, o Delphi o adicionará na caixa "Servers". Sendo necessário neste ponto somente selecioná-lo na mesma caixa "Servers" e clicar no botão "Install..." e aparecerá a janela demonstrada na figura 7.28**.**

Figura 7.28 – Janela de instalação da classe Automation OLE como componente



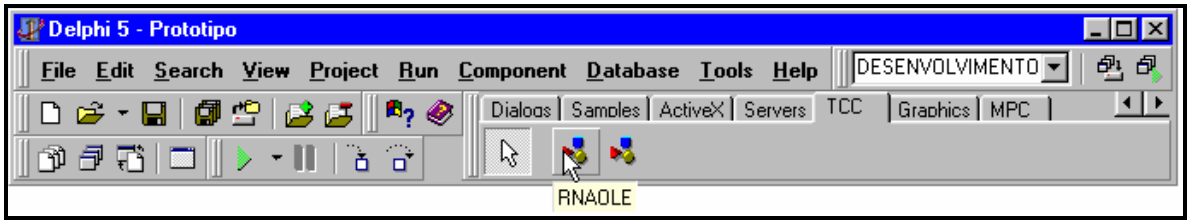
Clicando-se no botão “OK”, após definir as configurações conforme suas necessidades, o usuário verá a janela intitulada “Package” com os arquivos fonte da *Type Library* como sendo componentes comuns no pacote Delphi, demonstrado na figura 7.29.

Figura 7.29 – Pacote de componentes com as classe Automation OLE



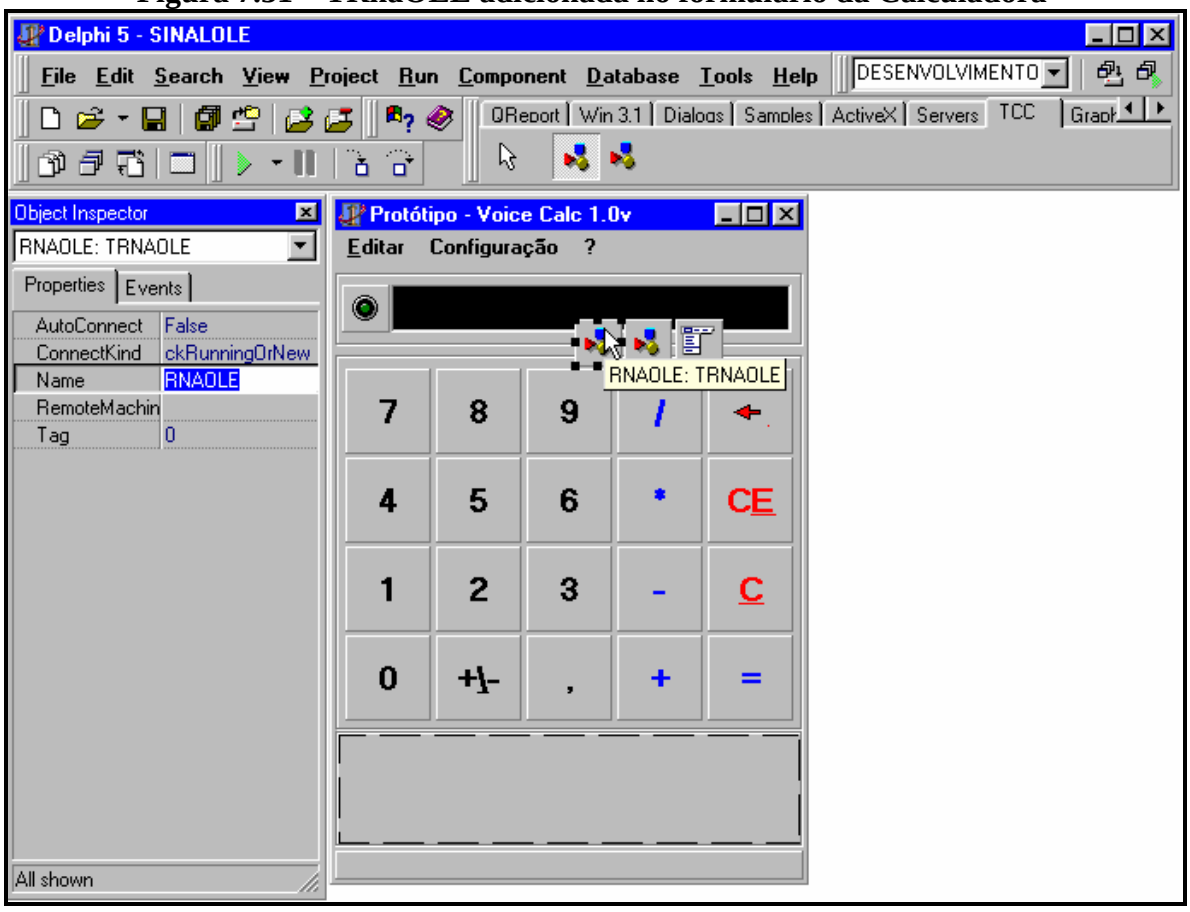
Efetuando-se a compilação e linkedição do pacote através da opção “Install” da janela demonstrada na figura 7.29, e caso estas ações não contenham erros, as classes Automation OLE serão adicionadas na paleta de componentes do Delphi, e ficando seu uso com o mesmo grau de dificuldade que um objeto disponibilizado pela VCL (mencionado no item 7.5) padrão do Delphi, demonstrado na figura 7.30.

Figura 7.30 – Classes Automation OLE na paleta de componentes do Delphi



Inclusa na paleta de componentes do Delphi, basta adicioná-la na aplicação como um componente comum, clicando-se sobre o componente na palete de componentes, e após no formulário da aplicação conforme figura 7.31. Lembra-se que estas etapas são feitas uma única vez, ou seja, uma vez incorporadas ao ambiente Delphi, podem ser utilizadas “n” vezes.

Figura 7.31 – TRnaOLE adicionada no formulário da Calculadora



Após esta etapa que torna a classe apta a ser utilizada no ambiente Delphi, o usuário, poderá adicionar os códigos seguintes em seu projeto para utilizá-las (TRnaOLE e TSinalOLE):

a) inicialização: o usuário deverá informar a classe TRnaOLE o projeto de RNA o qual deseja relacionar à aplicação que terá interface por fala. Esta operação pode ser feita através da propriedade da classe TRnaOLE “NomeRNA” e é exemplificada no quadro 7.3;

Quadro 7.3 – Exemplo de inicialização classe TRnaOLE

```
RNAOLE.NomeRNA:=ExtractFilePath(Application.ExeName)+
                'Calculadora.RNA';
```

c) início da captação do sinal: o usuário deve utilizar o método da classe TSinalOLE para efetuar a captação do sinal, intitulado “InicializaMicrofone”, conforme exemplo de código contido no quadro 7.4.

Quadro 7.4 – Exemplo de código para início de captação do sinal

```
SinalOle.InicializaMicrofone;
```

d) finalização da captação do sinal: o usuário deve utilizar o método da classe TSinalOLE para efetuar o término da captação do sinal, intitulado “FinalizaMicrofone”, conforme exemplo de código contido no quadro 7.5.

Quadro 7.5 – Exemplo de código para finalização captação do sinal

```
SinalOle.FinalizaMicrofone;
```

e) reconhecimento: para verificar se um determinado sinal é reconhecido como uma palavra de ação do usuário da aplicação do desenvolvedor, o desenvolvedor deverá incluir como exemplo, o código contido no quadro 7.6, que elimina as entradas atuais da classe TRnaOLE (método “LimpaEntrada”), informa a TRnaOLE o novo padrão de entrada (através do método “AdicionaEntrada”), e executa-o retornando o índice da palavra reconhecida (método “Executa”).

Quadro 7.6 – Código exemplo de reconhecimento pela classe TRnaOLE

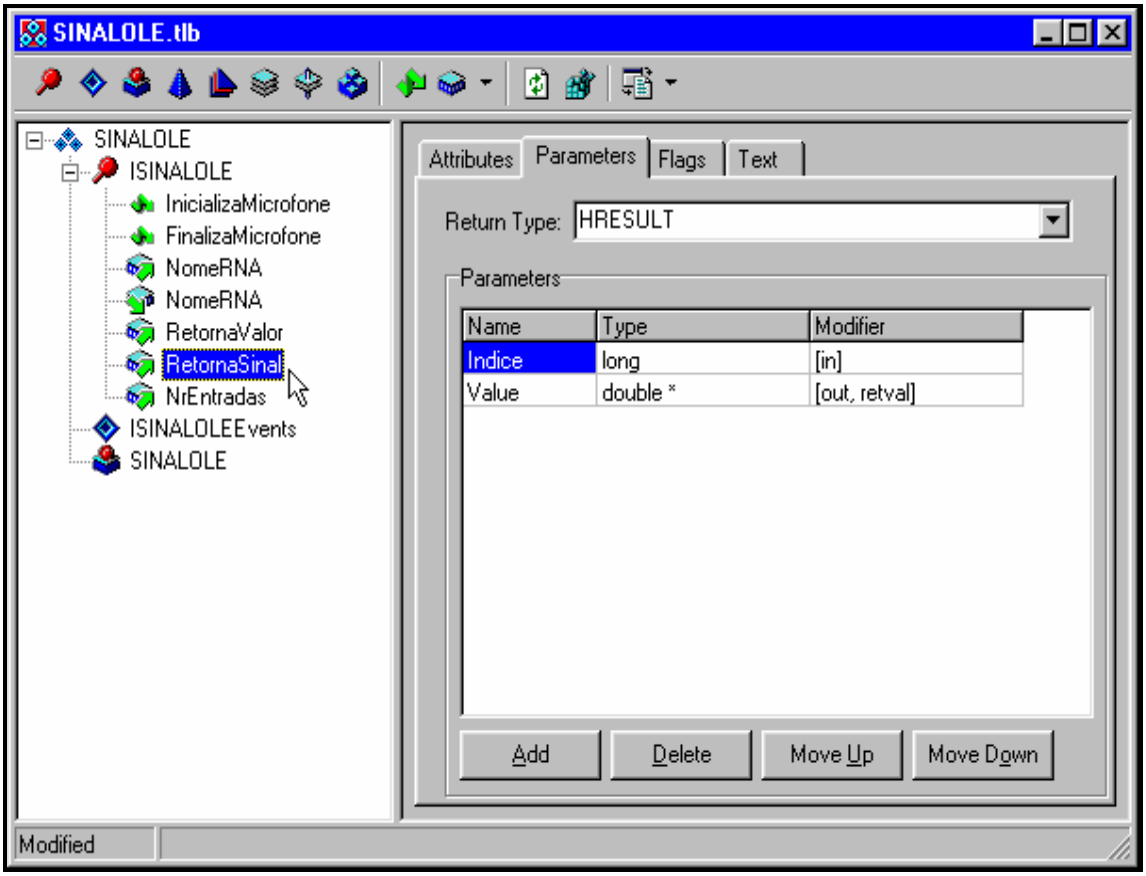
```
RNAOLE.LimpaEntrada;

For x:=0 to SinalOle.NrEntradas-1 do
     RNAOLE.AdicionaEntrada(SinalOle.RetornaValor[x]);

If  RNAOLE.Executa = -1 then begin //Palavra não reconhecida
   // Código para tratamento caso palavra não reconhecida
End
Else begin
  // Código para tratamento da palavra reconhecida
End;
```

7.5.3.2 REFERÊNCIA RÁPIDA DAS CLASSES AUTOMATION OLE

7.5.3.2.1 TSINALOLE

A *Type Library* da classe TSinalOLE está demonstrada na figura 7.32.

Figura 7.32 – Type Library da Classe Automation OLE TSinalOLE



A classe TSinalOLE apresenta as seguintes propriedades:

a) “NomeRNA”: a esta propriedade deve ser informado o arquivo de projeto da RNA criado no treinamento efetuado na ferramenta. É uma propriedade para leitura e gravação de valores do tipo *string*;
b) “NrEntradas”: contém o número de elementos do conjunto que forma o padrão de entrada para a RNA. Somente é possível ler o valor contido nesta propriedade. Seu valor é proveniente do arquivo de projeto da RNA e retorna um valor do tipo *integer*.

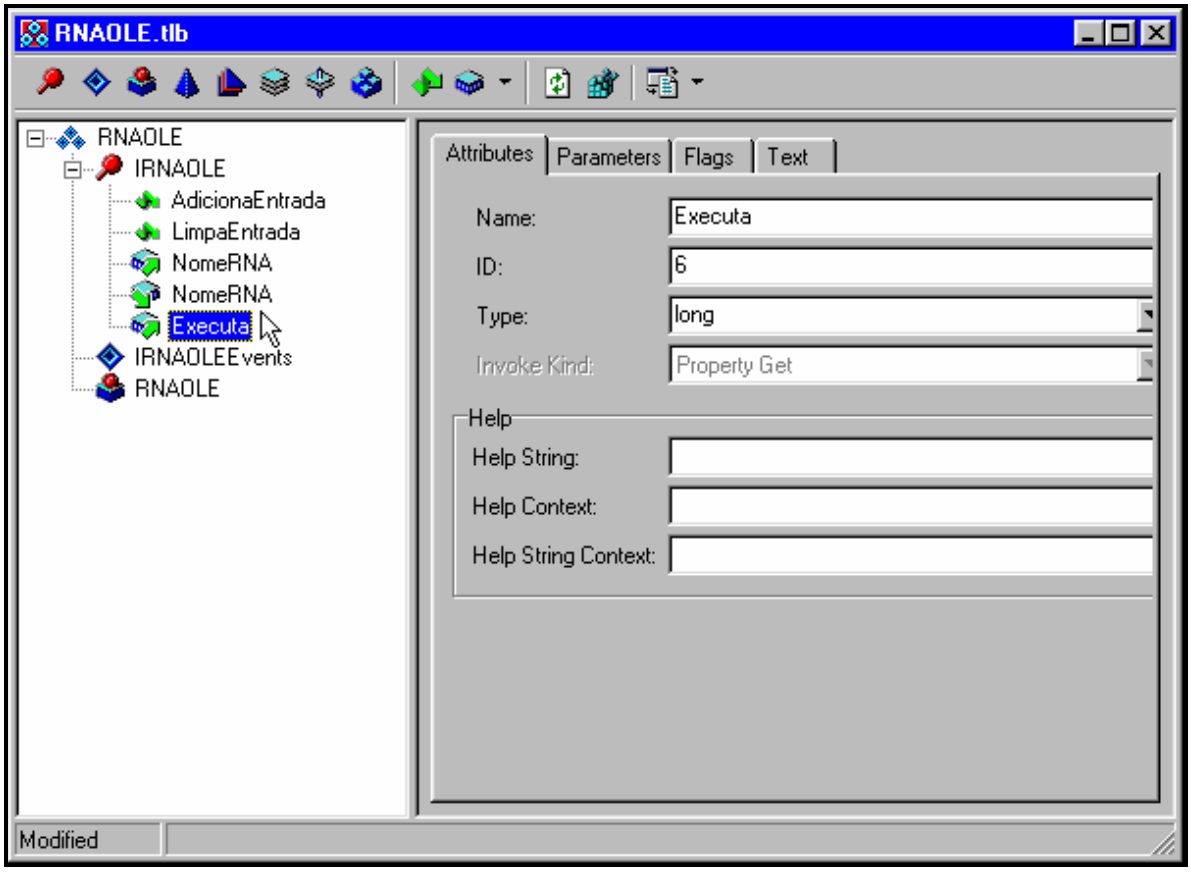
Os métodos disponíveis na classe TSinalOLE são:

a) “InicializaMicrofone”: possui a finalidade de iniciar a captação do sinal e não requer parâmetros;
b) “FinalizaMicrofone”: finaliza a captação do sinal se esta estiver inicializada, também não requerendo parâmetros;
c) “RetornaSinal”: retorna o sinal bruto coletado presente na posição citada no parâmetro do tipo *integer*. Retorna um valor do tipo *double*;
d) “RetornaValor”: retorna o valor que servirá como entrada para a RNA presente na posição mencionada no parâmetro do tipo *integer*. Retorna um valor do tipo *double.*

7.5.3.2.2 TRNAOLE

A *Type Library* da classe TRnaOLE está demonstrada na figura 7.33.

Figura 7.33 – Type Library da Classe Automation OLE TRnaOLE



A classe TRnaOLE apresenta somente a propriedade “NomeRNA”. Esta propriedade deve receber o arquivo de projeto da RNA criado no treinamento efetuado na ferramenta. É uma propriedade para leitura e gravação de valores do tipo *string*.

Os métodos disponíveis na classe TRnaOLE são:

a) “AdicionaEntrada”: adiciona o valor passado por parâmetro do tipo *double* no conjunto de entrada do padrão a ser apresentado a RNA. Não retorna valor;

b) “LimpaEntrada”: elimina valores contidos no conjunto de entrada do padrão a ser apresentado a RNA;

c) “Executa”: retorna o índice (*integer*) da palavra cadastrada na ferramenta para reconhecimento, ou valor igual à –1, caso não reconheça o padrão apresentado.

8. VERIFICAÇÃO DOS RESULTADOS OBTIDOS

Como visto no item 2.7, uma das etapas do reconhecimento de fala é a verificação dos resultados obtidos. Esta etapa visa identificar a performance da RNA *RBF-Fuzzy Artmap* e também do pré-processamento aplicados no reconhecimento de fala. No decorrer deste capítulo, serão demonstrados os resultados obtidos dos testes realizados e também a metodologia empregada para realizá-los.

8.1 PARÂMETROS UTILIZADOS

Para efetuar o treinamento na ferramenta foram definidos parâmetros necessários tanto para aquisição do sinal quanto para o modelo de RNA *RBF-Fuzzy Artmap*.

Utilizou-se como nível de vigilância e valor de corte para treinamento o valor 0,5, sendo que, quanto menor, maior é o reconhecimento, mas menor o acerto. Para a largura da função gaussiana também foi utilizado o valor 0,5. Estes valores foram mencionados por Tontini (1995) como sendo valores iniciais, sendo que com testes e dependendo da aplicação, podem ser alterados ou não, buscando uma melhor adaptação e performance da RNA *RBF-Fuzzy Artmap*.

Para aquisição de sinal, utilizou-se para os parâmetros: taxa de medição do sinal, o valor 2; nível de normalização, o valor igual à 170 e tempo de fixação do sinal, o valor correspondente a 0,5 segundos. Restando dúvidas sobre os parâmetros informados e valores, elas podem ser esclarecidas em Tafner (1996).

Sendo o número de entradas não somente um parâmetro da RNA, mas também necessário ao pré-processamento de sinais, utilizou-se o valor correspondente a 20 entradas, que serão geradas a partir do pré-processamento de sinais descrito no item 5.2, e informadas à RNA como padrão de entrada para cada sinal coletado.

Os parâmetros acima descritos podem ser observados na figura 7.17, que demonstra a entrada dos valores na ferramenta de treinamento RnaTools.

8.2 TREINAMENTO

Fazendo parte do objetivo deste trabalho, como exemplo, reconhecer os dígitos de 0 a 9, e do reconhecimento de fala verificar os resultados, adotou-se o mesmo procedimento empregado por Klabunde (1996) e Fischer (1999). Foram criadas duas massas de dados de locutores diferentes, contendo cada uma 100 palavras, sendo 10 de cada dígito, variando de 0 a 9.

Criou-se um projeto de RNA na ferramenta, e foi adicionada a massa de dados dos dois locutores. Após este etapa da aquisição de sinal terminada, duplicou-se os projetos, onde cada um ficou com as duas massas de dados. Cada projeto de RNA será treinado com apenas um locutor, sendo diferentes nos dois projetos, e a massa de dados presente em cada projeto que não foi utilizada para treinamento, servirá para efetuar os testes.

Na figura 7.19 pode-se verificar o treinamento efetuado no projeto de RNA “CALCULADORA-LOCUTOR 1”, que contém a primeira massa de dados provinda do locutor 1 e a massa de dados do locutor 2 com a finalidade de testar o seu reconhecimento.

8.3 EXECUÇÃO

Para obter os resultados que serão utilizados como base para avaliação, foi realizado o treinamento da massa de dados do locutor 1 no primeiro projeto de RNA, e posteriormente no segundo projeto de RNA, foi utilizada a massa do locutor 2 para treinamento.

Na ferramenta de treinamento RnaTools, pode-se verificar a execução da RNA através da opção “Testes” (descrita no item 7.5.1.2.5), onde a RNA *RBF-Fuzzy Artmap* está em modo de execução permanentemente, e só irá para o modo de treinamento ao se utilizar o recurso de correção “Treinar”, que colocará a RNA em modo de treinamento, fazendo o treinamento do padrão antes não reconhecido, ou reconhecido incorretamente.

Na figura 7.20 pode-se observar o início da apresentação da massa de dados do locutor 2 no projeto de RNA “CALCULADORA-LOCUTOR 1” que foi treinado com a massa de dados do locutor 1.

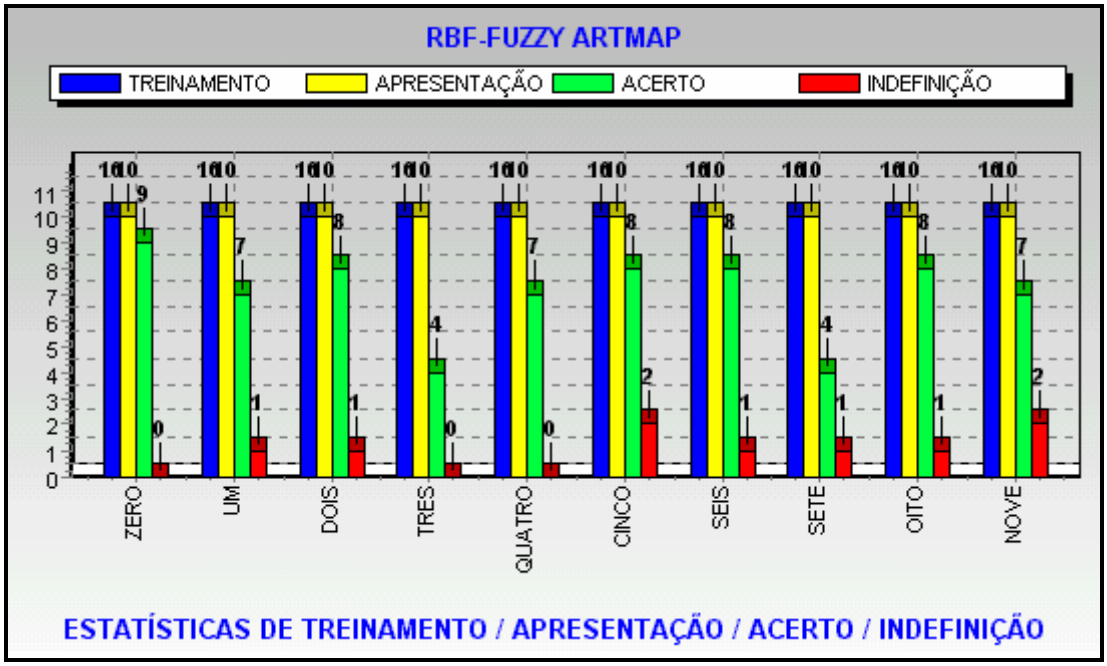
8.3.1 RESULTADOS

Para obter os resultados seguintes, apresentou-se a massa de dados do locutor 2 ao projeto de RNA “CALCULADORA – LOCUTOR 1”, o qual foi treinado com o locutor 1. O resultado da execução dos padrões de entrada na RNA provindos das 100 palavras apresentados pelo locutor 2 é demonstrado na tabela 8.1 e visualmente através do gráfico gerado pela própria ferramenta na figura 8.1.

Tabela 8.1 – Resultados da apresentação do locutor 2 no treinamento do locutor 1

Palavra	zero	um	dois	três	Quatro	Cinco	seis	sete	oito	nove	Total
Indefinida	0%	10%	10%	0%	0%	20%	10%	10%	10%	20%	9%
Incorreta	10%	20%	10%	60%	30%	0%	10%	50%	10%	10%	21%
Certa	90%	70%	80%	40%	70%	80%	80%	40%	80%	70%	70%

Figura 8.1 – Gráfico da apresentação do locutor 2 no treinamento do locutor 1

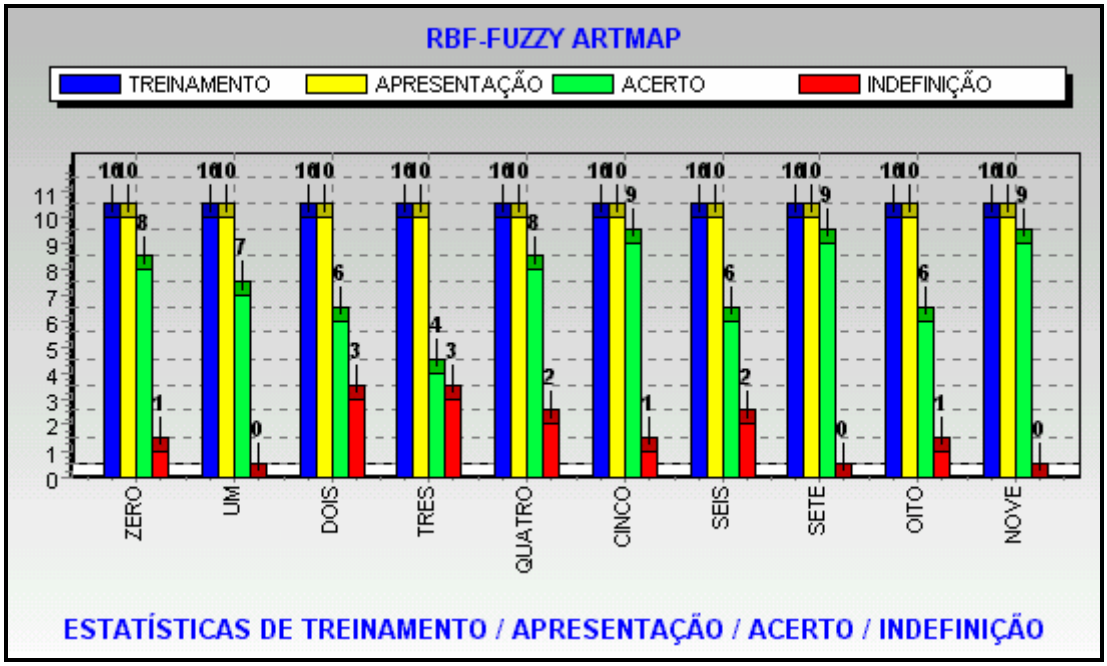


A operação contrária também foi realizada, ou seja, a massa de dados do locutor 1 foi apresentada ao projeto de RNA “CALCULADORA – LOCUTOR 2”, o qual foi treinado com o locutor 2. O resultado desta operação é apresentado na tabela 8.2 e visualmente na figura 8.2.

Tabela 8.2 – Resultados da apresentação do locutor 2 no treinamento do locutor 1

Palavra	Zero	um	dois	três	Quatro	cinco	seis	sete	oito	nove	Total
Indefinida	10%	0%	30%	30%	20%	10%	20%	0%	10%	0%	13%
Incorreta	10%	30%	10%	30%	0%	0%	20%	10%	30%	10%	15%
Certa	80%	70%	60%	40%	80%	90%	60%	90%	60%	90%	72%

Figura 8.2 – Gráfico da apresentação do locutor 2 no treinamento do locutor 1

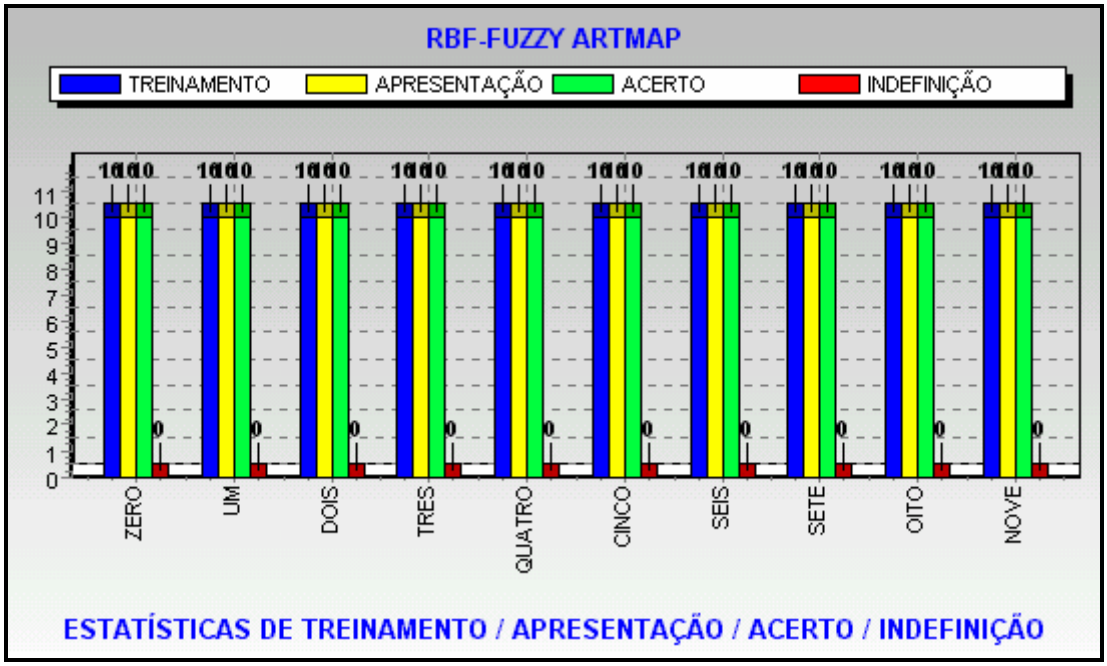


Com o intuito de verificar-se a vantagem mencionada no capítulo 6, item 6.5.3, onde Tontini (1995) afirma que uma das vantagens da *RBF-Fuzzy Artmap* é o seu aprendizado instantâneo, submeteu-se ao projeto de RNA "CALCULADORA – LOCUTOR 1", treinada pelo locutor 1, a mesma massa de dados de treinamento. Os resultados são apresentados na tabela 8.3 e visualmente na figura 8.3.

Tabela 8.3 – Resultados da apresentação do locutor 1 no treinamento do locutor 1

Palavra	zero	um	dois	três	Quatro	cinco	seis	Sete	oito	nove	Total
Indefinida	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Incorreta	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Certa	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Figura 8.3 – Gráfico da apresentação do locutor 1 no treinamento do locutor 1



Pode-se constatar através dos resultados apresentados que reforça-se a afirmação feita por Tontini (1995). A RNA *RBF-Fuzzy Artmap* possui aprendizado instantâneo, obtendo 100% de reconhecimento, necessitando de apenas uma apresentação dos padrões de entrada para treinamento.

8.4 OUTROS TESTES

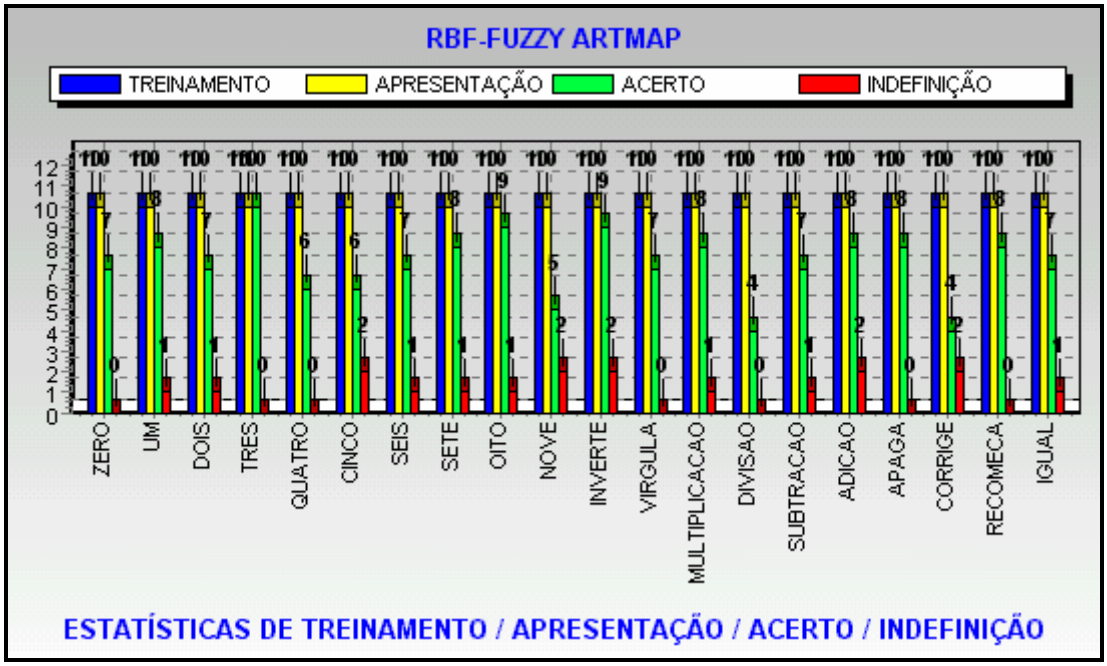
Com a finalidade de realizar outros testes além dos já mencionados, foram adicionadas as massas de dados já existentes, mais 10 palavras para reconhecimento, que foram definidas para as outras operações da aplicação exemplo (calculadora), mencionadas no item 7.5.2.3. A massa de dados será adicionada em 100 palavras, sendo 10 de cada operação da aplicação exemplo.

Após feito o treinamento seguindo o mesmo procedimento adotado para os dígitos de 0 a 9, mas com apenas um projeto de RNA, obteve-se os resultados demonstrados na tabela 8.4 e visualmente na figura 8.4.

Tabela 8.4 – Resultados da apresentação massa de dados de operações da calculadora

Palavra	Inverte	vírgula	divisão	multiplicação	Subtração	adição	apaga	corrige	recomeça	igual	Total
Indefinida	20%	0%	10%	0%	10%	20%	0%	20%	0%	10%	9%
Incorreta	10%	20%	20%	0%	30%	20%	30%	0%	10%	40%	18%
Certa	70%	80%	70%	100%	60%	60%	70%	80%	90%	50%	73%

Figura 8.4 – Gráfico da apresentação massa de dados de operações da calculadora



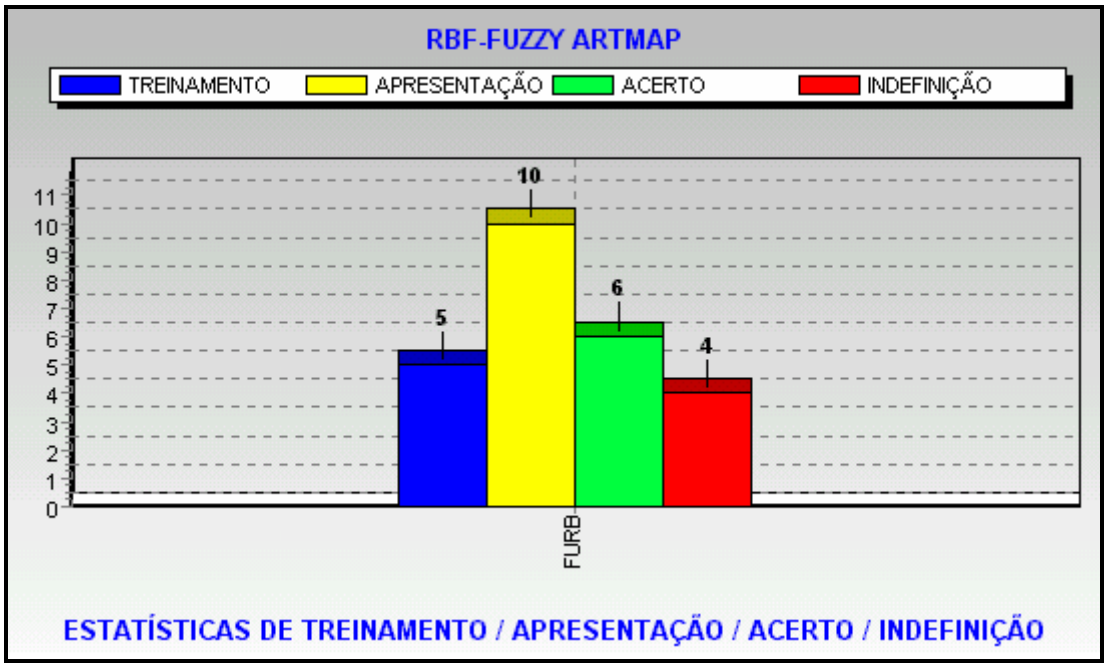
Nota-se um sensível aumento na porcentagem de acerto, que pode ser justificado pelo maior número de sílabas presente nas palavras desta massa de dados, e também pela maior distinção das palavras.

Outro teste realizado foi a criação de um novo projeto de RNA “FURB”, sendo adicionado a este projeto, uma massa de dados contendo 5 palavras “furb”, sendo essas treinadas posteriormente. Na seqüência do teste, foi apresentado a RNA em modo de execução um conjunto de 10 palavras “furb” sem treiná-la após cada apresentação. Lembra-se que para este teste utilizou-se os parâmetros citados no item 8.1, com exceção do número de entradas, optou-se por 10 entradas para a RNA. O resultado deste teste é demonstrado na tabela 8.5 e visualmente na figura 8.8.

Tabela 8.5 – Resultados teste do projeto de RNA “FURB” sem treinamento

Locução	1°	2°	3°	4°	5°	6°	7°	8°	9°	10°	Total
Indefinida	-	-	-	X	X	-	-	X	-	X	40%
Incorreta	-	-	-	-	-	-	-	-	-	-	-
Certa	X	X	X	-	-	X	X	-	X	-	60%

Figura 8.5 – Gráfico do teste do projeto de RNA "FURB" sem treinamento

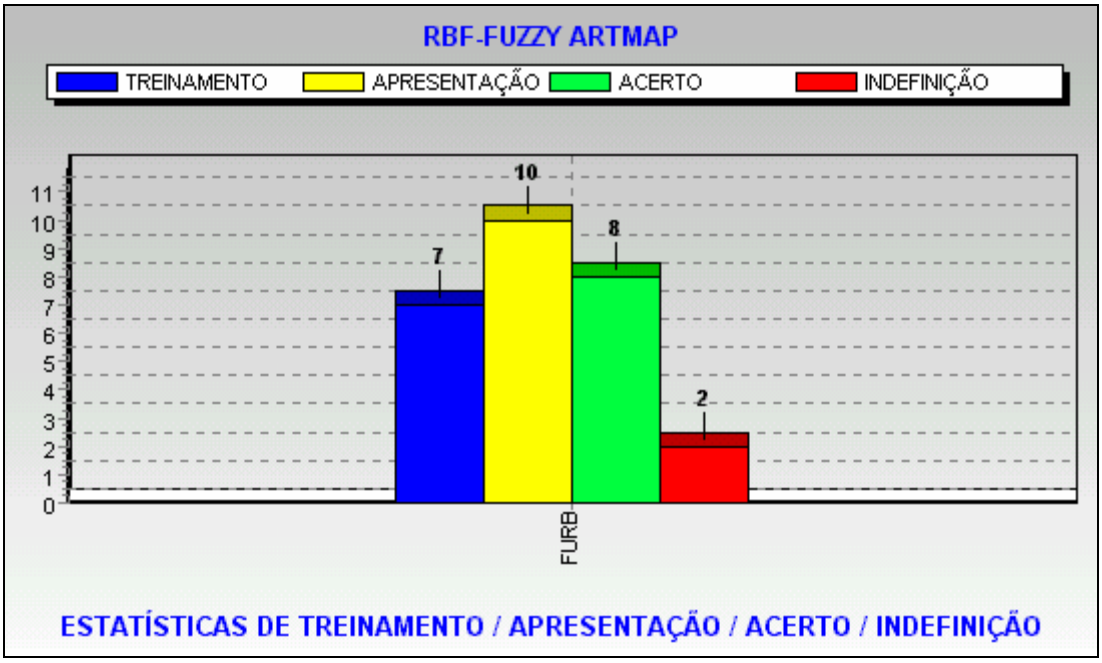


Finalizando este teste, foi realizado o mesmo procedimento, apresentando o mesmo conjunto de palavras anterior, mas efetuando-se o treinamento caso a RNA não tenha reconhecido, ou reconhecido incorretamente. O resultado final é demonstrado na tabela 8.6 e visualmente na figura 8.9.

Tabela 8.6 – Resultados teste do projeto de RNA "FURB" com treinamento

Locução	1°	2°	3°	4°	5°	6°	7°	8°	9°	10°	Total
Indefinida	-	-	-	X	-	-	-	X	-	-	20%
Incorreta	-	-	-	-	-	-	-	-	-	-	-
Certa	X	X	X	-	X	X	X	-	X	X	80%

Figura 8.6 – Gráfico do teste do projeto de RNA "FURB" com treinamento



Verificou-se o comportamento da RNA *RBF-Fuzzy Artmap* ao treinar-se novas palavras e os recursos que a ferramenta RnaTools apresenta para isso e constata-se através deste teste o aprendizado da RNA *RBF-Fuzzy Artmap* implementada.

8.5 COMPARATIVO

Klabunde (1996) realizou testes com a RNA *RBF-Fuzzy Artmap*, e deve-se levar em consideração que em seu protótipo, utilizava-se do pré-processamento do trabalho desenvolvido por Starke (1996). Os resultados obtidos por Klabunde (1996), ao treinar uma RNA com massa de dados de Klabunde (1996) e executá-la com massa de dados de Starke(1996) podem ser observados na tabela 8.7.

Tabela 8.7 – RNA treinada por Klabunde(1996) e executada por Starke(1996)

Palavra	zero	um	dois	três	Quatro	cinco	seis	sete	oito	nove	Total
Indefinida	0%	10%	10%	0%	50%	60%	20%	20%	60%	0%	23%
Incorreta	40%	50%	50%	10%	20%	0%	50%	70%	10%	70%	37%
Certa	60%	40%	40%	90%	30%	40%	30%	10%	30%	30%	40%

A operação contrária, onde a massa de dados de Starke (1996) foi treinada, e a de Klabunde (1996) executada podem ser observados na tabela 8.8.

Tabela 8.8 – RNA treinada por Starke (1996) e executada por Klabunde (1996)

Palavra	zero	um	dois	três	Quatro	cinco	seis	sete	oito	nove	Total
Indefinida	0%	0%	0%	0%	0%	50%	20%	30%	60%	0%	16%
Incorreta	30%	50%	60%	40%	90%	0%	70%	40%	10%	70%	36%
Certa	70%	50%	40%	60%	10%	50%	10%	30%	30%	30%	48%

Fischer (1999) também realizou testes para demonstrar a performance da RNA *RBF-Fuzzy Artmap*, como Klabunde (1996), utilizou dois locutores nos testes e obteve os resultados das tabelas 8.9 e 8.10.

Tabela 8.9 – RNA treinada por locutor 1 e executada por locutor 2

Palavra	zero	um	dois	três	Quatro	cinco	seis	sete	oito	nove	Total
Indefinida	0%	10%	10%	10%	0%	0%	20%	0%	0%	0%	5%
Incorreta	0%	10%	0%	80%	40%	20%	10%	90%	10%	10%	27%
Certa	100%	80%	90%	10%	60%	80%	70%	10%	90%	90%	68%

Tabela 8.10 – RNA treinada por locutor 2 e executada por locutor 1

Palavra	zero	um	dois	três	Quatro	cinco	seis	sete	oito	nove	Total
Indefinida	0%	0%	0%	50%	0%	0%	0%	0%	0%	0%	5%
Incorreta	0%	20%	20%	30%	0%	40%	20%	0%	70%	50%	25%
Certa	100%	80%	80%	20%	100%	60%	80%	100%	30%	50%	70%

Ressalta-se para comparação dos resultados os seguintes pontos:

a) utilizou-se como pré-processamento para este trabalho de pesquisa o utilizado por Tafner (1996), Klabunde (1996) utilizou mesmo de Starke (1996) e Fischer (1999) propôs um novo pré-processamento;
b) este trabalho de pesquisa, Klabunde (1996) e Fischer (1999) realizaram seus testes utilizando o modelo de RNA *RBF-Fuzzy Artmap*, enquanto que Starke(1996) utilizou outros;
c) neste trabalho de pesquisa foram criadas novas massas de dados, mas de locutores diferentes, como em Fischer (1999). Klabunde (1996) utilizou uma própria e a de Starke (1996);
d) o trabalho desenvolvido por Fischer (1999) utilizou os parâmetro nível de vigilância igual à 0.45 para treinamento e 0.22 para execução da RNA, que como mencionado, podem ser alterados buscando uma melhor performance da RNA.

Após considerar-se estes pontos, os resultados obtidos com este trabalho são demonstrados nas tabelas 8.1, 8.2 e 8.3. Visando uma melhor observação dos resultados, montou-se ainda, a tabela 8.11, que contém as porcentagens de acertos obtidos por este

trabalho, Klabunde (1996) e Fischer (1999), onde os valores demonstram a média obtida entre os testes realizados com dois locutores.

Tabela 8.11 – Média de porcentagens de acertos obtidas

Palavra	zero	um	dois	três	Quatro	cinco	seis	sete	oito	nove	Total
Klabunde(1996)	65%	45%	40%	75%	20%	45%	20%	20%	30%	30%	39%
Fischer(1999)	100%	80%	85%	15%	80%	70%	75%	55%	60%	70%	69%
Este trabalho	85%	70%	70%	40%	75%	85%	70%	65%	70%	80%	71%

9. CONCLUSÃO

Desenvolvendo o protótipo obteve-se a oportunidade de utilizar toda a teoria estudada e pesquisada durante os primeiros capítulos deste trabalho, verificando a sua autenticidade e valor científico que as tornam imprescindíveis nos assuntos que este trabalho de pesquisa menciona.

A RNA *RBF-Fuzzy Artmap* mostrou-se apta, após a verificação dos testes realizados, para a aplicação que foi submetida - o reconhecimento de fala. Ao estudá-la e implementá-la, teve-se a oportunidade de conhecer também os modelos de RNA que a compõem (RBF, Artmap e Fuzzy-Artmap) e as aplicações as quais melhor se adapta.

Em relação ao pré-processamento de sinal, notou-se que uma grande porcentagem do acerto no reconhecimento deve-se aos valores dos padrões apresentados a RNA e portanto, ao aplicar modelos de RNA no reconhecimento de fala, deve-se dar grande importância ao pré-processamento escolhido. Lembrando que este também depende de uma aquisição de sinal de igual performance, que represente fielmente o sinal captado.

Pode-se concluir que com uso das classes *Automation OLE* rompe-se a dependência de linguagem elevando a sua portabilidade, e principalmente, pratica-se um dos princípios da programação orientada a objetos, a reutilização.

Demonstrou-se através da aplicação exemplo (calculadora) a funcionalidade das classes *Automation OLE* desenvolvidas neste trabalho de pesquisa, e também a utilização da ferramenta de treinamento RNATools, que possibilita além de gerar a base de conhecimento para uso posterior, testar os padrões adquiridos, com a finalidade de buscar a melhor parametrização da RNA *RBF-Fuzzy Artmap*, visando a melhor performance no reconhecimento.

Através do protótipo apresentado, composto pela ferramenta, classes *Automation OLE* e aplicação exemplo, atingiu-se o objetivo de oferecer ao usuário final, uma interface por comandos de fala, não trazendo grande tempo de desenvolvimento e pesquisa ao desenvolvedor de software.

9.1 LIMITAÇÕES

A interface por comandos de fala, ou seja, possibilitar ao usuário interagir com o software através da pronúncia de comandos falados, visa atingir alta taxa de acertos dos padrões apresentados. Sabe-se que tal performance não é alcançada somente utilizando-se os valores sugeridos para os parâmetros, tais como fi, mas é responsabilidade do usuário da ferramenta (desenvolvedor de software) buscar os valores mais adequados à sua aplicação. Isto significa que ao adquirir o sinal, treinar e testar a RNA, ele deve identificar a melhor configuração do conjunto de parâmetros, obtendo assim, o melhor desempenho da sua utilização. Observa-se que para um sistema real e em produção não é possível alcançar 100% de acerto, porém deve-se sempre buscar a configuração mais adequada e com melhores resultados.

Menciona-se que para os testes apresentados no capítulo 9, não foi feito este tipo de aprimoramento e portanto, pode-se constatar que as porcentagens mencionadas poderiam ser melhoradas, dependendo do conjunto de parâmetros a se utilizar.

9.2 SUGESTÕES PARA TRABALHOS FUTUROS

Como sugestão para trabalhos futuros em relação a RNAs, sugere-se que sejam estudadas e implementadas as RNAs mencionadas no item 3.6, que são utilizadas no reconhecimento de padrões. Também sugere-se o uso de novas RNAs híbridas criadas a partir da junção de várias modelos combinados, como é o caso da RNA *RBF-Fuzzy Artmap*, buscando as vantagens de cada uma e adicionando-as em uma ferramenta de treinamento como desenvolvida neste protótipo, permitindo reutilizá-las quando necessário, sem dificuldades.

Recomenda-se desenvolver um estudo sobre pré-processamento de sinais a fim de extrair todo o potencial desta etapa, fornecendo à RNA um padrão de entrada que represente fielmente a palavra falada, distinguindo mudanças de timbre, altura e intensidade.

Referindo-se à aquisição de sinal poderia ser desenvolvido uma técnica de “auto-aquisição”, ou seja, a partir da captação de um único sinal, estabelecer níveis de

variação sobre este, buscando simular a locução das palavras. Com esta técnica poderia eliminar grande parte da tarefa de aquisição de sinal, bastando a locução de poucas palavras para treinamento. Dever-se-á levar em conta fatores como: intensidade, altura e timbre da voz e outros que poderão fazer parte do conjunto, como a freqüência e o tempo.

Em relação as classes *Automation OLE*, desenvolver as classes implementadas e utilizadas no protótipo deste trabalho de pesquisa, no ambiente de desenvolvimento Kylix, da empresa Borland Corporation, visando ser utilizadas no SO Linux.

REFERÊNCIAS BIBLIOGRÁFICAS

AMORIM, Antônio. **Fundamentos científicos da fonoaudiologia**. São Paulo: Ciências Humanas, 1981. 123p.

AMORIM, Antônio. **Fonoaudiologia geral**. São Paulo: Livraria Pioneira, 1972. 116p.

BAEKER, Ronald M.; Grudin, Jonathan. **Human-computer interaction** : toward the year 2000. San Francisco: Morgan Kaufmann Publishers, 2000. 949p.

BETEMPS, Rafael da Silva; Silveira, Liliana Nieto. **Redes neurais**: conceitos e métodos. Pelotas, RS, n. 1, p.31-42, 1997.

BORLAND Delphi, versão 5.02. Borland Corporation, 1999. Ambiente de desenvolvimento/Linguagem de programação Object Pascal. 1 CD-ROM.

BRUNS, Fábio Augusto. **Protótipo para o reconhecimento de palavras através da fala**. 1995. 67 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CAMPESTRINI, Márcio Rodrigo. **Protótipo de um sistema de reconhecimento de caracteres baseado em redes neurais**. 2000. 36 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CANTÚ, Marco. **Dominando o Delphi 5** – "a bíblia". São Paulo: Pearson Education do Brasil, 2000. 860p.

COLLING, Suzete Teresinha. **Utilização da tecnologia ActiveX Data Objects (ADO) em um sistema com objetos distribuídos**. 2000. 77 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

DALABRIDA, Alexandre Daniel; Itaborai, Gilvani Kisner**. Utilização da Inteligência artificial em reconhecimento de voz**. Universidade Regional de Blumenau, Blumenau, set. 2000. Disponível em: <http://www.inf.furb.rct-sc.br/~alx/ia.html>. Acesso em: 09 maio 2001.

FISCHER, Reinaldo dos Santos. **Aplicação de uma nova proposta de pré-processamento utilizando o modelo de redes neurais RBF-Fuzzy Artmap para o reconhecimento de fala**. 1999. 49 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

FURLAN, José Davi. **Modelagem de objetos através de UML – the unified modeling language**. São Paulo: Makron Books, 1998.

HECKMANN, Jacques Robert. **Sistematização das atuais técnicas de inteligência artificial e análise de sua aplicabilidade**. 1998. 69 f. Monografia (Curso de Pós Graduação), Universidade Regional de Blumenau, Blumenau.

HONORATO JUNIOR, Paulo Francioli**. COM - Component Object Model**. São Paulo. Universidade Regional de Blumenau, Blumenau, set. 2000. Disponível em: <http://www.inf.furb.rct-sc.br/~alx/ia.html>. Acesso em: 09 maio 2001.

HUGO, Marcel. **Construção de um protótipo de software comandado por voz**. 1992. 41 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

HUGO, Marcel. **Uma interface de reconhecimento de voz para o sistema de gerenciamento de central de informação de fretes.** 1995. 60f. Dissertação (Mestrado em Engenharia da Produção) – Universidade Federal de Santa Catarina, Florianópolis.

KITANO, Hiroaki. **Speech-to-speech translation**. Boston: Kluwer Academic Publishers, 1994. 193p.

KLABUNDE, Charles Cristiano. **Aplicação do modelo de rede neural RBF-fuzzy artmap para o reconhecimento de fala**. 1996. 64 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

LOESCH, Cláudio e SARI, Solange T. **Redes neurais artificiais :** fundamentos e modelos**.** Blumenau: FURB, 1996.

MATRAS, Jean-Jacques. **O som.** São Paulo: Universidade Hoje, 1995. 116p.

MOSER, Bóris. **Utilização da transformada de Fourier no tratamento de sinal para a utilização em uma rede neural**. 1997. 59 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

REVOX Sistema de Reconhecimento de Voz. Caxias do Sul. Desenvolvido pela Universidade Federal do Rio Grande do Sul (UFRGS), 1997. Apresenta informações sobre um sistema para reconhecimento de fala. Disponível em <http://www.ucs.br/revox/interface1.htm>. Acesso em: 20 abr. 2001.

SCHMITZ, Eber; SILVEIRA, Denis. **Desenvolvimento de software orientado a objetos –** utilizando UML e Delphi 5. Rio de Janeiro: Brasport, 2000.

STARKE, Júlio Frederico. **Comparativo entre modelos de redes neurais aplicadas ao reconhecimento da fala**. 1996. 69 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

STEVE, Teixeira; PACHECO, Xavier. **Delphi 5 -** developers guide. London: Pearson, 1999. 1556p.

TAFNER, Malcon Anderson. **O reconhecimento de palavras faladas isoladas usando redes neurais artificiais**. 1996. 89f. Dissertação (Mestrado em Engenharia da Produção) – Universidade Federal de Santa Catarina, Florianópolis.

TONTINI, Gérson. **Automatização da identificação de padrões em gráficos de controle estatístico de processos (CEP) através de redes neurais com lógica difusa**. 1995. 137 f. Tese (Doutorado em Engenharia Mecânica) – Universidade Federal de Santa Catarina, Florianópolis.

XEREZ, Marcos de; Filho, Ilson W. Rodrigues; Tafner, Malcon Anderson. **Redes neurais artificiais** – introdução e princípios de neurocomputação. Blumenau: Editora da Furb, 1996. 199p.