UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Reconhecimento Semântico Através de Redes Neurais Artificiais

por

DANIEL NEHME MÜLLER

Dissertação submetida à avaliação como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Philippe O. A. Navaux
Orientador

Porto Alegre, junho de 1996.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

MÜLLER, DANIEL NEHME

Reconhecimento Semântico Através de Redes Neurais Artificiais / DANIEL NEHME MÜLLER.—Porto Alegre: CPGCC da UFRGS, 1996.
168 p.: il.

Dissertação (mestrado)—Universidade Federal do Rio Grande do Sul. Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, BR-RS, 1996. Orientador: Navaux, Philippe O. A.

1. Redes Neurais Artificiais, 2. Inteligência Artificial, 3. Processamento Paralelo e Distribuído. I.Navaux, Philippe O. A. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Sistema de Biblioteca da UFRGS

33085

681.3.013(043)
M958R

INF
1997/194334-2
1997/07/10

MOD. 2.3.2

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Helgio Trindade.
Pró-reitor de Pesquisa e Graduação: Prof. Cláudio Scherer.
Diretor do Instituto de Informática: Prof. Roberto Tom Price.
Coordenador do CPGCC: Prof. Flavio Rech Wagner.
Bibliotecária do Instituto de Informática: Zita Prates de Oliveira.

AGRADECIMENTOS

Agradeço a Deus, pelo caminho apresentado.

Agradeço ao professor Navaux, pelas lições de sabedoria, discernimento e refinamento.

Agradeço aos colegas que me ajudaram a aprender a arte da programação das Redes Neurais Artificiais, em especial a Fernando Santos Osório.

Agradeço, enfim, às pessoas que propiciaram o ambiente de cooperação existente no laboratório do Instituto. Espero que esta cooperação dê frutos e contagie a nossa sociedade.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Um dos grandes desafios atuais da computação é ultrapassar o abismo existente entre o homem e a máquina. Para tanto, o desafio passa a ser a formalização de estados mentais e sua modelagem computacional. Isso é necessário, uma vez que o homem somente conseguirá comunicar-se com uma máquina quando esta puder dar e receber informações sem que o homem precise aprender uma forma especial de comunicação. É necessário, portanto, que a máquina aprenda a comunicar-se como o homem.

Neste sentido, o estudo da linguagem torna-se uma porta aberta para criar uma computação que se adapte ao homem e, ao mesmo tempo, favoreça pesquisas que visem uma melhor compreensão do funcionamento do cérebro, da linguagem e do aprendizado do próprio homem.

O presente trabalho mostra que o computador possui um potencial de comunicação ainda inexplorado. Por este motivo, em estudos anteriores procurou-se a verificação do atual estágio de modelagem de comunicação homem-máquina em comparação à evolução da linguagem humana. Constatou-se, então, que a máquina pode chegar a uma efetiva comunicação com o homem, embora jamais espontânea, como se vê na ficção científica. O que é possível é a auto-organização pelo computador de sinais provenientes de seu meio, visando a realização de determinadas tarefas.

Esses sinais do meio em que está o computador são exatamente o que justifica suas ações, o que dá significado ao que lhe é transmitido, assim como o que ocorre no homem. Para que se modele o reconhecimento semântico de frases é necessário que se encontre uma forma de codificar os sinais do meio para que estes, acompanhando a frase, permitam o reconhecimento de seu significado.

Porém, como o objetivo deste trabalho é a implementação do reconhecimento semântico, e não a recepção de sinais, optou-se por uma codificação representativa dos sinais externos. Esta codificação permite que, através da tecnologia das Redes Neurais Artificiais, seja possível a implementação de relações semânticas entre palavras e entre frases, permitindo a classificação para posterior reconhecimento.

A implementação computacional realizada permite o reconhecimento de frases, mesmo com alteração de palavras e número de palavras. O protótipo aqui apresentado mostra que, mesmo com uma estrutura extremamente mais simples que outros sistemas de reconhecimento de língua natural, é possível uma adequada identificação de frases.

Palavras-chave: reconhecimento semântico, redes neurais artificiais, cognição, linguagem natural.

TITLE: "SEMANTIC RECOGNITION THROUGH ARTIFICIAL NEURAL NETS"

ABSTRACT

One of the great challenges of computation nowadays is to cross the abyss between man and machine. Thus, the challenge becomes the formalization of mental states and its computational modelling. This is necessary since man will only get to communicate with a machine when this machine is able to give and receive information without man needs to learn a special way to communicate. Therefore, it is necessary that the machine learns to communicate with man.

In this sense, the study of the language becomes an open door in order to create a computation that may be adapted to man, and, at the same time, may help researches which aim at a better comprehension of the brain functioning of the language and of man's learning.

This work shows that the computer has a potential for communication that has not been explored yet. For this reason, in prior studies we tried to verify the present stage of man-machine communication modelling in comparison with the human language evolution. We verified, then, that the machine can reach an effective communication with man, but never spontaneous, as we see in scientific fiction (Sci-Fi). What can be possible is the self-organization by computer of signals deriving from its own environment, aiming at realization of specifics tasks.

Those signals of the computer environment are exactly what justifies its actions, what gives meaning to what is transmitted to it in the same way that happens with man. In order to mould the Semantic Recognition of phrases it is necessary to find out a way of codifying the signals of the environment so that these signals, accompanying a phrase, may permit recognition of its meaning.

However, as the purpose of this work is the implementation of the Semantic Recognition, and not the reception of signals, we have opted for a representative codification of external signals. This codification allows that, through the Artificial Neural Nets technology, the implementation of semantic relations among words and phrases may be possible, permitting the classification for posterior recognition.

The computational implementation realized permits the recognition of phrases, even with alteration of words and number of words. The prototype presented here shows that, even with one structure extremely simpler than other systems of Natural Language Recognition, an adequate identification of phrases is possible.

Key words: semantic recognition, artificial neural nets, cognition, natural language.

1 INTRODUÇÃO

Uma das fronteiras hoje existentes na Ciência da Computação é a representação da linguagem humana. Diversas metodologias foram usadas nos últimos 30 anos para possibilitar a compreensão da linguagem humana, mas sem muito sucesso. Dada sua complexidade e ambigüidade na compreensão de seu significado, a linguagem continua sendo uma fronteira em exploração na Computação.

Justamente no sentido de explorar o desconhecido é que se busca neste trabalho esclarecer a construção e o funcionamento da linguagem na tentativa de implementar seu mecanismo no computador. Para se atingir esta meta, procurou-se identificar em trabalhos anteriores ([MÜL 93] [MÜL 95]) a função da linguagem bem como sua forma de organização: porque o homem necessita da linguagem, como ele a constrói, quais estruturas neuronais permitem seu funcionamento. Baseando-se nas conclusões destes trabalhos, chegam-se a diversas hipóteses que podem auxiliar na modelagem da linguagem para um sistema computacional. Estas hipóteses serão analisadas a seguir.

1.1 Observações de Ordem Neurológica

O estudo do comportamento neurológico do cérebro possibilita a concepção de modelos matemáticos na tentativa de permitir uma reprodução da distribuição de cargas elétricas em seu interior. Por essas características, a pesquisa nessa abordagem é muito comentada por físicos, matemáticos e neurologistas. A contribuição de cientistas destas áreas foi muito importante para a evolução de sistemas conexionistas, também conhecidos como Redes Neurais Artificiais (RNAs). Dentre estas contribuições, pode-se citar:

- o primeiro neurônio formal, idealizado por McCulloch & Pitts [KOH 77];

- a descoberta da **transmissão gradativa do sinal elétrico** entre os neurônios, devido à natureza química da transmissão, o que deu origem à função de transferência *sigmoid*, que representa matematicamente essa transmissão gradativa do sinal [RUM 86]. Esse avanço permitiu a representação de padrões não-lineares, ou seja, qualquer tipo de padrão numérico;

- a descoberta da **formação de estruturas para detecção de características** - os chamados mapas cerebrais -, que possibilitam o aprendizado de objetos e ações provenientes do meio ambiente, permitiu o desenvolvimento de mapas auto-organizáveis de características [KOH 77] [KOH 90];

- a constatação de que **os sinais no cérebro são distribuídos assincronamente e paralelamente**, o que permite a concepção de computadores com o processamento paralelo e distribuído em vários processadores independentes [RUM 86].

Além destas descobertas, há outras que ainda estão por proporcionar desafios nesta área. Um exemplo refere-se à distinção da rede neural do cérebro em duas áreas, uma com neurônios geneticamente dispostos a determinadas tarefas, e outra, localizada no neocórtex, com neurônios de funções variáveis, permitindo assim o aprendizado e o processamento de informação num nível mais elevado do que o dos sentidos [ANB 75] [KOH 77] [CRI 86] [DUN 93].

Esse exemplo pode sugerir, dentre outras coisas, que a idealização de sistemas computacionais sofisticados através das RNAs podem vir a exigir a concepção de neurônios especializados, ou seja, com o aprendizado já formado ou sem capacidade de aprendizado. Isso possibilitaria a representação dos sentidos (visão, audição, etc.), numa alusão ao pré-processamento das informações do mundo exterior que chega ao sistema de aprendizado propriamente dito. No presente trabalho será utilizada esta concepção na definição dos campos semânticos de palavras e frases.

1.2 Observações de Ordem Psicológica

O estudo do comportamento contribui muitas vezes para reforçar hipóteses do estudo neurológico, e servindo como auxiliar na concepção de sistemas computacionais que busquem representar um determinado comportamento ou a união entre este e o neurológico. A contribuição do estudo psicológico vem ao encontro com a modelagem de determinado sistema, dependendo da aplicação que se pretende. Como este trabalho pretende a abordagem do comportamento com relação à linguagem, ele ficará restrito à análise dos apectos que giram em torno deste tema.

1.2.1 Modelo de Processamento da Linguagem

Para melhor visualizar o comportamento de como o homem recebe sensações do exterior para posteriormente transformá-las em informações internas, será utilizada a abordagem de Anderson [ANB 75]. Nesta, Anderson divide a recepção e o processamento de sensações em três partes: *percepção primária*, *percepção secundária* e *memória associativa*.

A *percepção primária* seria a recepção dos sinais provenientes dos canais sensoriais. Esta recepção seria analógica (diversos níveis de sinal) e paralela (várias avaliações simultâneas), onde haveria um processo de seleção dos sinais mais intensos. A rede neural responsável pela percepção primária é predisposta geneticamente para possibilitar a identificação de objetos do mundo: formas, cores, sons, sabores, odores, etc.

A *percepção secundária* é o aprendizado resultante do desenvolvimento do sistema motor do indivíduo. Ela proporciona a seleção dos elementos provenientes da percepção primária e o relacionamento destas informações com outras existentes na memória. Isso seria possível graças ao processo de diferenciação, que transformaria o sinal proveniente da percepção primária em digital (dois níveis de sinal)

e serial (apenas uma avaliação por vez). Esta representação planificada facilitaria a combinação entre os elementos externos (sensoriais) com os internos (memorizados). Esta combinação seria possível graças ao processo de enriquecimento, onde haveria a comparação entre os elementos externos e internos, resultando numa nova representação da realidade verificada na percepção primária.

A *memória associativa* realiza a composição entre os elementos resultantes da percepção e outros elementos construídos sobre associações. A função desta composição seria a classificação conceitual de padrões, que é a base do esquema associativo (esquema cognitivo), o qual faria a preparação do organismo para o momento seguinte, para próximo evento. A organização da memória aparentemente divide-se numa parte não-verbal e outra verbal, a qual, por sua vez, possui uma porção representacional, responsável pela compreensão da linguagem, e outra executiva, dedicada a sua produção.

Uma vez estabelecida esta estruturação da recepção e processamento da informação pelo homem (veja figura 1.1), pode-se idealizar uma modelagem computacional para sua implementação. Para a percepção primária, pode-se conceber sensores e codificadores adequados para visualização e audição de padrões e um sistema de seleção dos sinais mais intensos. Para a percepção secundária seria necessário um sistema de busca e seleção na memória das informações mais adequadas com o sinal recebido do exterior e sua combinação. Caso a combinação não seja semelhante a nenhuma outra existente na memória, esta sofre uma reacomodação com a inserção de novas informações. Isto poderia ser auxiliado, ainda, por um sistema de produção de sons e escrita. A memória em si seria um sistema classificatório que auxiliaria na identificação das informações provenientes do exterior através da percepção secundária.

Evidentemente a modelagem descrita poderia ser implementada através de diversas metodologias exitentes em Inteligência Artificial (IA), porém aqui será utilizada a abordagem das RNAs, conforme será detalhado posteriormente.

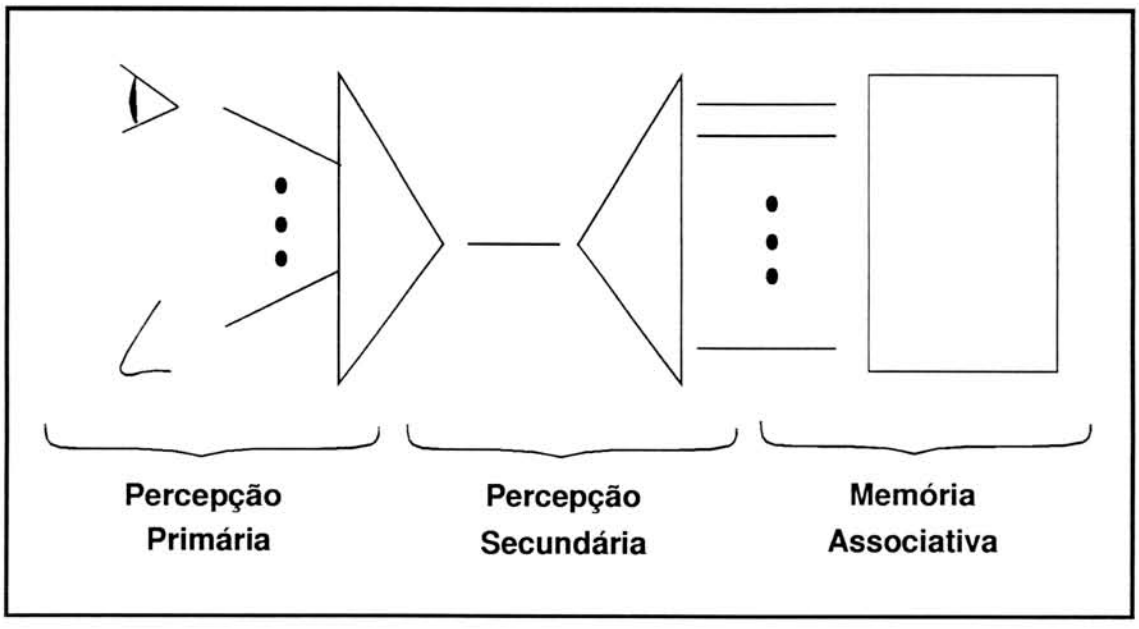


FIGURA 1.1 - **Modelo:** as três fases do modelo de processamento de informação.

1.2.2 O Desenvolvimento da Linguagem

Uma vez definido um Modelo para o Processamento da Linguagem (MPL), deve-se buscar a verificação da forma pela qual este processamento deverá realizar-se. Em outras palavras, há a necessidade de se definir como as informações são organizadas no interior do MPL. Para tanto, será feita uma análise de como a criança representa a informação da qual deriva a linguagem [PIA 71] [FAR 89].

Segundo Piaget [PIA 71], a criança, nos seus primeiros anos de vida, possui *indícios*, ou seja, características de determinados objeto de seu mundo, compreendidos como ações do meio que correspondem a determinada percepção. De acordo com o MPL, para permitir a produção do indício seria necessário a utilização somente da percepção primária e secundária, sem o uso da memória associativa.

Numa fase posterior, após criada a função simbólica, a criança desenvolve *semi-signos*, ou seja, a representação mental de características de um determinado objeto na forma de uma imagem deste, compreendida como sendo ações internalizadas do meio. A criança já possui a compreensão formada das relações dos objetos de seu mundo, porém ainda não possui a capacidade de produção verbal delas. Neste

ponto, voltando ao MPL, já é utilizado todo o sistema de percepção e memória, sem, no entanto, existirem associações de alto nível, que permitirão, na próxima etapa, a abstração dos objetos [MÜL 95].

Finalmente, por volta dos 7 anos, a criança já possui a capacidade de abstração de objetos, permitindo a associação de palavras a conceitos, sendo então possível o desenvolvimento de todas as operações lógicas, formando, assim, *signos*. As três fases do desenvolvimento da linguagem na criança são mostradas na figura 1.2.

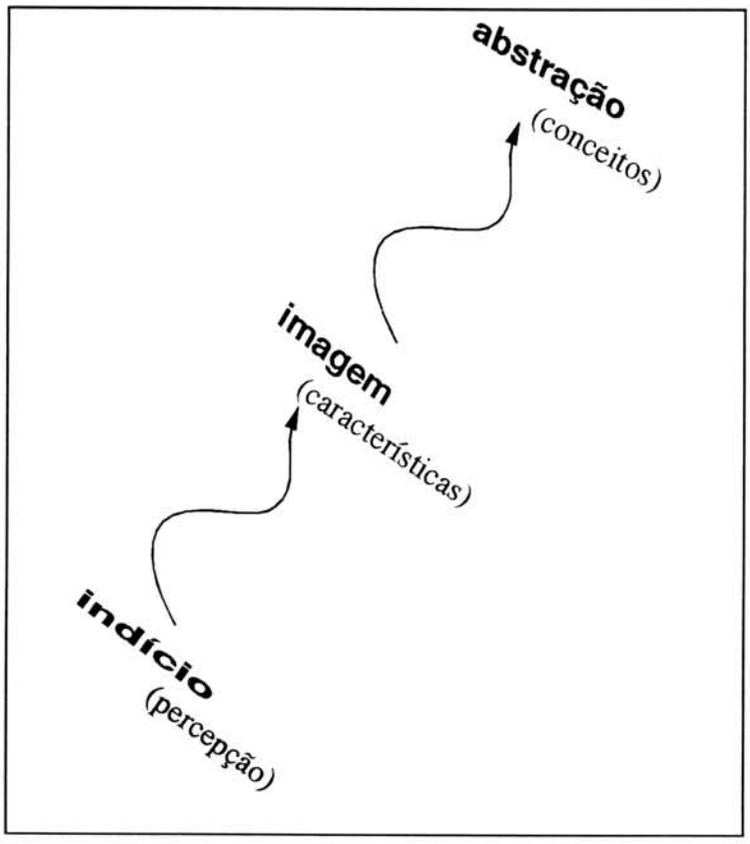


FIGURA 1.2 - **Fases:** o desenvolvimento da linguagem na criança.

Para efetuar o desenvolvimento da linguagem no MPL, poder-se-ia implementar, no máximo, até a segunda fase, uma vez que a associação de características a determinado objeto não chega a ser tão complexo quanto as relações necessárias para a produção de frases.

Neste trabalho serão empregadas as concepções anteriormente abordadas no MPL, o modelo de processamento e desenvolvimento da linguagem. Não objetiva-se aqui a reprodução da linguagem nas duas primeiras fases, uma vez que isso exigiria uma formalização ainda não existente deste processo. Visa-se, isso sim, a reprodução *parcial* do MPL e, em linhas gerais, do processo de utilização da linguagem na forma da primeira e/ou segunda fase de desenvolvimento anteriormente analisadas. Cabe ressaltar, ainda, que não foi feita uma implementação *ao nível da linguagem de uma criança*, mas sim foram utilizadas estas concepções para aperfeiçoar a comunicação homem-máquina, objetivo este da própria IA.

1.3 Uma Abordagem Psicolingüística do Significado

Uma vez estabelecido o MPL, é preciso identificar os processos mais internos dos elementos que permitem sua compreensão. Estes elementos são comuns à origem do símbolo, e a partir deles é que se forma o significado. Uma vez possuindo-se o significado da linguagem, obtém-se sua compreensão. Partindo deste princípio, será investigada a forma pela qual é construído e processado o significado.

Segundo Clark & Clark [CLA 90], costuma-se chamar o estudo do significado de *semântica*. Dificilmente será encontrada uma abordagem que satisfaça a origem do significado na linguagem humana. Estudos sobre a origem antropológica da linguagem indicam que a comunicação tornou-se necessária para melhorar a organização da sobrevivência em grupo [DUN 93]. Mas o significado provavelmente surgiu antes da própria linguagem, quando o homem apenas fazia um contato vocal avançado.

Analisando-se, por outro lado, a origem da linguagem na criança, ve-se que são construídos por ela indícios de objetos que facilitam sua posterior comunicação com o mundo. Estes indícios são características de objetos físicos ou não

que chegam até a criança através de seus sentidos (seção 1.2). Talvez seja este o ponto inicial da origem do significado e da linguagem.

Piaget [PIA 80] afirmava que a linguagem seria uma decorrência do processo de formação da função semiótica ou simbólica. Essa função seria a responsável pela construção de uma abstração, o que permitiria a reconstituição de um objeto sem sua presença. A linguagem teria a função de comunicar e aprimorar o processo de abstração, dando-lhe o desenvolvimento necessário para a pessoa alcançar a realização de todas as operações lógicas. Em suma, a linguagem deriva do processo de criação de símbolos, do significado.

Para sustentar esta hipótese pode-se argumentar da seguinte maneira: inicialmente a linguagem serve como uma *expressão de sensações* (primeira fase do desenvolvimento da linguagem - ver seção 1.2.1), ou seja, há a *compreensão* de sinais provenientes do meio em que a pessoa vive e a *produção* de sons ou gestos para exprimir os sentimentos relacionados a esses sinais (veja figura 1.3). Obviamente não há neste argumento uma linguagem no sentido literal, com palavras e conceitos, mas há, ao menos, a compreensão do *significado* das ações do meio. A partir dessa compreensão primária, o organismo realiza determinadas reações que certamente comunicam sensações. Nesta comunicação primária ainda não existe a intenção de comunicar, a expressão das sensações é feita sem hesitação, uma vez que nesta fase inicial não há relações de informações que permitam fazer julgamentos ou valorações de atitudes.

Supondo que o significado da linguagem seja oriundo das sensações do meio juntamente com outras informações existentes na memória, então se poderia afirmar que o que é produzido na forma de linguagem ou comunicação toma por base a síntese dessas sensações e informações. Em outras palavras, tudo o que é comunicado teria um significado.

Partindo do princípio de que *o que é comunicado faz sentido*, será construído, com base no MPL, o significado de palavras e o significado de frases. Assim

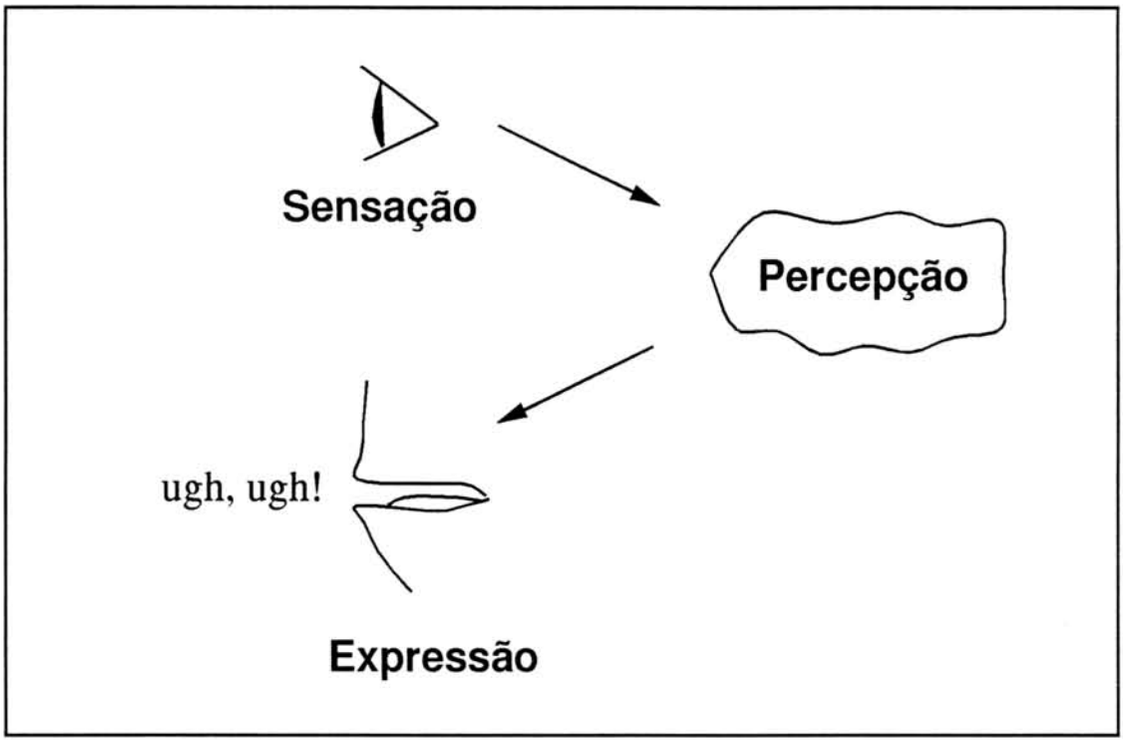


FIGURA 1.3 - **Linguagem:** o início da expressão oral.

como as crianças necessitam aprender o significado das palavras [CLA 90], também o modelo computacional do MPL terá de ter um sistema para aprendizado do significado das palavras e frases.

1.4 O Modelo Proposto

O que se propõe é um modelo de sistema computacional que não leve em conta a tradicional abordagem simbólica da IA, mas sim a abordagem subsimbólica representada pelas RNAs. No momento em que se opta por este paradigma, pode-se renunciar às regras que a lingüística nos apresenta, uma vez que as RNAs são modelos matemáticos capazes de aprendizado através de exemplos. Uma experiência interessante neste sentido é a do sistema NETtalk [SMO 90], que será apresentado na seção 3.3.

A representação da linguagem através de técnicas convencionais de IA simbólica exigem um excesso de formalização de regras lingüísticas, o que é dispensado nas técnicas de IA subsimbólica (RNAs), as quais, a partir de exemplos de frases, possuem a característica de captação das invariantes estatísticas dos exemplos, permitindo o treinamento de padrões de frases, para posterior reconhecimento de outras frases dentro dos mesmos padrões, seguindo as mesmas regras de construção, sem, no entanto, tê-las definido explicitamente. Por outro lado, as técnicas de IA simbólica exigem a verificação caso-a-caso da forma de construção de padrões de frases, sem mencionar as limitações existentes em termos de combinação de elementos dentro das construções previstas. Por estas razões o presente trabalho busca aprimorar o desenvolvimento de técnicas de reconhecimento de linguagem através de RNAs, entendido como um campo ainda pouco explorado em termos de IA.

A partir do paradigma das RNAs, foi definido, com base em estudos anteriormente citados, um modelo para o processamento da linguagem (MPL). Unindo as premissas do MPL e da metodologia das RNAs foi constituído um protótipo computacional para a compreensão semântica de uma frase. Evidentemente que este protótipo é limitado a um contexto de uma determinada aplicação onde há um universo restrito de discurso. Esta restrição dá-se devido tanto à capacidade do equipamento utilizado quanto à praticidade da aplicação.

O protótipo em si é composto basicamente de duas partes principais: uma rede neural para o aprendizado das relações semânticas de palavras e outra para as relações de frases. Na primeira rede, obtém-se um mapa das relações semânticas das palavras envolvidas, e na segunda, outro mapa das relações das frases, como se pode observar na figura 1.4. O posterior reconhecimento é feito a partir deste último mapa, que indica qual frase-padrão (com um significado definido) é semelhante à frase a reconhecer. Com este mecanismo básico é possível realizar-se o *reconhecimento do significado* de quaisquer frases dentro do contexto pré-definido, sem a necessidade de formalização e definição de regras lingüísticas ou suas relações, como em muitos sistemas convencionais para reconhecimento de língua natural.

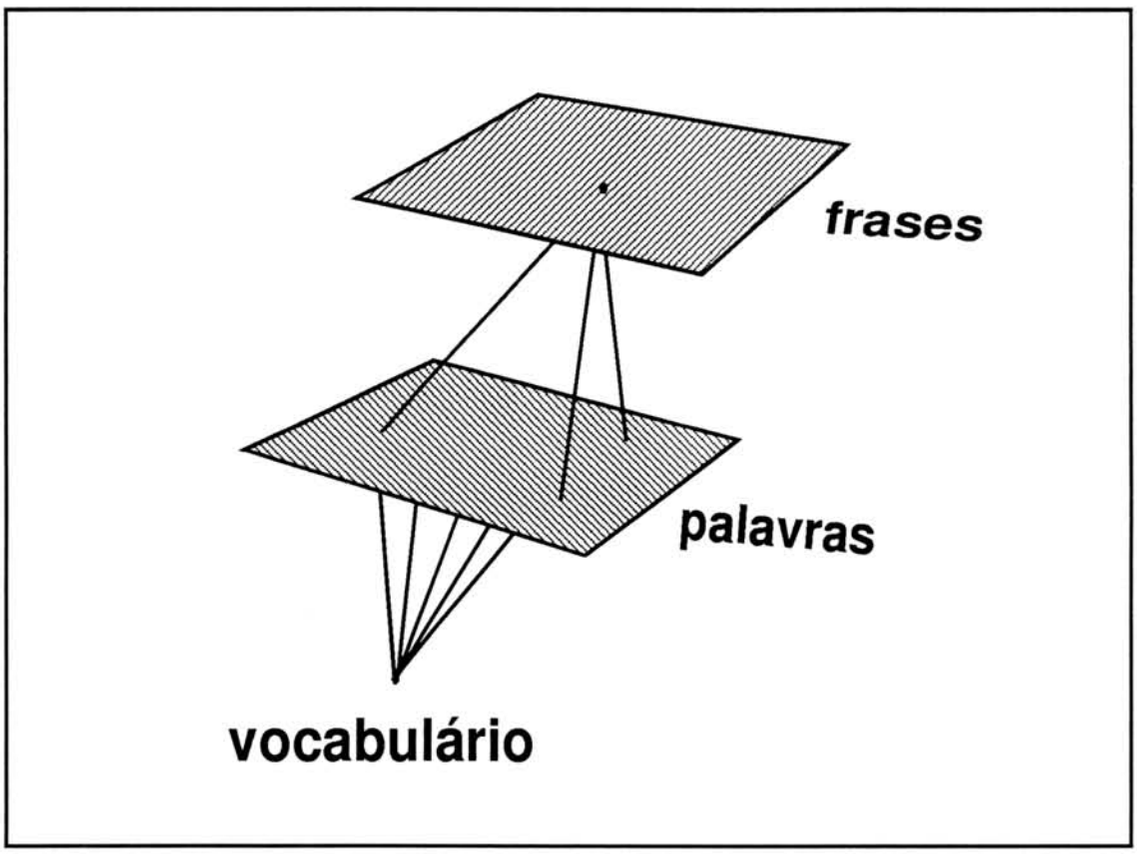


FIGURA 1.4 - **Mapas:** encadeamento de relações semânticas.

1.5 Guia para Leitura

Este capítulo introdutório teve como objetivo colocar o leitor dentro da visão do autor sobre a relação entre a semântica do indivíduo e as possibilidades de reconhecimento da linguagem no computador. Sem essas premissas acredita-se que este trabalho seria de difícil compreensão dada a pouca pesquisa existente que una as RNAs com o reconhecimento da língua natural. O que favorece a confusão nesta área específica é o fato de que muitas pesquisas sobre a linguagem serem voltadas ao acesso de banco de dados, visando a composição pergunta-resposta. Ao contrário, o que pretende-se aqui é a simples facilitação do acesso ao computador ou quaisquer outras máquinas. Aparentemente isso implica apenas na criação de uma interface mais sofisticada, o que é um equívoco. Se for analisado o sistema sensorial do ser humano ver-se-á que ele é muito mais do que uma interface, é uma

forma de auto-organização do organismo como um todo. Sem sentidos, o homem não subsiste. Portanto, o que se objetiva aqui é um primeiro passo na direção de tornar o computador numa máquina auto-organizável.

Os capítulos que seguem darão ao leitor uma visão para compreeder o processo de criação do protótipo baseado no MPL. No capítulo 2 será feita uma distinção entre os níveis simbólico e o subsimbólico dentro da IA, para que se possa saber qual a melhor área de atuação para cada um desses paradigmas. No capítulo 3 será detalhada a utilização do paradigma subsimbólico (RNAs) para o reconhecimento da linguagem e as teorias e sistemas mais conhecidos na área. Isso dará suporte para que se defina o protótipo no capítulo 4 com todos os pormenores de implementação, e no capítulo 5 será visto o funcionamento do protótipo e seus resultados. Por fim, no capítulo 6 serão apresentadas as conclusões sobre o protótipo e futuros aperfeiçoamentos.

2 O NíVEL SIMBÓLICO E O NíVEL SUBSIMBÓLICO

Analisando de forma global o funcionamento perceptivo, cognitivo e (por que não?) emotivo, observa-se a possibilidade de analisar o comportamento humano de diversos ângulos: funcionamento de redes de neurônios, variação de fluxos energéticos, construção de operações lógicas, etc. Desses pontos de vista, pode-se salientar dois que foram alvo de estudos na área de Computação:

1. a construção do raciocínio, com a formulação das operações lógicas envolvidas neste processo;
2. o comportamento do sistema neurológico, principalmente do cérebro, através da tranferência de cargas elétricas do ponto de vista físico e estatístico.

Esses dois pontos de vista foram alvo do segmento da Ciência da Computação ao qual chama-se de *Inteligência Artificial* (IA). Este termo foi utilizado devido à intenção de se reproduzir ou simular no computador aspectos da inteligência humana. Desta intenção, com o passar do tempo destacaram-se as duas abordagens citadas: uma voltada para as operações de raciocínio, denominada aqui de *Paradigma Simbólico ou Inteligência Artificial Tradicional*, e outra visando o funcionamento físico do cérebro, denominada de *Paradigma Subsimbólico ou Redes Neurais Artificiais* (RNAs). Apesar de ambos serem complementares, o desenvolvimento das pesquisas não convergiu, por muitas décadas, para este fim. Atualmente já são conhecidos muitos sistemas - chamados *Sistemas Híbridos* - que permitem a união destes dois paradigmas, como mostra a figura 2.1.

Uma vez que o presente trabalho não foi desenvolvido na linha do Paradigma Simbólico, e sim puramente na linha das Redes Neurais Artificiais, será

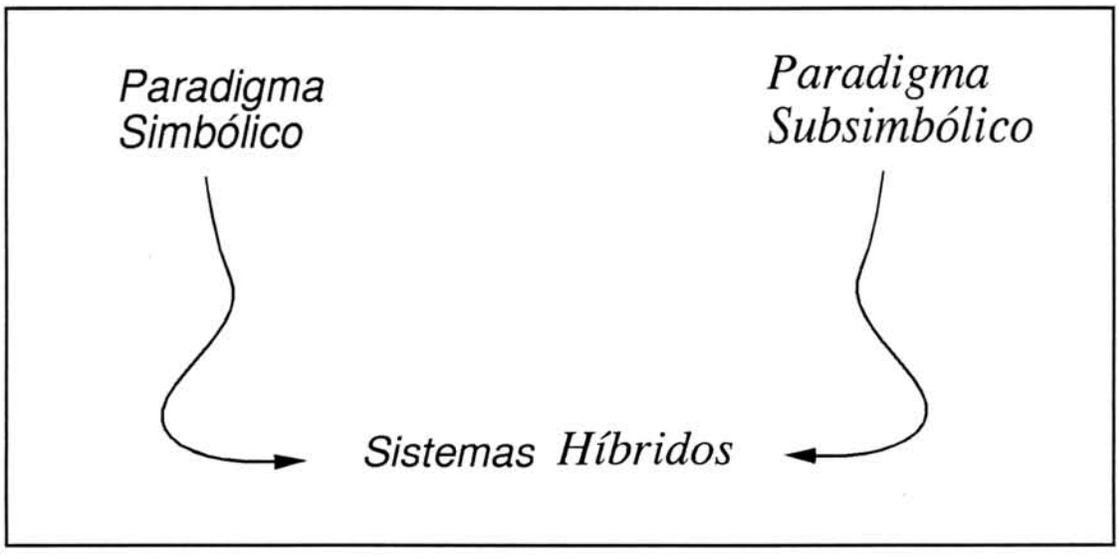


FIGURA 2.1 - **Paradigmas:** o simbólico e o subsimbólico unem suas vantagens nos sistemas híbridos.

feita uma primeira distinção entre os paradigmas e num capítulo posterior serão detalhadas as características das RNAs.

Para melhor compreensão da distinção entre os paradigmas, a seguir serão colocados breves históricos de ambos, na esperança de dar uma sólida base na justificativa do porquê da escolha do paradigma Subsimbólico para o presente trabalho.

2.1 Paradigma Simbólico

Segundo Jackson [JAC 86], as metodologias da IA começaram após a Segunda Guerra Mundial com a produção de programas de jogos e quebra-cabeças. Uma vez sendo baseados na busca de soluções para as jogadas, estes programas de jogos realizavam uma pesquisa heurística em espaços de estados para que se efetuasse uma jogada, por exemplo, de xadrez.

Durante os anos 50, esse conceito foi expandido para a realização de prova de teoremas, permitindo a formação clara de axiomas (afirmações tidas como verdadeiras) e suas combinações. Para a prova de um teorema, deve haver as afirmações

iniciais e as finais, que validarão a prova. Entre esses dois pontos, deve ser construído o processo de apresentação da solução.

Em 1963, Feigenbaum e Feldman publicaram o livro *Computers and Thought* [KLA 86], onde organizaram o conhecimento adquirido a partir das experiências e o denominaram de *Período Clássico da Inteligência Artificial* [JAC 86]. Este Período teve como mais importantes descobertas:

1. a princípio, problemas de qualquer tipo poderiam ser reduzidos a meros problemas de busca, e daí formalizados como tendo um estado inicial, um estado final, e um conjunto de regras que permitam a passagem de um estado a outro;

2. o conjunto de regras deve ser estabelecido por uma dada representação do conhecimento que permita a resolução do problema.

Jackson, na linha de Feigenbaum e Feldman, coloca o período entre os meados dos anos 60 e meados dos anos 70 como sendo o *Período Romântico* da IA. Essa afirmação tem como objetivo expressar a utopia, vivida no período, de poder tornar o computador uma máquina capaz de compreender a linguagem humana. Obviamente que houve uma decepção ao enfrentarem o problema da representação do conhecimento embutida na concepção das frases da linguagem. E esse foi o ponto para a volta à realidade, quando se descobriu que a compreensão da linguagem por um computador não era possível.

Marcando a preocupação desta época com a relação *homem-máquina*, Klahr e Waterman [KLA 86] salientam a preocupação existente sobre os ambientes operacionais voltados ao usuário, dando início às interfaces interativas. Estas interfaces tinham como características permitir a entrada de dados com caracteres manuscritos e dar uma resposta rápida ao usuário - características estas bem mais realistas do que as outras concepções que buscavam a utilização da linguagem humana.

O período seguinte a esse - chamado por Jackson de *Período Moderno* - vai de meados dos anos 70 até hoje. Esta foi uma fase de realismo e auto-crítica, com a revisão de diversos estudos e técnicas. Isso se deu devido à supervalorização de áreas do conhecimento humano e subestimação de habilidades, tais como o reconhecimento de padrões e correção de erros.

Assim, foi feita a descoberta de que é mais importante a identificação da representação do conhecimento do que as formas de inferência e valoração passíveis de serem feitas sobre este conhecimento [JAC 86]. E foi da modularização /estruturação do conhecimento que surgiram os chamados *Sistemas Especialistas*.

Sistemas especialistas são compostos de uma *base de conhecimentos*, onde se localiza a representação de determinado conhecimento, e de uma *máquina de inferência*, a qual é a responsável pela execução do raciocínio. Estas duas partes são construídas independentemente uma da outra, podendo a base de conhecimentos ser alterada sem influenciar a máquina de inferência e vice-versa [JAC 86].

Este é o estado atual do paradigma simbólico, que atua na produção de sistemas especialistas para auxílio a profissionais de diversas áreas. Mas a superação desta dimensão só se deu a partir da junção de características simbólicas com as do paradigma subsimbólico, formando os chamados *Sistemas Híbridos*, que serão abordados no final deste capítulo.

2.2 Paradigma Subsimbólico

Numa época não muito distante do início da concepção do paradigma simbólico, a estruturação subsimbólica teve seus primeiros fundamentos na concepção de um modelo matemático para o funcionamento do neurônio biológico. A identificação do funcionamento do neurônio data de 1890, quando foi publicado um trabalho do psicólogo norte-americano William James [JAM 1890], mas a modela-

gem matemática foi fruto do trabalho de Warren McCulloch e Walter Pitts em 1943 [MCW 43] [HEC 90] [EBE 90] [HER 91]. Detalhes deste modelo matemático foram discutidos em trabalho anterior [MÜL 93], portanto serão daqui omitidos.

Outro personagem importante para a fundamentação às RNAs foi Donald Hebb, em sua publicação de 1949 [HEB 49]. Nesta, ele fez uma análise não do modelo de um neurônio apenas, mas da modelagem do relacionamento entre eles, permitindo a formação de regras de aprendizado. Foi a partir daí que surgiu o termo *conexionismo* utilizado até hoje [HEC 90] [EBE 90].

A implementação prática das RNAs teve seu início com Frank Rosenblatt em 1958, quando testou seu modelo de rede neural, chamado **Perceptron** [ROS 58]. Este modelo foi acompanhado pela implementação em hardware do primeiro neurocomputador, o *Mark I Perceptron*, também desenvolvido por Rosenblatt e sua equipe [HEC 90].

Paralelamente ao trabalho de Rosenblatt, Bernard Widrow e Marcian Hoff tiveram publicado, em 1960, um artigo apresentando seu modelo, denominado **ADALINE** (Adaptive Linear Element) [WID 60]. Este modelo tem uma vantagem sobre o Perceptron: possui uma regra de aprendizado mais eficiente [HEC 90] [EBE 90].

Assim como no histórico do paradigma simbólico, aqui também houve uma fase de euforia, com a previsão de absurdos como a possibilidade de criação de um cérebro artificial [HEC 90]. Isso auxiliou a intenção dos simpatizantes do paradigma simbólico de atrair pesquisadores para seu campo. Isso se concretizou com a publicação, em 1969, do livro *Perceptrons* [MIN 69], onde Minsky e Papert criticam o modelo de Rosenblatt e o apresentam como *sem valor científico*, além de provarem que o modelo não consegue aprender uma função lógica simples como a XOR [HEC 90] [EBE 90] [HER 91].

A repercussão das afirmações de Minsky e Papert foram fortes o suficiente para abalar a credibilidade das pesquisas em RNAs. Os anos 70 foram, portanto, os piores para os pesquisadores das RNAs.

Apesar de não haver credibilidade, a pesquisa neste campo não só não parou como também teve novas e grandes descobertas. Em outras palavras, foi um dos períodos mais férteis para o paradigma subsimbólico.

Como grandes pesquisadores deste período, sobressaem-se, com trabalhos similares, Teuvo Kohonen, com sua estrutura de *memória associativa* [KOH 72], e James Anderson, com seu conceito de *memória interativa* [AND 68]. Pode-se citar ainda Stephen Grossberg com o modelo *ART* (Adaptive Resonance Theory) [GRO 69] e Kunihiko Fukushima, com o *Cognitron* [FUK 75]. E ainda: Shunichi Amari [AMA 72], David Willshaw [WIL 76], entre outros [HEC 90] [EBE 90] [HER 91].

O ressurgimento da credibilidade no paradigma subsimbólico iniciou quando John Hopfield, um físico respeitado na comunidade internacional, teve em 1982 publicado um artigo que apresentava um novo modelo de RNA (que leva hoje seu nome) e reproduzia os conceitos das RNAs dentro de uma nova perspectiva.

A consagração do paradigma subsimbólico deu-se quando uma equipe de 16 pesquisadores publicaram em 1986 o livro *Parallel Distributed Processing* [RUM 86], apresentando de uma maneira clara e prática os fundamentos e aplicações das RNAs. Na liderança desta equipe estavam James McClelland e David Rumelhart, os criadores do modelo *Backpropagation*, que utilizou-se das concepções do Perceptron/Adaline para compor uma modelagem robusta, suportando o aprendizado de diversos tipos de padrões [EBE 90].

Após 1987, começam a proliferar-se as pesquisas em RNAs nas universidades do mundo inteiro. Atualmente são dezenas de variações de modelos de RNAs que surgem em congressos internacionais todo ano. As aplicações destes modelos

são as mais diversas possíveis, indo desde o primordial reconhecimento de caracteres, passando por reconhecimento musical, de voz, de sinais, programação automática de controle de processos, processamento de linguagem natural, previsão meteorológica, até a síntese de novas proteínas. Como se pode observar, a disseminação das RNAs tornou-se uma necessidade para a evolução da IA e da própria Computação, na ampliação progressiva de seus benefícios enquanto Ciência.

2.3 Sistemas Híbridos

Pelo que se pode perceber, as filosofias de ambos os paradigmas, simbólico e subsimbólico, têm características distintas decorrentes de evoluções peculiares e sem intersecções em pesquisas. Enquanto as representações simbólicas, utilizadas no paradigma simbólico, são independentes, discretas e dependentes de uma forma de composição, as representações distribuídas são mutuamente reconstituíveis (holográficas - parte de uma representação pode reconstituir outra), podem ser sobrepostas e possuem similaridades para conceitos similares [MII93].

Como conseqüência, por um lado o paradigma simbólico possui a vantagem de permitir uma especificação clara das regras de inferência, por outro o paradigma subsimbólico possui uma capacidade de generalização, sendo tolerante a erros e permitindo o aprendizado. Nesses aspectos apresentados, ambos os paradigmas são complementares e há casos que necessitam deste tipo de abordagem [EBE 90] [MII93].

Por outro lado, há casos mal analisados em que a utilização de ambas as abordagens tornam-se um trabalho redundante e equivalente ao da implementação de apenas uma delas [EBE 90].

No presente trabalho, será analisado, no capítulo 3, alguns modelos alternativos para implementação de modelos de processamento de linguagem, o que esclarecerá ainda mais as razões da utilização do paradigma subsimbólico.

3 AS REDES NEURAIS ARTIFICIAIS E O RECONHECIMENTO DA LINGUAGEM

Segundo Miikkulainen [MII93], não há muitos modelos de processamento distribuído e paralelo (RNAs) para compreensão da linguagem natural. Há algumas razões para este fenômeno. A pesquisa de redes neurais têm se concentrado no estudo de seu próprio funcionamento, esquecendo-se de esclarecer a relação existente entre este e os fenômenos cognitivos de alto nível. De fato, pode-se observar que são poucos os pesquisadores que se preocupam em definir o uso de modelos de RNAs para aplicações de caráter simbólico (uso de inferências). Na maioria das vezes simplesmente são definidos os objetivos e motivações de alto nível das pesquisas, sem uma relação de implementação entre estes e as RNAs.

Talvez isso tenha resultado dos ataques sofridos ao embasamento dos primeiros modelos, como visto na seção 2.2, dando origem a uma extensiva análise das capacidades das RNAs. Isso dificultou o desenvolvimento de sistemas neurais que se aproximassem da descrição real de padrões, necessitando, em alguns modelos, de poderosos sistemas de pré-processamento de padrões.

Em conseqüência disso, as tarefas cognitivas de baixo nível utilizadas nas RNAs têm sido desenvolvidas dentro de um domínio próprio, visando apenas a demonstração de técnicas específicas, sem a proposição de compreensão de um dado problema de alto nível [MII93].

A visão científica estanque que as RNAs sofreram (e ainda sofrem) geram uma preocupação, que é também um desafio: como proporcionar, ao mesmo tempo, técnicas sobre os modelos que auxiliam tanto na modelagem de padrões quanto na resposta ao usuário?

Os trabalhos realizados em RNAs que visam o processamento de linguagem natural necessariamente buscam respostas a esse vazio deixado na pesquisa. É devido a este lapso que o presente trabalho foi produzido.

Sendo, portanto, ainda falha a pesquisa que busca a relação entre a cognição de alto e baixo níveis, cabe aqui o desenvolvimento e análise desta relação.

Seguindo o objetivo deste trabalho, serão analisados, dentro da perpectiva apresentada, alguns modelos de sistemas neurais que visam o processamento de linguagem natural. Pode-se distinguir os modelos mais importantes em algumas categorias básicas: *sistemas para processamento de sentenças*; *sistemas para processamento de scripts*; e *sistemas linguísticos*.

3.1 Sistemas para processamento de sentenças

Estes sistemas são, basicamente, ferramentas voltadas ao reconhecimento e classificação de frases, simples ou compostas (veja figura 3.1). Esta classificação tem, em geral, o objetivo de consultar algum banco de conhecimento, formado por um conjunto de frases aprendidas. Geralmente as frases são ensinadas visando a resposta a determinadas perguntas, formando um sistema de consulta por linguagem natural.

Um dos primeiros sistemas desenvolvidos dentro dessa categoria foi o concebido por McClelland e Kawamoto [MCC 86a]. Este sistema consiste numa rede Perceptron de duas camadas, para a qual é ensinado um padrão de entrada que contém a montagem sintática das palavras, as quais têm em si características semânticas codificadas à mão. Um determinado conjunto dessas características (sentença) é associado a um padrão de saída, o qual é um código representando o contexto semântico.

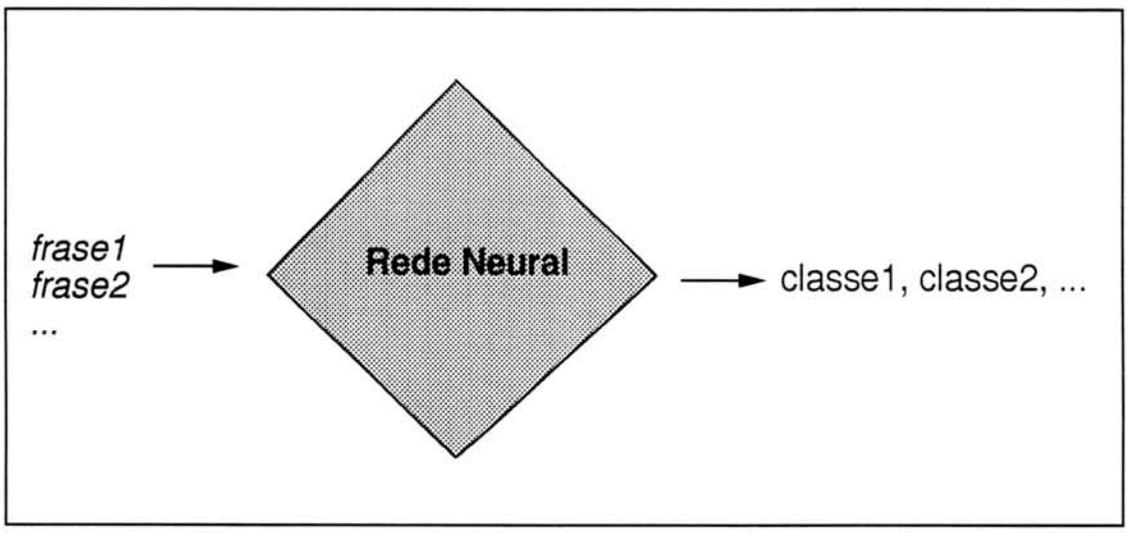


FIGURA 3.1 - **Processamento de Sentenças:** reconhecimento e classificação.

Esse sistema foi concebido para demonstrar a possibilidade de se produzir regras semânticas automaticamente através das RNAs, baseando-se apenas na sintaxe e no contexto semântico de sentenças.

No ano de 1990 houveram diversas publicações sobre o reconhecimento de sentenças. Uma das publicações a se ressaltar é de St. John e McClelland [STJ 90], a qual apresenta um sistema composto de duas redes backpropagation concatenadas, sendo que a saída da primeira é a entrada da segunda rede. A função da primeira rede é aprender a formação das frases, para que na sua saída resulte um tipo padrão de sentença, ao qual os autores denominam *sentence gestalt*. Esta sentença faz uma composição com uma pergunta, formando a entrada da segunda rede, cuja função é responder questões.

Desta forma tem-se um sistema que, através da codificação à mão de características semânticas utilizadas na identificação das características das sentenças, permite a associação das questões aos contextos semânticos das sentenças. O resultado deste processamento é um sistema ágil de consulta a determinado banco de conhecimento.

Outro sistema de 1990 é o de Jain e Waibel [JAI 90], que, a exemplo do sistema de St. John e McClelland, possui um encadeamento de três redes em backpropagation que permitem a distinção de três níveis para o reconhecimento de uma sentença:

1. nível de frase;
2. nível de oração (sentença de mais de uma frase);
3. nível de estrutura de oração e relacionamentos entre as frases da oração.

O objetivo deste sistema é permitir a identificação das relações semânticas dos componentes de uma sentença. Essa identificação permite inclusive o reconhecimento para palavras fora da ordem treinada, uma vez que nos níveis 2 e 3 é feito reconhecimento de várias combinações possíveis dos componentes da oração.

Ainda em 1990, Kohonen [KOH 90] apresentou um sistema simples para reconhecimento das características semânticas de palavras de frases compostas apenas por três elementos. Utilizando um mapa de características auto-organizáveis (SOM), e um sistema de pré-processamento de frases de três elementos, foi possível a distinção entre as características semânticas, sem a necessidade da realização de uma atribuição manual de códigos, permitindo um treinamento automático na relação frase-reconhecimento.

A idéia que permitiu a Kohonen conceber este sistema foi a de que a relação semântica estava contida na coerência da construção da frase. Daí concebe-se que, para uma dada sintaxe de palavras, existe uma determinada relação semântica relativa a esta organização. O sistema apresentado neste trabalho utilizou-se desta concepção para sua construção.

Finalmente, Miikkulainen [MII 90] [MII93], criticando a falha de outros sistemas por não permitirem a composição dos sistemas neurais a plataformas de alto nível cognitivo, propôs o sistema FGREP. Este sistema consiste numa rede

backpropagation de três camadas que aprende uma composição de palavras que formam uma sentença pré-definida. As palavras possuem uma parte que é pré-codificada e outra que vai sendo codificada a medida em que vão sendo aprendidas as palavras. Desta forma, é formada uma codificação de acordo com as características das demais palavras da frase. Resumindo, este sistema é um codificador semântico de palavras, com a vantagem de pressupor a construção modular de frases, podendo ser composto numa plataforma de nível simbólico.

Percebe-se que a tendência do desenvolvimento de sistemas para processamento de sentenças tem sido a de compor o paradigma subsimbólico, aqui identificado pela diferenciação semântica das palavras e frases, com o paradigma simbólico, ou seja, o contexto prático de utilização: consulta a banco de dados, processamento de textos, etc.

3.2 Sistemas para Processamento de *scripts*

A Teoria dos *Scripts* busca a organização de uma seqüência de eventos esteriotipados para melhor representação do conhecimento. Computacionalmente, os *scripts* são semelhantes aos *frames* utilizados na IA tradicional, os quais são baseados na idéia da Teoria dos Esquemas [MII93]. O conceito de esquema foi abordado em trabalho anterior ([MÜL 95]). Para melhor compreensão do esquema cognitivo no presente trabalho, veja seção 1.2.1.

Utilizando modelos do paradigma subsimbólico, percebe-se que, para uma dada estória (conjunto de *scripts*), é possível o reconhecimento do *script* adequado, sua instanciação e inferência, tudo automaticamente, de forma inerente aos modelos [MII93] (veja figura 3.2). Neste sentido, foi importante a contribuição de Harris e Elman [HAR 89], com a demonstração de como podem ser representados relacionamentos de estórias baseadas em *scripts*. Esta idéia foi testada num sistema que foi treinado para prever qual a próxima palavra de um dado *script*. Estudando o

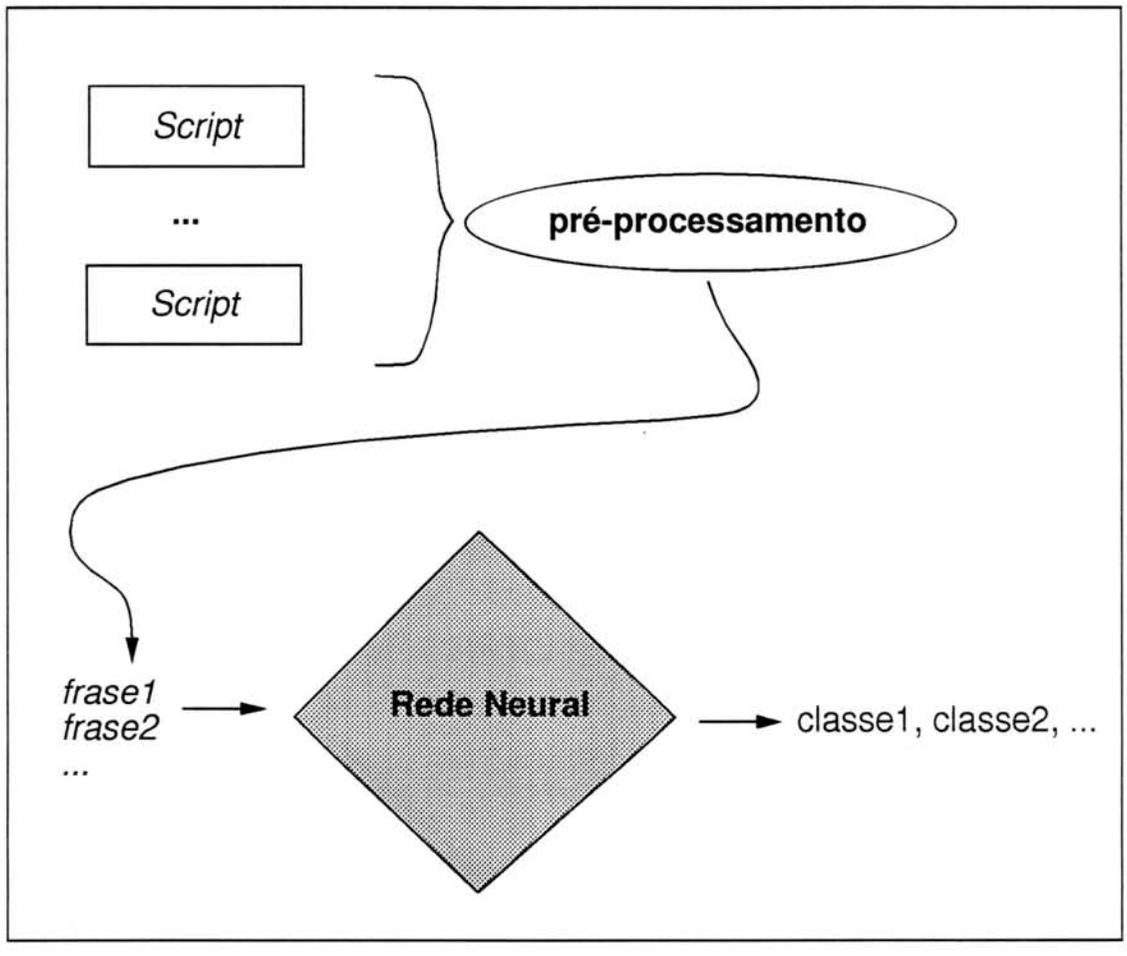


FIGURA 3.2 - **Processamento de Scripts:** possibilidade de manipulação simbólica.

comportamento deste sistema, verificou-se que a camada escondida (*hidden*) representava os relacionamentos entre regras da estória, e estes relacionamentos eram construídos a partir apenas dos dados de entrada. Assim, demonstrou-se que é possível a representação de *scripts* através do paradigma subsimbólico.

Outro sistema que baseou-se em *scripts*, foi um desenvolvido a partir do modelo de *sentence gestalt* de St. John e McClelland, apresentado na seção 3.1. St. John realizou uma extensão do modelo para a produção de *story gestalt*, de funcionamento semelhante ao sistema de processamento de sentenças [STJ 92]. Neste novo modelo, ao invés de serem aprendidas frases, são apresentadas afirmações ao sistema, que servirão como base para as respostas das perguntas que serão posteriormente feitas ao sistema. As vantagens deste sistema é que são aprendidas as

características dos esteriótipos da estória, mas tem como desvantagem a dificuldade de aplicação de novos scripts.

3.3 Sistemas linguísticos

Há diversos modelos que utilizam conceitos lingüísticos, inclusive para demonstração de conceitos lingüísticos através de RNAs (figura 3.3). Um dos trabalhos pioneiros foi o produzido por Rumelhart e McClelland, que utilizava uma rede back-propagation para o aprendizado de verbos da língua inglesa no passado [RUM 86a]. Eles demonstraram que o aprendizado dos verbos na rede neural assemelha-se ao comportamento de uma criança frente ao aprendizado dos mesmos verbos, buscando mostrar a viabilidade deste recurso computacional para representação da linguagem.

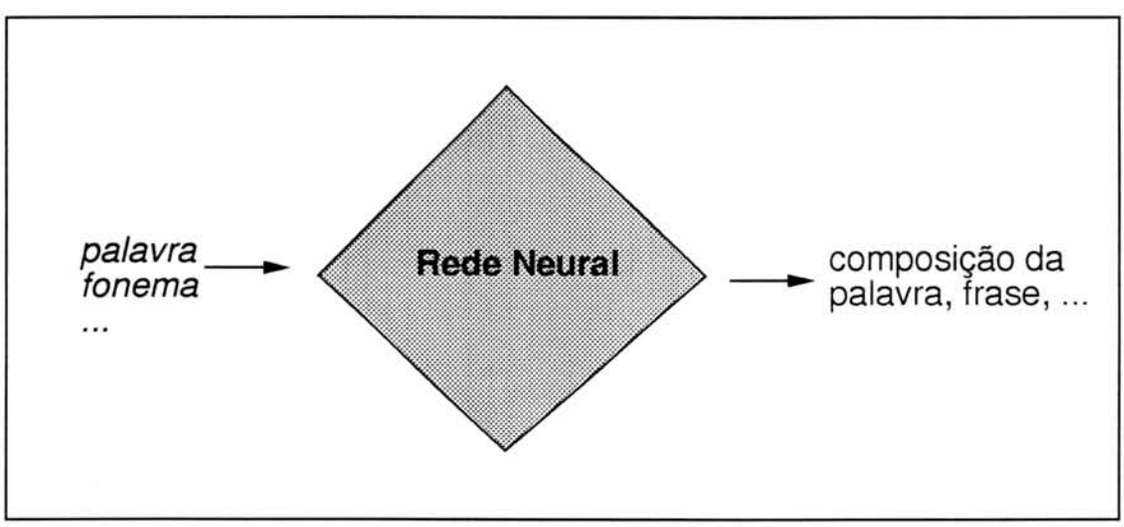


FIGURA 3.3 - **Sistemas Lingüísticos:** geração de linguagem.

Outro sistema que obteve relativo impacto na comunidade científica foi o de Sejnowski e Rosenberg, conhecido como NETtalk [SEJ 87] [HEC 90] [SMO 90]. Este sistema basicamente fazia a pronúncia da leitura de um texto em inglês. Ele foi baseado no sistema DECtalk (da DEC Corporation), que consistia num sintetizador de voz e um gerador de som. Este sistema especialista possuía centenas de regras desenvolvidas por lingüistas durante décadas, e convertia uma cadeia de caracteres

em comandos para geração de som. Sejnowski e Rosenberg tentaram (e conseguiram) a reprodução do comportamento do DECtalk numa rede neural em backpropagation, ao que chamaram de NETtalk. Para o treinamento da rede neural foi utilizado o sistema especialista para a apresentação dos fonemas. Ao final, observou-se que ambos sistemas eram funcionalmente idênticos. Isso gerou questionamentos de onde, na rede neural, estariam armazenadas as centenas de regras utilizadas no sistema especialista. Não foi possível chegar a uma resposta clara, mas a hipótese mais plausível é a de que a representação da informação está dentro da rede na forma de transformações matemáticas. Acredita-se que uma resposta clara somente se dará quando forem compreendidos os mecanismos de comportamento dos neurônios do cérebro, possibilitando a demonstração de seu funcionamento.

Outro aspecto lingüístico explorado em RNAs é o aprendizado de gramáticas. Servan-Schreiber, Cleeremans e McClelland apresentaram uma rede neural que aprendem uma gramática simples de estados finitos [SER 89] [SER 91]. O sistema era treinado para indicar qual os próximos elementos possíveis de uma seqüência, com sucesso. Por outro lado, para gramáticas livres de contexto, o comportamento não foi preciso, aproximando-se do comportamento humano, como constatou Elman [ELM 91].

3.4 Apresentação de Um Novo Modelo

Após analisado o contexto dos modelos para processamento de linguagem envolvendo as RNAs, é possível localizar o presente trabalho dentro da tecnologia existente. O protótipo que será apresentado no capítulo 4, localiza-se dentro dos Sistemas de Processamento de Sentenças, e, dentro da conceituação do MPL apresentado anteriormente, procura inserir-se como um elemento de um sistema maior, de processamento de linguagem, e não apenas de sentenças.

Nessa linha, utilizando e aperfeiçoando os recursos dos Sistemas de Processamento de Sentenças, chegou-se a um protótipo para identificação semântica de frases mediante o estabelecimento de duas redes neurais em seqüência, uma para a identificação de palavras, e outra para as frases formadas com as palavras da primeira rede. Este sistema será apresentado em detalhes no capítulo 4 e, espera-se que seja uma contribuição no sentido de aperfeiçoar a relação homem-máquina e incentivar a pesquisa em torno da pesquisa computacional relacionada aos estados mentais do homem.

4 UM PROTÓTIPO PARA O RECONHECIMENTO DA LINGUAGEM

4.1 O Processamento de Sentenças e O Processamento de Linguagem

O protótipo aqui apresentado foi proposto a partir do MPL apresentado na seção 1.2.1, que apresenta uma modelagem baseada na comunicação humana, na sua linguagem. O MPL apresenta recursos de entrada de sinais do meio, processamento destes sinais de entrada visando sua classificação, e por fim a verificação destes sinais classificados dentro de uma memória associativa de contexto, para previsão do próximo evento.

Assim, o que está se propondo aqui é uma simplificação computacional do processamento dos sinais externos (como ocorre no homem), obtidos na forma de palavras e codificação numérica, para sua classificação dentro de um contexto de aplicação. *Não será aqui verificada nenhuma forma de entrada do meio, seja ela sonora, visual, etc. Também não será elaborada nenhuma proposta de previsão do próximo evento segundo contextos de aplicação. Estas abordagens fogem ao escopo deste trabalho.*

Aqui é apresentado um modelo para o processamento de sentenças, provenientes de um contexto específico, cujas classes de palavras e das próprias frases são identificadas no aprendizado e são refletidas no reconhecimento. Desta forma, pretende-se um sistema classificatório de sinais do meio (classes de palavras e frases) atuando em conjunto com palavras e frases propriamente ditas, de forma a permitir o reconhecimento semântico de frases. Em outras palavras, *o que se quer é a possibilidade de alteração de palavras e do números de palavras nas frases, permitindo ainda assim sua interpretação.*

O processamento de sentenças aqui desenvolvido segue uma representação subsimbólica, como discutido no capítulo 2. Isto significa, em contrapartida, que não haverá uma análise lingüística de sentenças, mas sim a utilização de um modelo matemático para a identificação das constantes estatísticas de um conjunto de sentenças. O que se quer é a identificação de sentenças através de suas características invariantes, ou seja, fazer o reconhecimento de uma dada sentença, mesmo com alterações de palavras e número de palavras que a compõe.

Para efetivar a identificação de sentenças através de suas características invariantes, torna-se necessária a utilização de modelos de RNAs. O modelo neural escolhido foi o denominado *Mapa Auto-Organizável* (Self-Organizing Map), proposto por Teuvo Kohonen em 1984 [KOH 84], dadas suas peculiaridades para organização de características de padrões.

4.2 O Modelo Neural Escolhido

4.2.1 Representação Topográfica

O modelo do Mapa Auto-Organizável possui duas *camadas* de neurônios, onde a primeira possui tantos neurônios quanto for o tamanho do vetor de valores de entradas, e a segunda possui o número de neurônios e a forma - retangular ou hexagonal - dependente da aplicação a que se destina a rede. A denominação de *mapa* ocorre devido ao mapeamento de características que é feito nesta segunda camada da rede. Ela forma um plano bidimensional no qual os padrões semelhantes agrupam-se em neurônios próximos, permitindo, assim, a classificação por zonas do mapa, conforme vê-se na figura 4.1.

Quanto às conexões, todos os neurônios da primeira camada são conectados aos da segunda, uma vez que o vetor de entradas (neurônios da primeira camada)

são comparados aos pesos de cada neurônio da segunda camada. O modelo ainda compreende interações laterais nos neurônios desta segunda camada, que podem ser entendidas como uma influência da saída de um dado neurônio no ajuste dos pesos dos seus vizinhos. Por esta características, dá-se o nome de *conexão competitiva* a este tipo de organização de rede.

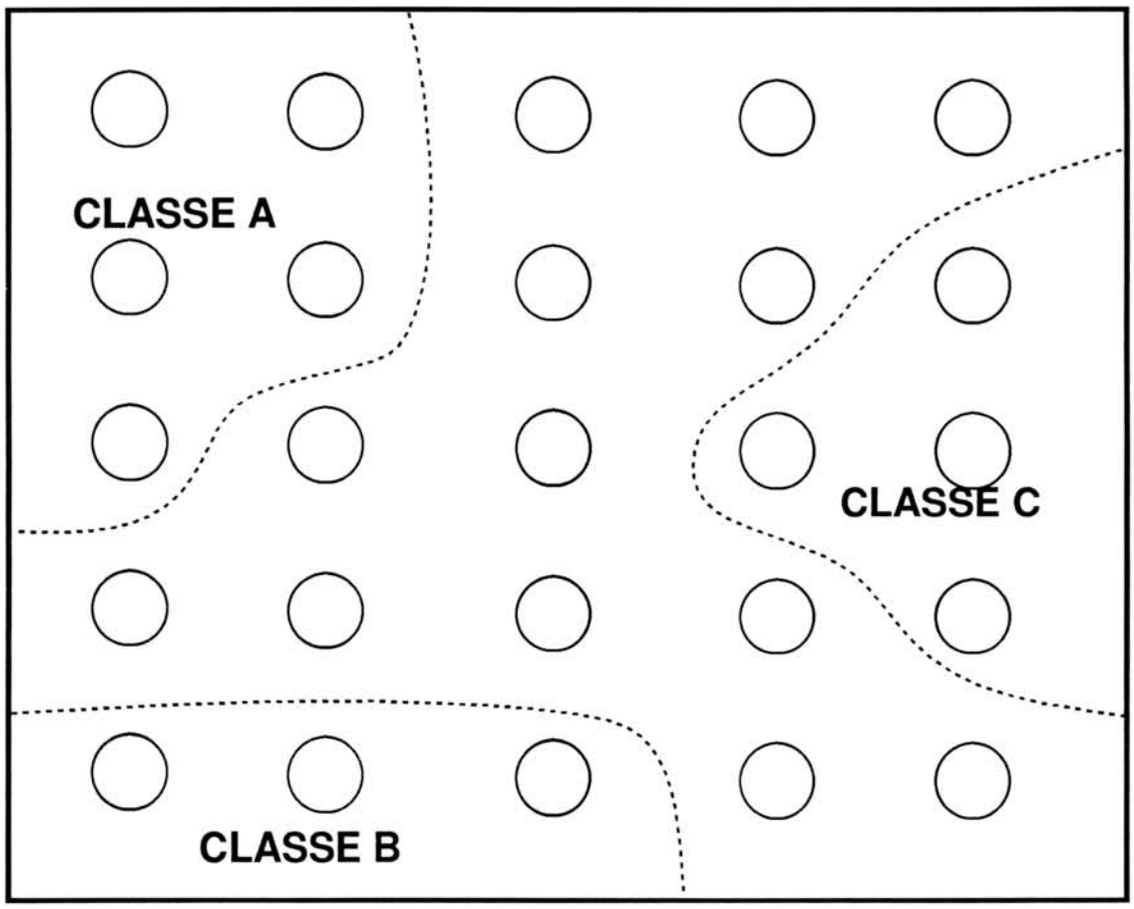


FIGURA 4.1 - **Classificação:** distinção de grupos de neurônios proporcionada pelo Mapa Auto-Organizável.

4.2.2 Motivação Matemática

O Mapa Auto-Organizável de Kohonen tem o princípio de representar a freqüência de ativação média de cada neurônio do mapa, mais a interferência dos sinais superpostos dos demais neurônios vizinhos. Este princípio pode ser descrito pelo sistema não-linear de equações:

$$f_r = \mu \left(\sum_l w_{rl}\, v_l + \sum_{r'} g_{rr'} f_{r'} - \theta \right), \qquad (1)$$

onde f_r é a ativação média para o neurônio r, que envolve a ativação do neurônio, através da soma ponderada do vetor de pesos w_{rl} e vetor de entradas v_l menos o limiar de ativação θ, somados à soma ponderada dos sinais de ativação do neurônio vizinho $f_{r'}$ e a intensidade da sinapse $g_{rr'}$, entre o neurônio atual r e o vizinho r'. O símbolo μ representa uma função do tipo *sigmoid*, que normaliza a resposta de ativação para o intervalo (0;1).

A resposta de ativação do neurônio ($\sum_l w_{rl}\, v_l$) é utilizada em grande parte dos modelos de RNAs, porém Kohonen insere em seu modelo a interferência dos neurônios vizinhos na resposta de ativação de um dado neurônio. Isso se dá através do controle da função $g_{rr'}$ que é definida segundo alguma distribuição probabilística, dado que haja sinapses inibitórias ($g_{rr'} < 0$) para grandes distâncias entre os neurônios e excitatórias ($g_{rr'} > 0$) para pequenas distâncias. Esta distância seria a distância vetorial entre dois pontos, ou seja, $\|r - r'\|$.

O raciocínio anterior permite uma simplificação, afirmando-se que *o neurônio vizinho de maior ativação será aquele com menor distância vetorial entre os pesos e as entradas*, ou seja,

$$\|v - w_{r'}\| = \min_r \|v - w_r\|. \qquad (2)$$

Este cálculo permite uma aproximação do valor de r', necessário para chegarmos às soluções da equação (1).

Como para satisfazer-se o cálculo de (1) necessita-se ainda a determinação da intensidade dos sinais dos neurônios vizinhos, ou seja, a forma de distribuição de seus sinais, é definida uma função de distribuição $h_{rr'}$ entre dois neurônios. Seguindo

o raciocínio da função $g_{rr'}$ de (1), a distribuição deve realizar uma sinapse excitatória para os neurônios da vizinhança do neurônio calculado, e inibitória para os neurônios mais distantes, conforme visualiza-se na figura 4.2.

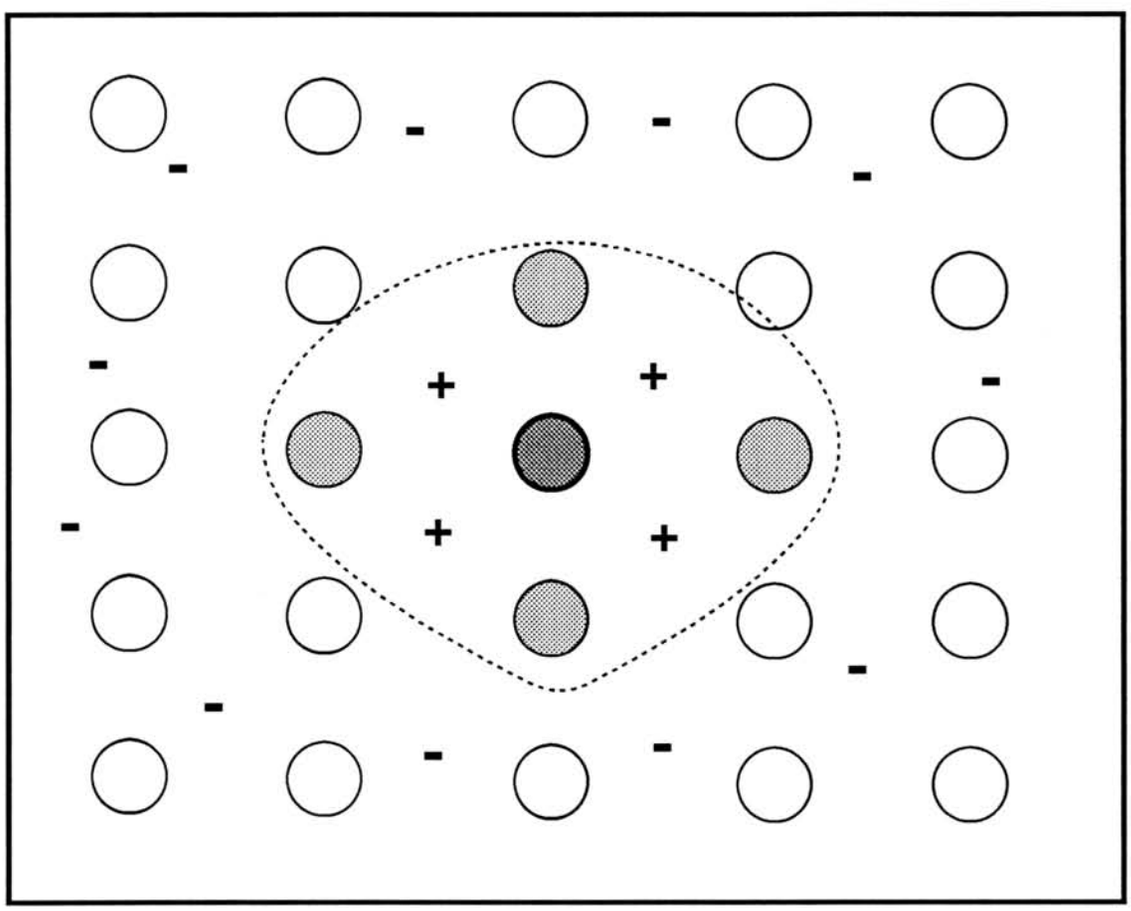


FIGURA 4.2 - **Vizinhança:** os neurônios vizinhos recebem sinapses excitatórias e os mais distantes recebem sinais inibitórios.

Essa forma de distribuição é possível de ser compreendida como uma troca de pesos sinápticos, na forma:

$$w_r^t = w_r^{t-1} + \epsilon\, h_{rr'}(v - w_r^{t-1}), \qquad (3)$$

onde a atualização dos pesos do neurônio r, dado por w_r^t, é resultante da aplicação de um valor de adaptação ϵ juntamente com o resultado da função de distribuição $h_{rr'}$ sobre a diferença dos valores de entrada e de pesos $v - w_r^{t-1}$. A

função de adaptação para o valor ϵ deve ser decrescente com o tempo t, permitindo a adaptação gradativa à redução da diferença $v - w_r^{t-1}$.

Em outras palavras, esse processo permite a adaptação dos pesos às características do padrão de entrada da rede. Isso se dá através da alteração gradativa dos pesos de blocos de neurônios. Estes blocos são definidos através da função de distribuição $h_{rr'}$, e terão seu tamanho reduzido com o aumento do tempo t. Isso porque cada saída de neurônio tenderá a responder por um dado vetor de entradas. Obviamente não se quer uma rede neural para o aprendizado de apenas um padrão, um vetor de entradas, o que se quer é a representação de diversos padrões, tantos quantos forem possíveis de serem adaptados na rede. Assim, cada padrão de entrada será melhor adaptado por um determinado neurônio. A ordenação dos neurônios que dão a resposta dá-se de forma a representar as proximidades numéricas dos padrões aprendidos ($v - w_r^{t-1}$), e, por conseqüência, a proximidade entre os neurônios com uma resposta semelhante ($\|r - r'\|$).

Como resultado desse processo, os padrões com codificação semelhante serão representados por neurônios próximos, dado um plano bidimensional. Isso permite a criação de zonas de características que são descritas pelos próprios padrões de entrada. Compreende-se, então, um processo de auto-organização de características, ou seja, o modelo que Kohonen propôs é um sistema de auto-classificação de padrões, que registra as características com as quais foram codificados os vetores de entrada. Por estas propriedades, diz-se que este é um modelo com *auto-aprendizado.*

4.2.3 Algoritmo Utilizado

Para se ter uma idéia mais clara da motivação matemática, a seguir é apresentado o algoritmo básico do modelo de Kohonen, com variantes propostas pelo próprio Kohonen em aplicações de mapas semânticos [KOH 90], utilizado na implementação computacional do presente trabalho.

1. Inicializar os pesos da rede com valores randômicos no intervalo [0.01;0.1].

2. Inserir o padrão de entrada.

3. Calcular as distâncias vetoriais ($\|v - w_r\|$):

$$d_l = \sum_{i=0}^{N-1} (v_l(t) - w_{rl}(t))^2$$

onde d_l é a distância entre a saída do nodo j com a entrada e N é o número de entradas.

4. Selecionar a menor distância, segundo a equação (2).

5. Atualizar os pesos, segundo a equação (3), com uma distribuição gaussiana para $h_{rr'}$:

$$h_{rr'} = exp\left(-\frac{\|r - r'\|^2}{\sigma^2}\right)$$

onde o raio σ é decrementado com o tempo:

$$\sigma(t) = \sigma_i \left(\frac{\sigma_i}{\sigma_f}\right)^{\frac{t}{t_{max}}}$$

onde σ_i é o valor inicial do raio e σ_f seu valor final.

6. Repetir a partir do passo 2, até t_{max}.

Os procedimentos apresentados formam o módulo central do protótipo computacional, onde são calculados os resultados dos mapas de palavras e de frases. Os demais módulos do protótipo são destinados ao reconhecimento de padrões e ao pré-processamento das palavras e frases.

4.3 Organização do Protótipo

Basicamente, há quatro módulos no sistema:

1. codificação de palavras;
2. codificação de frases;
3. modelo neural para aprendizado;
4. modelo neural para reconhecimento.

A seqüência de apresentação dos dados (veja figura 4.3) segue primeiramente a codificação numérica de palavras, que vai ser utilizada no aprendizado do mapa de palavras. Uma vez realizado este aprendizado, ocorre a codificação das frases, utilizada, por sua vez, no aprendizado do mapa de frases. Terminada esta etapa, o mapa de palavras já não é mais necessário, pois a fase de aprendizado já está concluída.

A segunda etapa é o reconhecimento, onde é necessária a codificação numérica da frase a reconhecer, e a realização do reconhecimento através do mapa de frases.

O pré-processamento que possibilita a codificação numérica, tanto de palavras como de frases, é essencial para a construção das relações semânticas que permitirão o reconhecimento das frases.

4.4 Pré-processamento

A preparação dos vetores de valores a ensinar a uma rede neural é um passo de fundamental importância para a constituição de uma aplicação em RNAs. Uma vez que as RNAs identificam constantes estatísticas, é necessário que os padrões

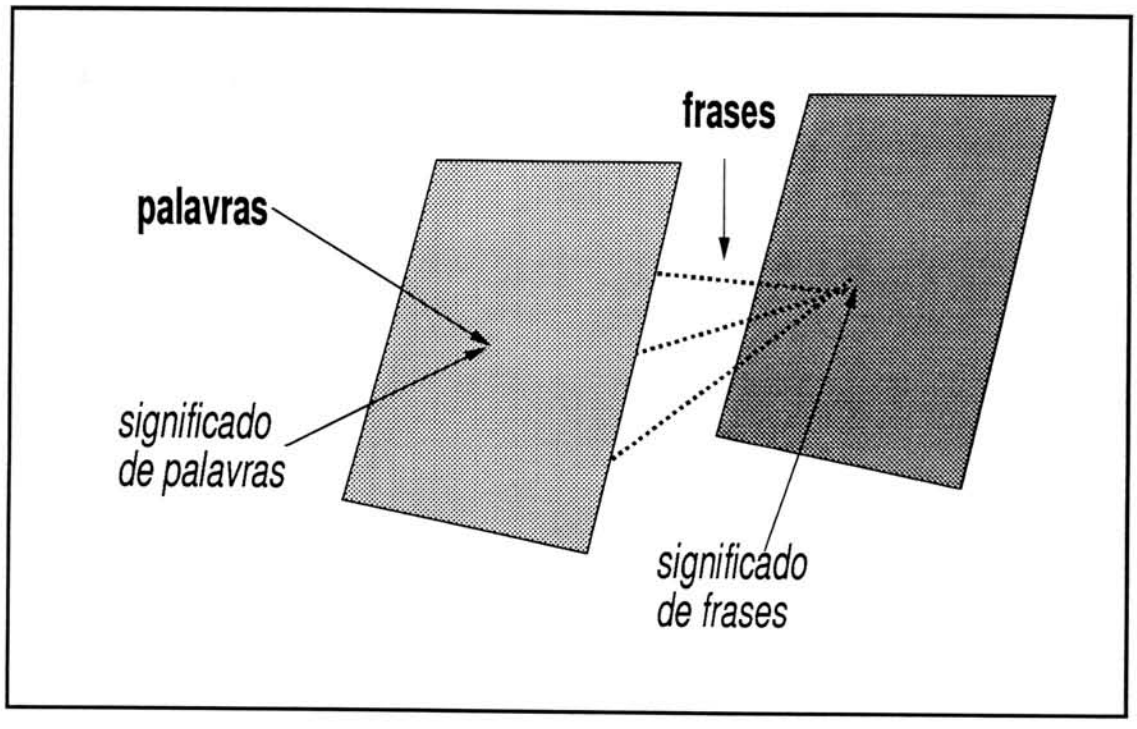


FIGURA 4.3 - **Seqüência:** a inserção de dados ocorre tanto no mapa de palavras como no mapa de frases.

apresentados tenham as características fundamentais para a generalização, que permitirá o reconhecimento de padrões não aprendidos.

4.4.1 Codificação de Palavras

O primeiro passo é a codificação das letras de cada palavra. Um vez escolhido o vocabulário para a aplicação que se pretende, passa-se à codificação. Cada uma das letras recebe um código de acordo com a estatística de seu preenchimento num *bitmap*, como utilizado em trabalhos como o de Hinton [HIN 90] e Miikkulainen [MII93]. As palavras passam a ser, portanto, um vetor de valores, sendo que estes estão situados no intervalo (0;1).

Para este trabalho, definiu-se que cada palavra teria um máximo de 16 letras, sendo, portanto, um vetor máximo de 16 valores para a definição da palavra.

O passo seguinte é unir a codificação da palavra com um outro código identificador da semântica daquela palavra (figura 4.4). Este código identificador é o elemento que irá definir a proximidade das palavras de uma certa classe no primeiro mapa, funcionando então como um endereçamento, uma **chave classificatória**. Esta chave funciona de modo a forçar a proximidade dos valores de ativação de um conjunto de neurônios, e, por conseqüência, a ter uma pequena distância entre estes neurônios, conforme analisado na seção 4.2.2.

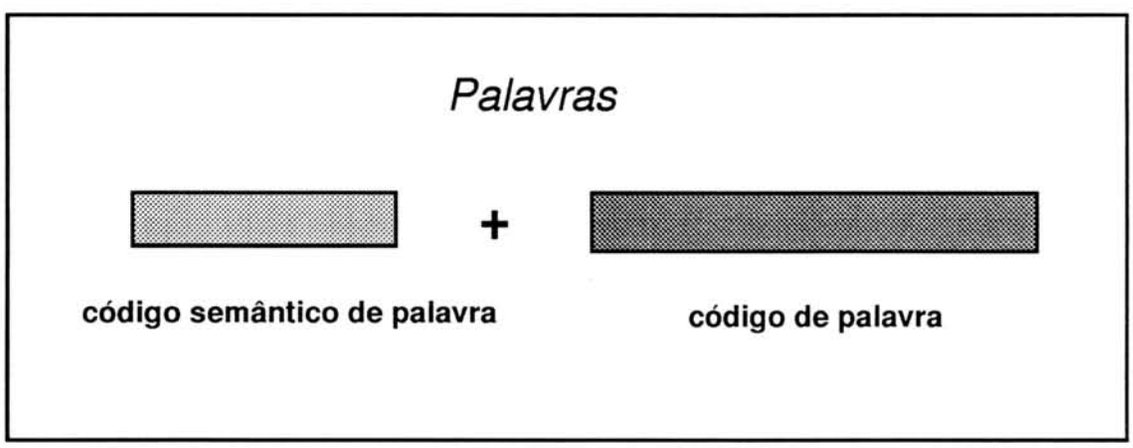


FIGURA 4.4 - **Código da Palavra:** o padrão de entrada para o mapa de palavras é a união entre o código da palavra em si e o da classe semântica.

Assim, de acordo com a aplicação desejada, o usuário seleciona até dez campos semânticos, nos quais pretende classificar seu vocabulário. O limite de dez campos é puramente para efeito de implementação, uma vez que há cinco campos de valores para a combinação de classes. As combinações desses valores são extremamente distintas, de modo a acelerar o processo de aprendizado, mas eventualmente poderiam ter valorações mais aproximadas.

Como abordado na seção 4.1, o código identificador da semântica representa sinais provenientes do meio que permitem o aprendizado do significado relativo a determinada palavra que acompanha estes sinais. O código para o aprendizado do mapa de palavras é composto, portanto, dos valores referentes à palavra e à sua classe, perfazendo, na presente implementação, a um máximo de 21 valores. Uma codificação semelhante pode ser encontrada no trabalho de Miikkulainen [MII93].

Uma vez codificadas, as palavras são ensinadas para o mapa de palavras, seguindo o algoritmo de aprendizado descrito na seção 4.2.3. Como resultado, tem-se a distribuição por áreas comuns de palavras de uma mesma classe definida.

4.4.2 Codificação de Frases

Uma vez distribuídas as palavras no primeiro mapa, as posições dos neurônios ativados por cada palavra são armazenadas para posterior consulta. Em outras palavras, o primeiro mapa serve como codificador para o mapa de frases. Uma vez extraídas as posições dos neurônios para cada palavra do vocabulário, o mapa de palavras não é mais necessário.

Mais uma vez, visando uma determinada aplicação, o usuário define as frases (combinações de palavras) e suas classes (figura 4.5). Como no pré-processamento das palavras, há a união do código das frases com o código semântico. O código das frases é formado por um par de valores - resultantes do mapa de palavras - para cada palavra da frase, tendo sido a implementação limitada em dez palavras por frase. O código semântico segue a mesma regra utilizada para as palavras.

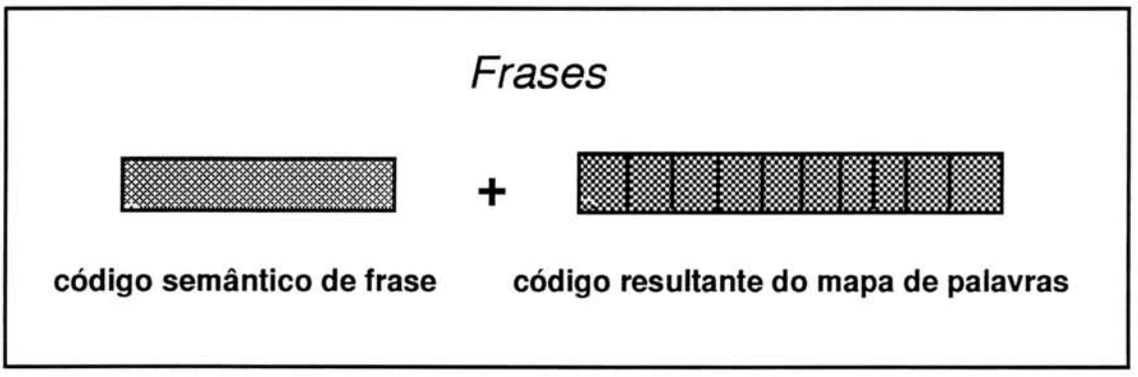


FIGURA 4.5 - **Código da Frase:** o padrão de entrada para o mapa de frases é a união entre o código da frase em si e o da classe semântica para a frase.

Uma vez feito o pré-processamento das frases, a codificação é ensinada ao segundo mapa, para o qual valem os mesmos comentários feitos para o mapa de palavras.

Após o aprendizado deste segundo conjunto de códigos, temos a formação de mapas semânticos de frases, a partir dos quais estarão formados os conjuntos passíveis de reconhecimento. No capítulo 5 são apresentados os resultados de reconhecimento de diversos conjuntos de aprendizado, que demonstram a grande utilidade e plausibilidade do sistema apresentado. Outros detalhes sobre a codificação de pré-processamento podem ser analisados na seção 2.

5 EXEMPLOS DE APLICAÇÃO

Como foi exposto na seção 4.3, o protótipo está dividido em quatro módulos, cada qual com sua função específica. São descritos neste capítulo dois exemplos de vocabulário e frases utilizando estes vocabulários, processados nos quatro módulos apresentados. Estes exemplos têm por objetivo apresentar a viabilidade da utilização do presente protótipo em futuros projetos de sistemas mais complexos, permitindo a comunicação do homem com a máquina de uma forma mais amigável, como comandos por voz, através de frases em linguagem humana.

Esta comunicação permitirá, num futuro próximo, o controle por voz de artefatos com componentes eletrônicos, tais como veículos, eletrodomésticos, brinquedos, controladoras de processos industriais, etc. Os exemplos aqui apresentados abordarão o controle da própria interface computacional contruída para teste do protótipo e o comando de um toca-discos *laser*, um eletrodoméstico comum nos dias atuais. Isso dará uma noção bem clara do funcionamento do protótipo e de seu potencial de utilização.

Para uma efetiva demonstração da capacidade de generalização e classificação do protótipo apresentado, foi construída uma interface para apresentação dos dados e também para ser ela própria objeto de teste de comandos frasais. A apresentação detalhada desta interface pode ser encontrada no apêndice 1.

Os resultados de utilização dos exemplos apresentam análises de desempenho de erro de reconhecimento semântico por alteração de palavras. Estes resultados, no entanto, são fruto de uma adequada distinção de classes semânticas pelo usuário do protótipo. O uso de abordagens incorretas pode resultar em respostas indesejadas. Isso é demonstrado posteriormente e é analisado o que se faz necessário para a obtenção de um bom desempenho, através de exemplos com alterações no projeto de definição de classes semânticas.

Antes, porém, da apresentação dos resultados, faz-se necessária a adoção de certas premissas de restrição à entrada de dados. São elas:

- *uma vez apresentado um determinado vocabulário de palavras para o aprendizado, as frases construídas deverão obrigatoriamente seguir este vocabulário;*
- *ao solicitar o reconhecimento, as palavras devem ser escritas com a grafia correta;*
- *os arquivos de entradas de dados editados pelo usuário não poderão conter erros no formato;*
- *a utilização dos módulos do protótipo possui uma ordem seqüencial de processamento, uma vez que existe a dependência dos resultados entre eles.*

Uma vez seguindo essas premissas elementares, será possível o processamento de palavras e frases através dos módulos do protótipo. A utilização dos módulos, seus parâmetros e formatos de arquivos de entrada estão detalhados no apêndice 2, e seus respectivos códigos-fonte, escritos na linguagem C, encontram-se no apêndice 3.

A seguir serão descritos dois exemplos possíveis de aplicação para o protótipo. O primeiro trata da construção de frases visando o controle de uma interface, mais exatamente aquela descrita no apêndice 1. O segundo exibe a formação de frases para o controle de um toca-discos a *laser*. A partir destes exemplos serão obtidos os dados necessários à análise do protótipo. Para uma melhor compreensão da utilidade dos exemplos, deve-se ter em mente seu uso através de uma interface de reconhecimento de voz, uma vez que não há praticidade em comandos frasais escritos.

5.1 Exemplo1: Controle de Interface

Esta interface, como comentado anteriormente, é voltada à visualização dos resultados do protótipo apresentado no presente trabalho. Para tanto, ela permite:

1. a carga de arquivos de palavras;
2. a carga de arquivos de frases de treinamento;
3. a carga de arquivos de frases de reconhecimento;
4. o treinamento do mapa de palavras;
5. o treinamento do mapa de frases;
6. a visualização do mapa de palavras;
7. a visualização do mapa de frases;
8. a visualização do reconhecimento de um conjunto de frases;
9. encerramento da utilização da interface.

Para que fosse possível o gerenciamento das funções, propôs-se o seguinte vocabulário:

```
abra
de
visualizar
mapa
palavras
ler
sair
ver
```

```
arquivo
editar
frases
verificar
interface
da
treinar
reconhecimento
```

5.1.1 Treinamento de Palavras

Uma vez que o protótipo necessita da classificação semântica dada pelo meio, as palavras foram separadas em cinco conjuntos, de acordo com a concepção da interface:

- Classe 1: os verbos *abra, visualizar, ler, ver, editar, verificar, sair, treinar.*
- Classe 2: as preposições *de, da.*
- Classe 3: as palavras indicativas de módulos de utilização *mapa, arquivo.*
- Classe 4: a indicação do aplicativo *interface.*
- Classe 5: as palavras utilizadas nos módulos *palavras, frases, reconhecimento.*

Estas palavras foram então codificadas através do programa *tranp* para serem utilizadas como padrão de entrada no programa *tremapa*, de aprendizado e construção do mapa de palavras. Os parâmetros de aprendizado utilizados foram o raio inicial $\sigma = 4.0$, o valor de adaptação $\epsilon = 0.7$ e um tempo máximo $t = 10000$, para um mapa retangular 10 x 10.

Para obtenção dos resultados, foi necessária uma nova aplicação do programa *tranp* sobre o conjunto de palavras a reconhecer, e a utilização das palavras codificadas como entrada no programa *recmapa*. O mapa gerado é apresentado na figura 5.1.

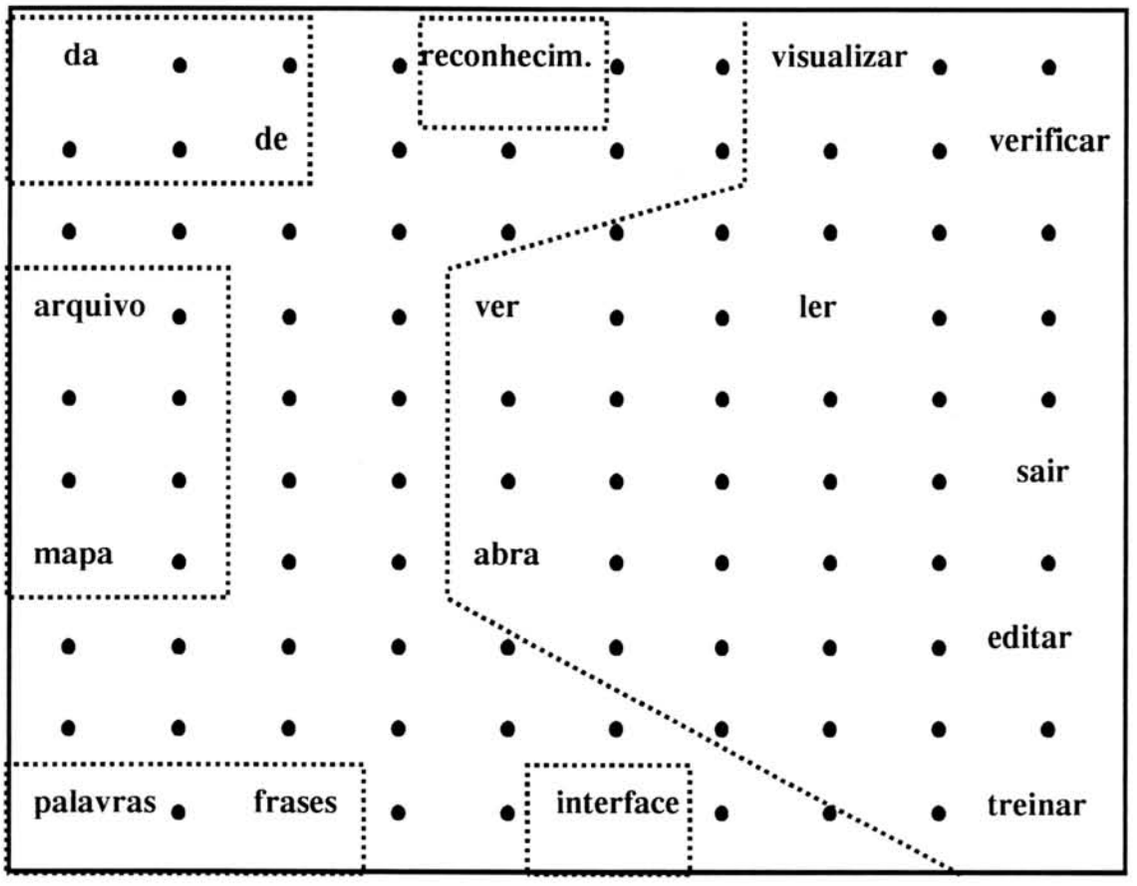


FIGURA 5.1 - **Mapa de Palavras:** Distribuição das palavras conforme sua classe.

Cabem aqui algumas observações sobre este resultado gerado, tais como:

1. a proximidade entre as palavras de uma mesma classe;
2. a distribuição coerente das classes dentro do mapa, sem sobreposições das mesmas;
3. o número de iterações (tempo) de aprendizado relativamente baixo.

5.1.2 Treinamento de Frases

O passo seguinte é a determinação das frases e suas respectivas classes semânticas. Como a organização das frases é um conjunto de dados mais complexos que as palavras, torna-se necessário sua separação por tamanho (número de palavras), criando-se um mapa para cada conjunto de frases de mesmo tamanho. Estes conjuntos foram divididos em nove classes, distribuídas em quatro mapas, respeitando o tamanho das frases. A ordem por classe é a seguinte:

- Classe 1: *abra palavras, ler arquivo de palavras, abra arquivo de palavras, editar palavras, arquivo de palavras.*
- Classe 2: *ver mapa de palavras, ver palavras, mapa de palavras, visualizar mapa de palavras, verificar palavras.*
- Classe 3: *ler frases, ler arquivo de frases, editar arquivo de frases.*
- Classe 4: *mapa de frases, verificar mapa de frases, mapa frases, abra mapa de frases, visualizar frases.*
- Classe 5: *treinar palavras, treinar mapa de palavras.*
- Classe 6: *treinar frases, treinar mapa de frases.*
- Classe 7: *ler reconhecimento, ler arquivo de reconhecimento.*
- Classe 8: *mapa reconhecimento, ver mapa de reconhecimento, mapa de reconhecimento, reconhecimento.*
- Classe 9: *sair, sair da interface.*

Assim, as frases de um mesmo conjunto foram então codificadas através do programa *tranf* para serem utilizadas como entrada no programa *tremapa*, de aprendizado do mapa de frases. Os parâmetros de aprendizado utilizados para todos os conjuntos foram raio inicial $\sigma = 4.0$, o valor de adaptação $\epsilon = 0.7$ e os demais foram definidos em função do número de componentes:

- mapa de uma palavra: tempo máximo $t = 5000$ e mapa retangular 3 x 3;
- mapa de duas palavras: tempo máximo $t = 10000$ e mapa retangular 10 x 10;
- mapa de três palavras: tempo máximo $t = 10000$ e mapa retangular 7 x 7;
- mapa de quatro palavras: tempo máximo $t = 50000$ e mapa retangular 10 x 10;

Novamente, para a obtenção dos resultados, é necessária a utilização do programa *tranf* sobre os conjunto de frases a reconhecer, e a utilização das frases codificadas como entrada no programa *recmapa*. Cabe aqui salientar que as palavras utilizadas para o reconhecimento do padrão treinado tiveram suas classes semânticas zeradas, a fim de não haver muita distorção quando do reconhecimento de frases não-treinadas. Desta forma, a codificação para o reconhecimento é distinta daquela apresentada na seção 4.4.2 - utilizada para o treinamento - uma vez que não é apresentada a definição de um número de classe semântica. Os mapas resultantes são apresentados a seguir.

Para o primeiro conjunto - de **uma palavra** - há os seguintes padrões:

```
sair
reconhecimento
```

os quais foram organizados no mapa da figura 5.2.

Para o conjunto de **duas palavras** temos os seguintes padrões:

```
ler frases
ver palavras
abra palavras
```

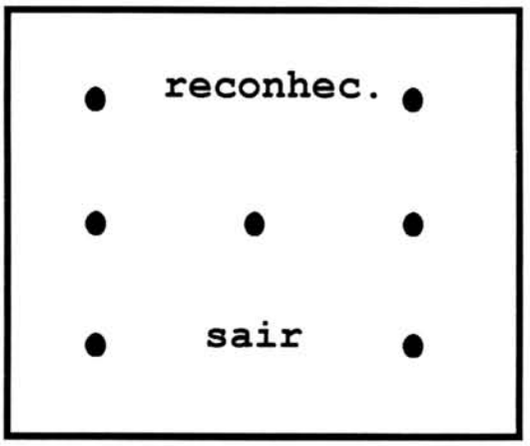


FIGURA 5.2 - **Mapa de Frases de Uma Palavra:** como frases com uma palavra não são muito usadas, aqui utilizam-se apenas dois padrões.

```
treinar palavras
verificar palavras
ler reconhecimento
mapa frases
sair interface
visualizar frases
editar palavras
mapa reconhecimento
treinar frases
```

os quais foram organizados no mapa da figura 5.3.

O conjunto de **três palavras** é formado pelos padrões:

```
mapa de frases
sair da interface
mapa de palavras
mapa de reconhecimento
arquivo de palavras
```

que estão classificados no mapa da figura 5.4.

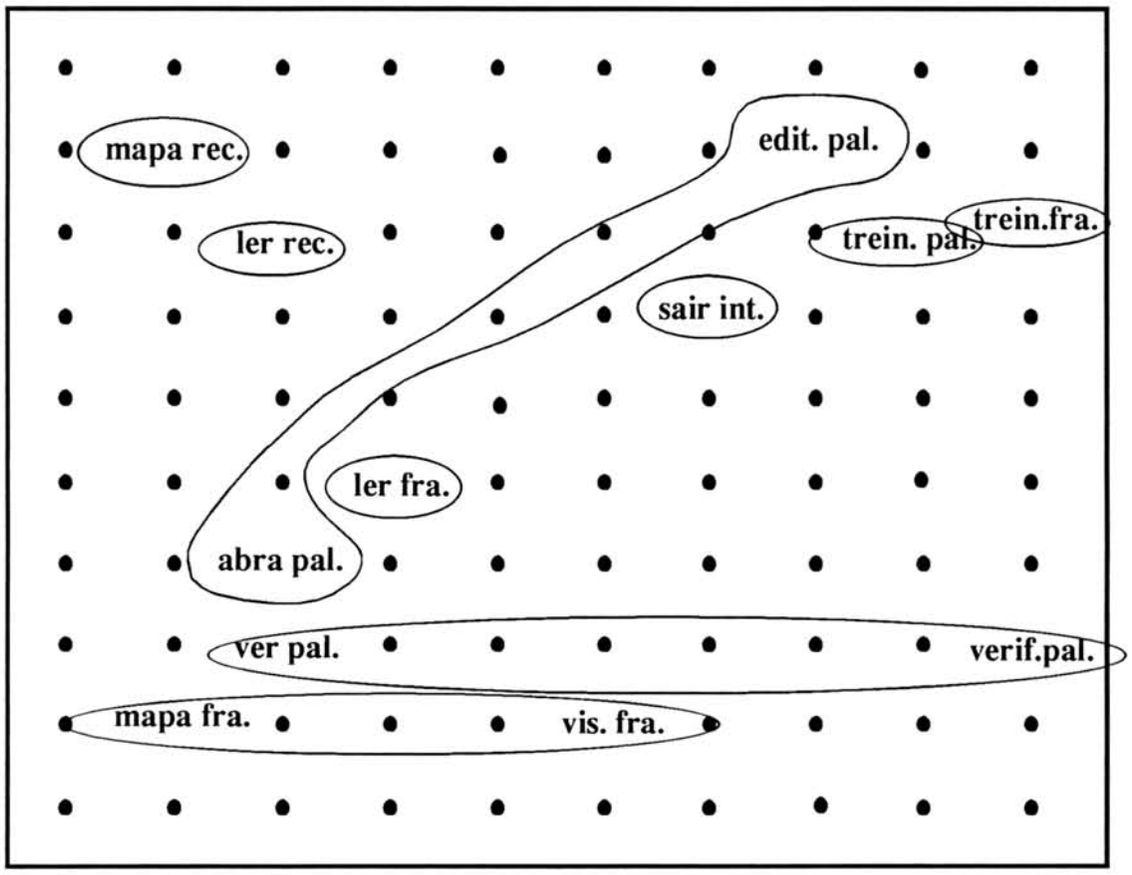


FIGURA 5.3 - **Mapa de Frases de Duas Palavras:** diversos padrões de aprendizado.

Por fim, o conjunto de **quatro palavras** é composto pelos padrões:

```
ler arquivo de palavras
ver mapa de palavras
treinar mapa de palavras
verificar mapa de frases
abra arquivo de palavras
ver mapa de reconhecimento
abra mapa de frases
ler arquivo de frases
visualizar mapa de palavras
editar arquivo de frases
ler arquivo de reconhecimento
```

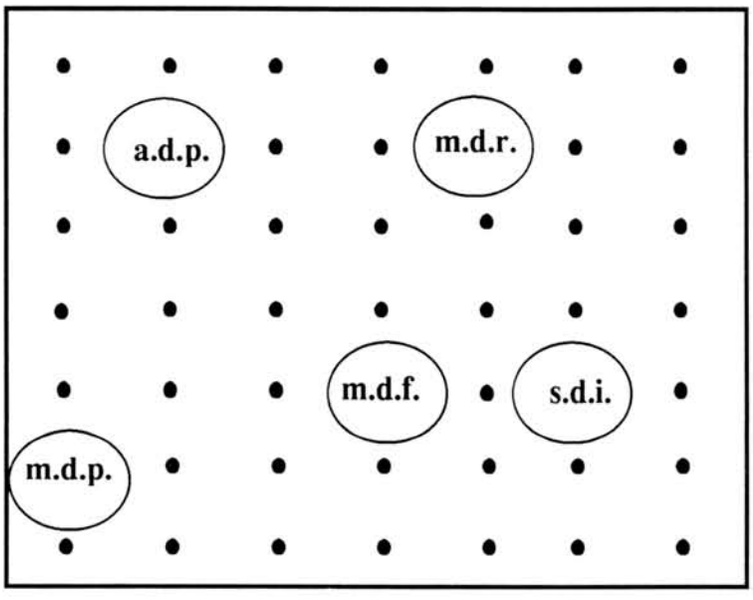


FIGURA 5.4 - **Mapa de Frases de Três Palavras:** número de padrões mais restrito.

`treinar mapa de frases`

que estão classificados no mapa da figura 5.5.

Há algumas observações importantes sobre os resultados gerados:

1. nem sempre há uma proximidade entre as palavras de uma mesma classe, podendo ser mais próxima a outra classe;
2. existe a organização de grupos de frases com mesmas palavras que possuem mesma posição dentro da frase;
3. a proximidade das frases é proporcional à proximidade das palavras no mapa de palavras;
4. há a tendência a sobreposições de classes;
5. o número de classes dentro de um mapa determina a capacidade de classificação (ver mapa de frases de duas e quatro palavras);
6. o número de iterações (tempo) de aprendizado é baixo como no aprendizado de palavras.

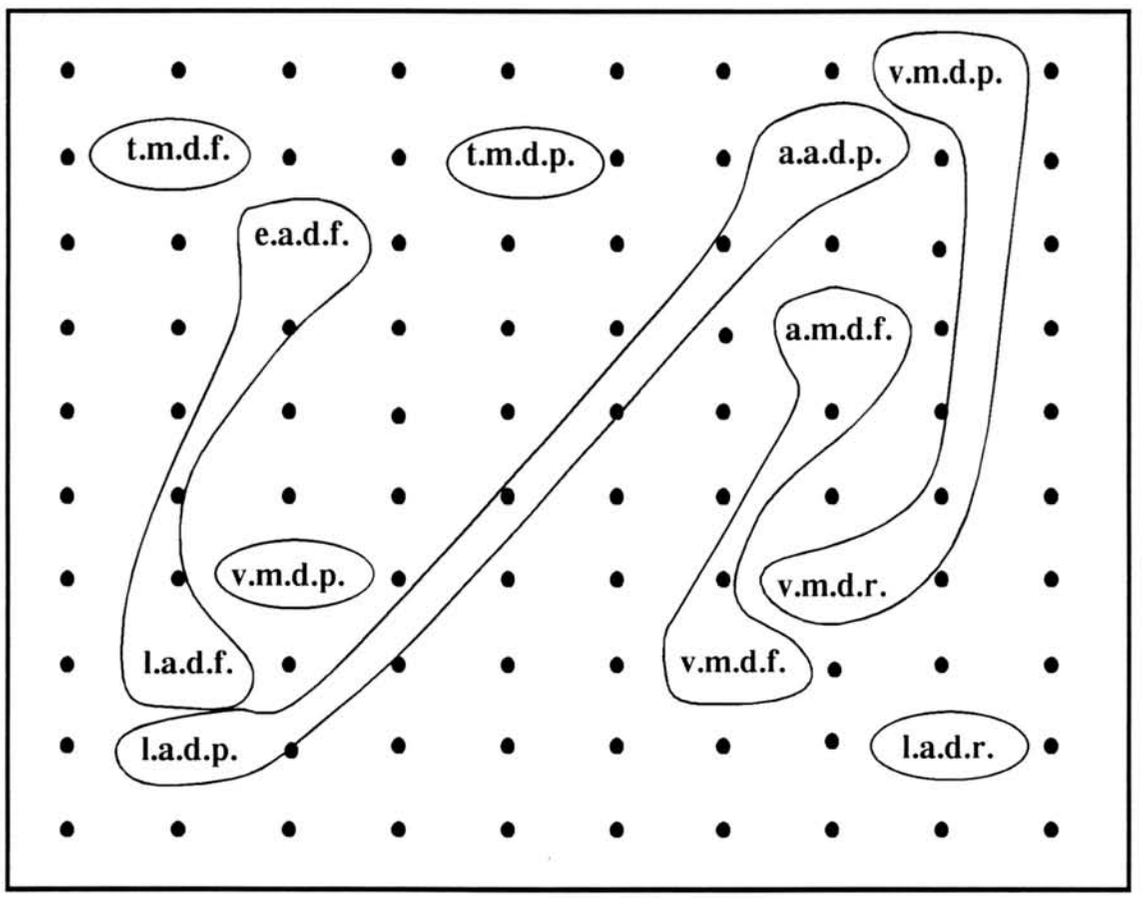


FIGURA 5.5 - **Mapa de Frases de Quatro Palavras:** diversos padrões de aprendizado.

5.1.3 Reconhecimento de Frases Não Treinadas

Para testar a robustez e capacidade de generalização do protótipo, torna-se necessária a validação através de exemplos não-treinados que permitam a constatação de um percentual de erro no reconhecimento desses exemplos. Deste modo, apresenta-se um conjunto de frases fora do escopo de treinamento que determinará o *grau de acerto*. Isso torna-se necessário uma vez que os padrões treinados não possuem erro de reconhecimento, afora os casos de troca de caracteres das palavras treinadas, o que não é aceito pelo protótipo.

O *grau de acerto* pode ser definido como sendo o quanto um padrão de teste se aproxima do padrão treinado desejado, ou da área em torno deste padrão. Aqui tem-se uma diferenciação da codificação apresentada na seção 4.4.2, uma vez

que, ao reconhecer uma frase, não se sabe a qual classe semântica pertence o padrão apresentado. Portanto, como foi visualizado na figura 4.5, o campo semântico não contém nenhum valor válido, sendo então zerado. Obviamente que a nível de codificação haverão valores válidos que busquem a aproximação da frase não-treinada com o conjunto de treinamento. Para mais detalhes, veja o apêndice 2.

O reconhecimento realizado é, portanto, apenas uma aproximação, uma vez que não é possível ter-se o valor de classe semântica. Finalmente, cabe aqui a colocação de mais algumas observações:

1. na ausência de um conjunto de palavras treinado, o *esquema de decisão* toma por base a proximidade das palavras no mapa de palavras;
2. o reconhecimento sempre toma por base frases com palavras comuns.

O item 1 indica que o conjunto da classificação realizada pelas redes de Kohonen e da aproximação semântica realizada através do reconhecimento formam um *esquema de decisão.* Em outras palavras, o processamento das características de frases-padrão permite um posterior reconhecimento de frases não treinadas, através da verificação de equivalência da codificação do padrão de treinamento com o padrão de reconhecimento. Quanto mais semelhanças as frases tiverem entre si, mais alta será a probabilidade de um reconhecimento preciso. Por semelhança de frase entenda-se palavras em comum e ordenadas de uma mesma forma.

Tem-se, então, o conjunto de frases a seguir:

- **para frases de duas palavras:**

```
ler palavras
abra frases
abra reconhecimento
verificar reconhecimento
```

```
visualizar palavras
treinar reconhecimento
mapa palavras
sair frases
editar reconhecimento
ver frases
```

A visualização do reconhecimento deste conjunto pode ser feita comparando-se o mapa original (figura 5.3) com a figura 5.6. As frases de teste estão em letras maiúsculas.

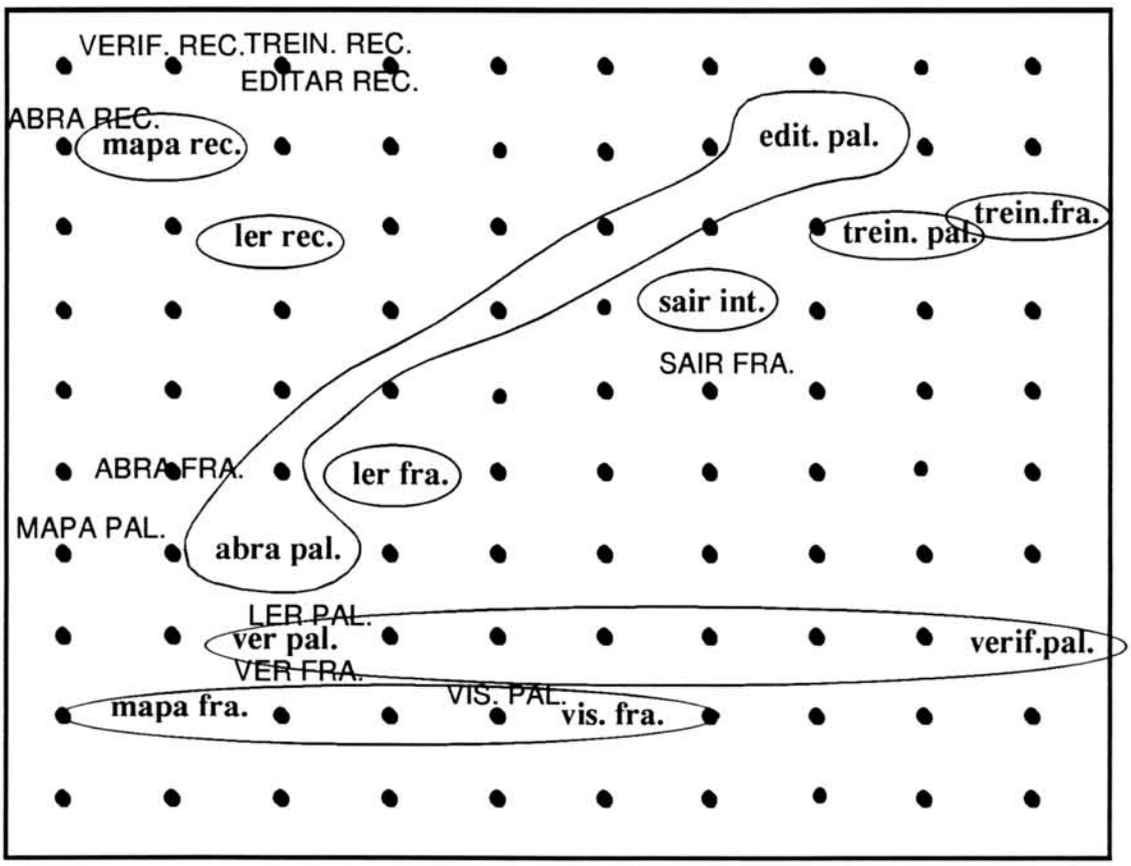


FIGURA 5.6 - **Comparação:** frases não aprendidas têm pouca possibilidade de reconhecimento.

- **para frases de três palavras:**

```
mapa de interface
```

```
sair de frases
arquivo de frases
arquivo de reconhecimento
```

Mais uma vez, é feita a comparação entre o mapa treinado e as palavras a reconhecer na figura 5.7.

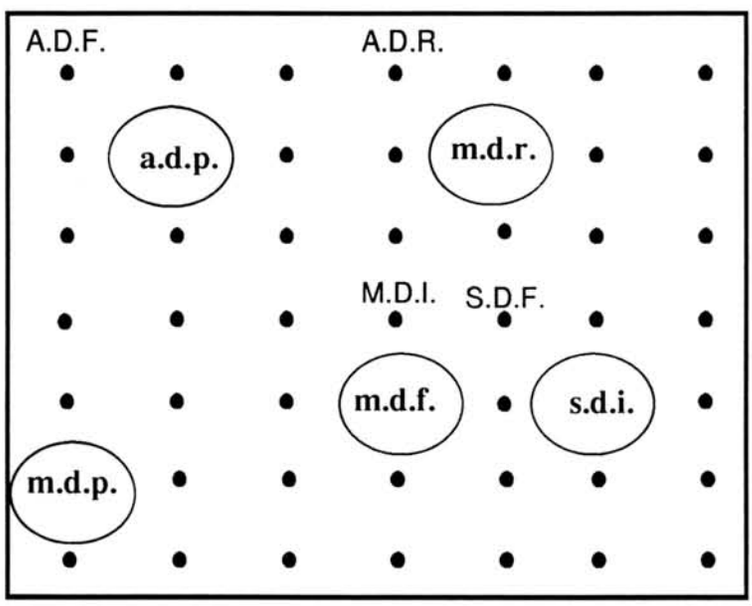


FIGURA 5.7 - **Frases Sem Sentido:** as frases de teste não possuem relações claras com aquelas treinadas.

- **para frases de quatro palavras:**

```
ler mapa de palavras
ver arquivo de frases
ver mapa de frases
treinar arquivo de palavras
treinar mapa de reconhecimento
verificar mapa de palavras
verificar arquivo de frases
abra arquivo de reconhecimento
abra mapa de palavras
```

```
visualizar arquivo de reconhecimento
visualizar mapa de frases
editar arquivo de palavras
editar mapa de reconhecimento
```

Por fim, as frases de quatro palavras são comparadas na figura 5.8.

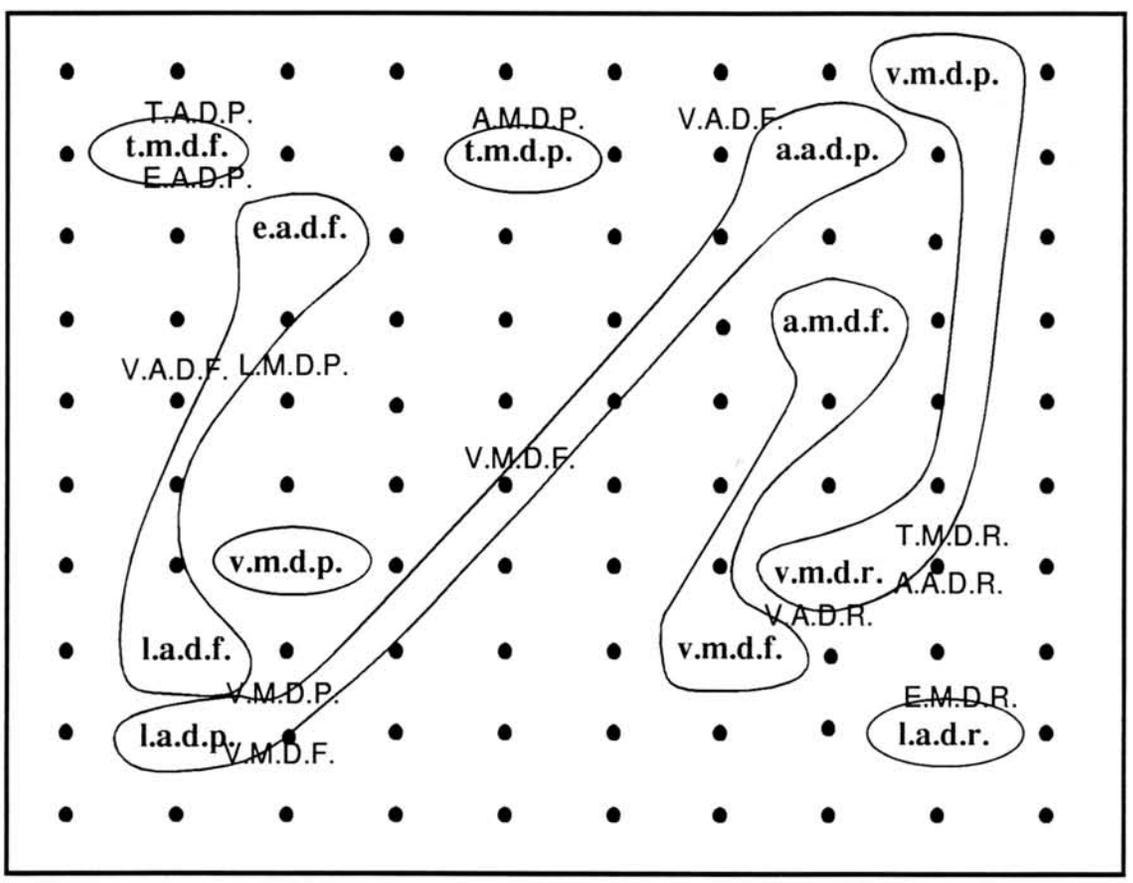


FIGURA 5.8 - **Frases Perdidas:** há frases de teste bem orientadas e outras fora de contexto semântico.

Após obtidos os mapas, deseja-se identificar o grau de acerto que eles obtiveram. Para tanto, foi explicitado na tabela 5.1 a quantidade de frases treinadas, testadas, sua coerência frasal, ou seja, a porcentagem de acerto quanto à correspondência das palavras nas frases, e sua coerência semântica, que nada mais é do que o grau de acerto semântico, o quão próximo está a frase-teste da frase treinada no que se refere a seu significado.

TABELA 5.1 - **Comparação:** quantificação do grau de acerto.

pals. na frase	treinadas	testadas	c. frasal (%)	c. semântica (%)
2	12	10	100	50
3	5	4	100	100
4	12	13	53,8	30,7

Com base nos resultados obtidos, pode-se observar:

1. existe aproximação entre frases com determinado número de palavras comuns;
2. a composição (ordem de disposição) das palavras na frase treinada e na frase de teste é determinante para um bom resultado;
3. *a aproximação entre as frases treinadas e de teste têm como fator principal sua proximidade no mapa de palavras.*

5.1.4 Alterações para Otimização

Para otimizar o desempenho do índice de coerência semântica (veja tabela 5.1), torna-se necessária a alteração no projeto desde a classificação das palavras. Isso demonstra que não há trivialidade na definição das palavras constituintes do campo semântico, necessitando, portanto, uma análise mais aprofundada da dinâmica treinamento-reconhecimento.

Como a relação de proximidade entre as palavras do mapa de palavras é um elemento básico para a definição correta das relações em nível mais alto, foi necessária a alteração da classe que envolvia os verbos (veja figura 5.1) para melhorar o desempenho do reconhecimento semântico. Conceberam-se, então, novas classes por afinidade das palavras. A classe 1 é pulverizada nas classes 1, 7, 8, 9, além da palavra *reconhecimento* da classe 5, ter sido isolada na classe 6:

- Classe 1: *abra, ler, editar* para indicar leitura de arquivo;

- Classe 7: *visualizar, verificar, ver* que indicam a visualização de mapa;
- Classe 8: *treinar* para acionar o treinamento de padrões;
- Classe 9: *sair* para encerrar a interface;
- Classe 6: *reconhecimento* para reconhecer um padrão desconhecido.

Além das classes, foi alterado também o valor do tempo máximo de treinamento, que passou de 10000 para 100000. Os demais parâmetros permaneceram inalterados. O mapa resultante do treinamento está na figura 5.9.

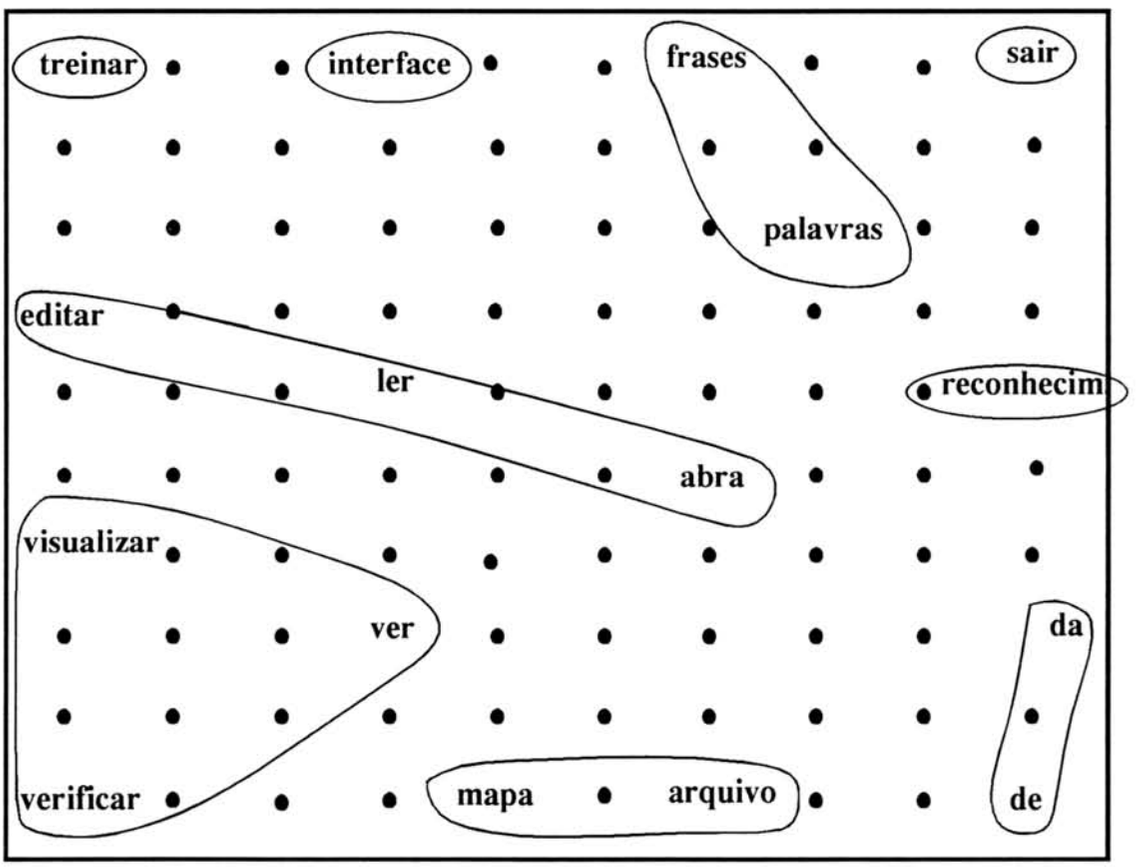


FIGURA 5.9 - **Melhoramento:** uma boa definição semântica é fundamental.

Verifica-se neste mapa de palavras a adequada distribuição das classes, com exceção à classe 1 (abra, ler, editar), dada sua variação das letras e tamanho das palavras, que foge às características das demais classes. Dado isto, sua organização no mapa foi prejudicada, o que poderá acarretar alguns problemas de

reconhecimento. Isto significa que se deve levar em conta estas variantes na hora de propor um padrão frasal, sabendo-se que a palavra *editar* está mais próxima à palavra *visualizar* do que a palavra *abra*, a qual é da mesma classe da primeira (veja a figura 5.9).

Os mapas de frases de duas ou mais palavras também foram remodelados com a inserção de frases com a construção sem nexo, no sentido da linguagem. As dimensões dos mapas e o tempo de treinamento também foram alterados, de modo a permitir um melhor desempenho.

Deste modo, o mapa de frases de duas palavras teve as seguintes inserções:

```
abra frases
palavras ler
visualizar palavras
frases verificar
editar frases
arquivo reconhecimento
ler palavras
```

E foram retiradas as seguintes frases:

```
abra palavras
ler frases
verificar palavras
editar palavras
visualizar frases
mapa reconhecimento
treinar frases
```

Com estas alterações, foi possível prever a distribuição no mapa de frases, uma vez que a formação destas frases foram planejadas segundo a distinção de

classes no mapa de palavras, e alternativas de reconhecimento de palavras de mesma classe. Em outras palavras, *as palavras treinadas necessariamente devem permitir o reconhecimento de palavras não-treinadas, uma vez que estas pertençam a uma mesma classe daquelas treinadas.*

Para um mapa de dimensões 4 x 5 e $t = 10000$, foram obtidos os campos semânticos visualizados na figura 5.10.

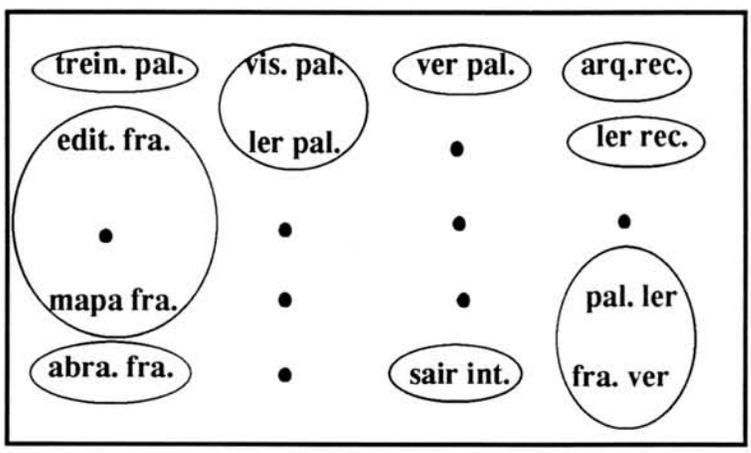


FIGURA 5.10 - **Adequação:** o mapa redimensionado permitiu uma melhor distribuição das frases.

Neste mapa observa-se uma boa distribuição das classes definidas, com excessão daquela formada pelo par *editar frases* e *mapa frases*, uma vez que as palavras *editar* e *mapa* não possuem proximidade no mapa de palavras. Ambas as palavras estão mais próximas àquelas de maior proximidade no mapa de palavras, no caso, *editar* está mais próxima a *treinar* e *mapa* está mais perto de *abra*. Isto demonstra que elaboração das frases para treinamento exige coerência com a organização apresentada no mapa de palavras. Caso contrário, criam-se obstáculos a uma boa classificação.

Para o mapa de frases de três palavras, as inserções foram as seguintes:

```
de frases arquivo
palavras de mapa
```

Foi excluída a frase:

`mapa de palavras`

As alterações seguem os mesmos critérios do mapa de duas palavras, observando-se o posicionamento das palavras no mapa de palavras e sua capacidade de reconhecimento de elementos de uma mesma classe.

O mapa resultante, de dimensões 4 x 4, com $t = 10000$, é apresentado na figura 5.11.

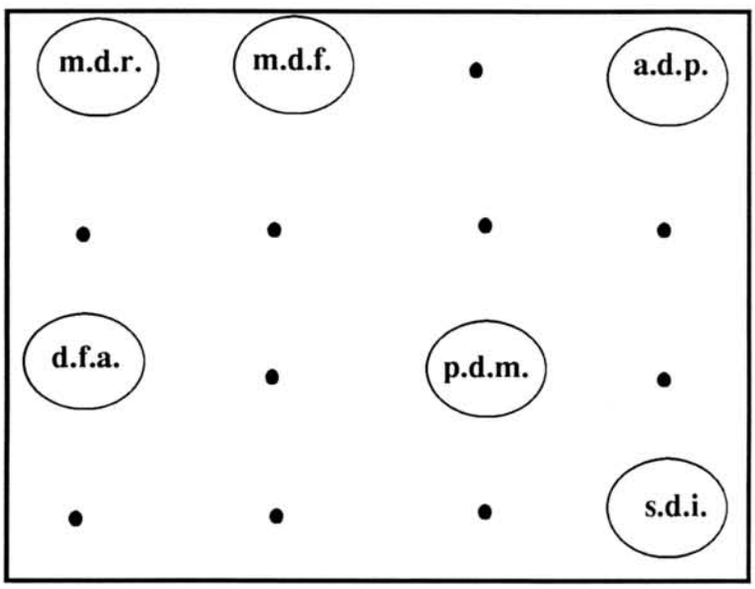


FIGURA 5.11 - **Organização:** Distinção de campos permite a visualização das palavras válidas e inválidas (sem nexo).

Dada a simplicidade de definição das classes, sua distribuição ocorreu de forma clara e consistente.

Já ao mapa de frases de quatro palavras foram acrescentadas as frases:

```
frases treinar mapa de
palavras de arquivo ver
reconhecimento mapa de ver
ver mapa de frases
```

E foram retiradas algumas frases, por serem consideradas redundantes em relação a outras do contexto:

```
verificar mapa de frases
ler arquivo de palavras
ver mapa de palavras
abra arquivo de palavras
treinar mapa de frases
editar arquivo de frases
```

Assim, contando com um mapa redimensionado para 5 x 7, e com tempo de treinamento $t = 50000$, o resultado das alterações é mostrado na figura 5.12

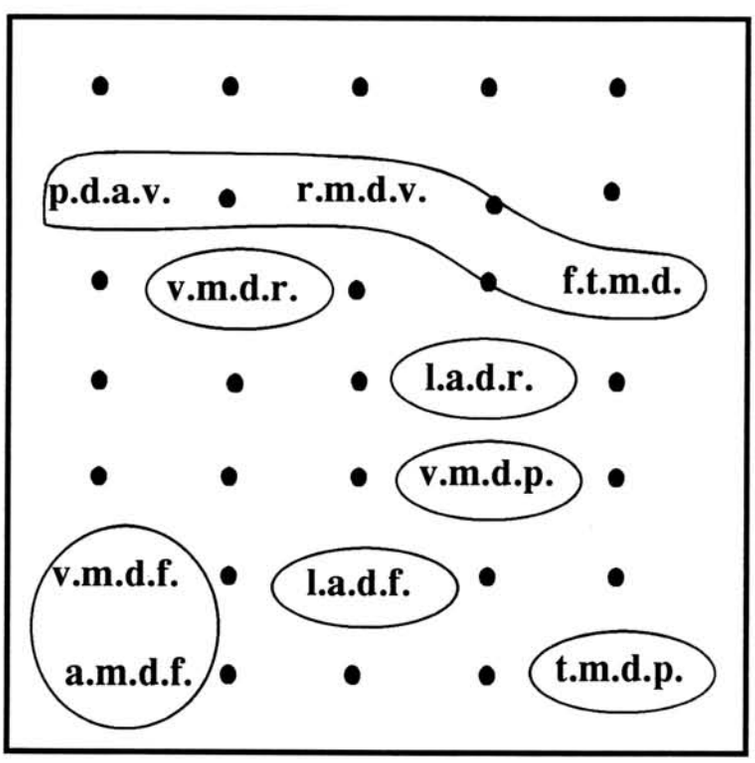


FIGURA 5.12 - **Clareza:** Apesar da complexidade de campos semânticos, há uma boa distinção entre eles.

Observa-se neste mapa que a única classe que ficou prejudicada na distribuição foi a classe das frases sem nexo, formada pelas frases: *palavras de arquivo*

ver, reconhecimento mapa de ver, frases treinar mapa de. As demais classes estão dentro do previsto quando da organização das classes de frases para o treinamento.

5.1.4.1 Novos Resultados

Para uma efetiva otimização dos mapas, torna-se necessária a realização de novos testes de reconhecimento para verificação da capacidade de generalização do protótipo. Como foram alterados os padrões de treinamento, fez-se necessária também a alteração dos padrões de reconhecimento apresentados na seção 5.1.3. Desta forma, são apresentados a seguir os padrões de reconhecimento e seus respectivos mapeamentos para cada um dos mapas de frases.

O reconhecimento feito no **mapa de duas palavras** foi proporcionado pelas seguintes frases:

```
editar palavras
ler frases
abra reconhecimento
palavras editar
verificar reconhecimento
visualizar frases
treinar reconhecimento
frases abra
mapa palavras
reconhecimento ler
sair frases
palavras treinar
editar reconhecimento
ver frases
visualizar reconhecimento
treinar interface
```

```
abra palavras
reconhecimento mapa
palavras ver
ver reconhecimento
verificar palavras
treinar frases
frases ver
mapa reconhecimento
```

A distribuição destes padrões pode ser observada na figura 5.13.

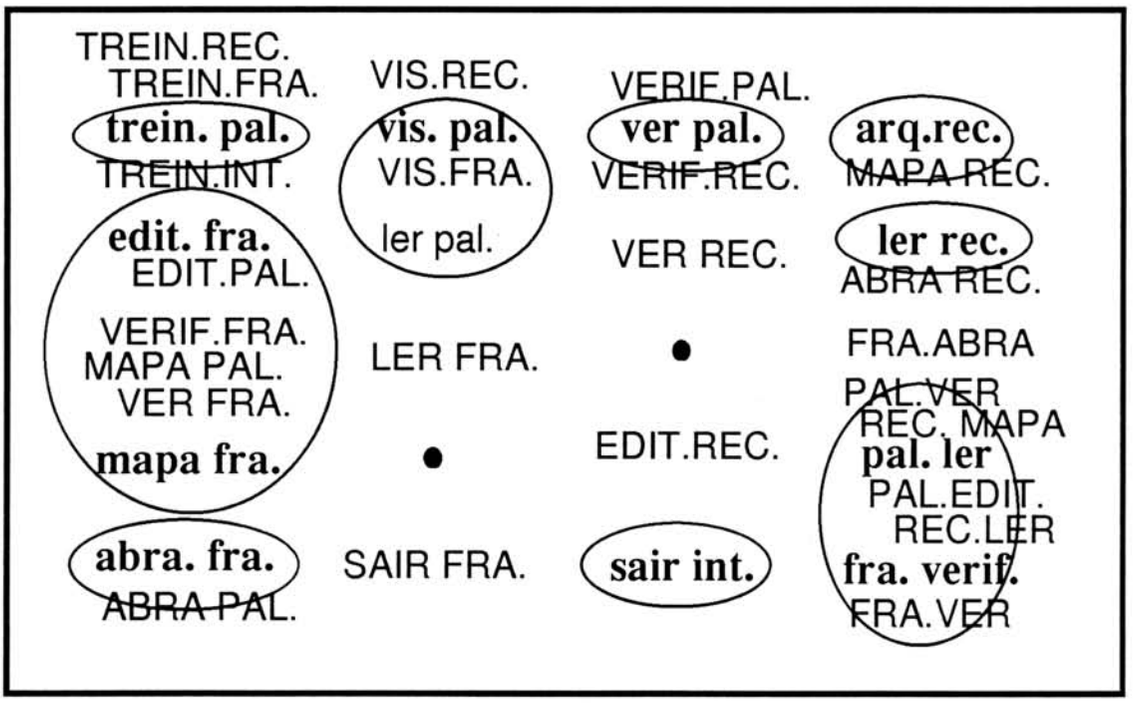


FIGURA 5.13 - **Robustez:** Mesmo com grande quantidade de frases a reconhecer, foi possível um bom reconhecimento.

Como se pode observar, o reconhecimento funciona para as palavras não-treinadas que pertençam a uma mesma classe (ou classe vizinha) de uma palavra treinada existente no interior de uma frase. Isto é mostrado nos exemplos apresentados na figura 5.13, dentre os quais se pode citar a proximidade do reconhecimento de *treinar frases* à frase *treinar palavras*, ou então o reconhecimento de *abra palavras* como sendo semelhante a *abra frases*. A razão disto ocorrer está na definição

das palavras *frases* e *palavras*, as quais pertencem a uma mesma classe no mapa de palavras. Da mesma forma, o reconhecimento do padrão *mapa reconhecimento* coincide com a definição da frase treinada *arquivo reconhecimento*, pois as palavras *mapa* e *arquivo* pertencem a uma mesma classe.

Há, evidentemente, um contra-exemplo que é o fato da frase de reconhecimento *ver frases* estar distante de seu par ideal que é a frase *ver palavras*. A ocorrência disto pode estar no fato dos padrões que possuem o segundo membro como sendo a palavra *frases* estarem concentrados na área esquerda do mapa, fortalecendo as relações para a ocorrência desta distorção.

Finalmente, para este mapa de duas palavras, o fato de terem sido treinados padrões sem nexo, teve como resultado o triunfante reconhecimento de palavras cujos termos estavam invertidos quanto ao restante dos padrões. Como se pode observar na figura 5.13, todos os padrões invertidos foram reconhecidos dentro de uma mesma área, tendo sido, portanto, um reconhecimento com êxito.

Para o mapa de três palavras, foram elaborados os seguintes padrões:

```
mapa de interface
sair de frases
palavras de arquivo
de mapa palavras
arquivo de reconhecimento
mapa de palavras
frases de mapa
```

O reconhecimento destas frases está mapeado na figura 5.14.

Valem aqui as mesmas observações feitas para o mapa de duas palavras, podendo ser ainda acrescentadas mais algumas. Há o caso do padrão *mapa de palavras* que está eqüidistante entre o padrão *mapa de frases* e o padrão *arquivo*

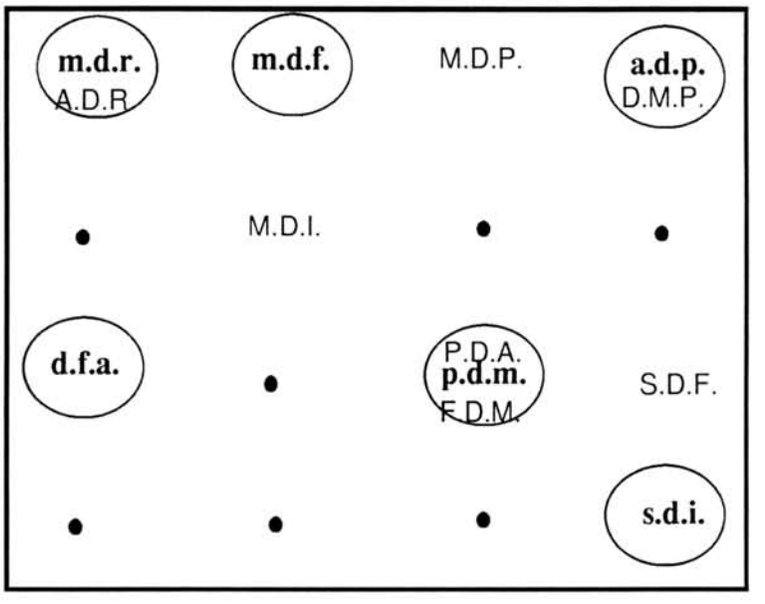


FIGURA 5.14 - **Reconhecimento:** Quanto menos padrões no treinamento, menor a capacidade de distinção.

de palavras. Isto porque o primeiro termo da frase de reconhecimento é *puxado* pelo primeiro de uma das frases, o termo do meio existe em ambas, e o terceiro existe na segunda, de modo que semanticamente quaisquer uma das frases poderia satisfazer a concordância, embora saibamos que apenas a relação *mapa de...* deveria ser suficiente para determinar qual a proximidade correta. Este erro existe porque foram inicialmente definidas as palavras *mapa* e *arquivo* como sendo da mesma classe, o que gerou a ambigüidade neste caso.

Outro caso gerado foi a proximidade entre a frase de reconhecimento *mapa de interface* com a frase treinada *mapa de frases.* Embora as palavras *mapa* e *interface* não sejam de uma mesma classe, elas estão muito próximas no mapa de palavras, o que facilita o reconhecimento apresentado.

Por fim, um contra-exemplo é o da frase sem nexo *de mapa palavras*, que deveria estar próxima ao grupo das demais palavras sem nexo, mas, no entanto, ela foi reconhecida como sendo semelhante à frase *arquivo de palavras*, o que obviamente é um equívoco. Este foi gerado pela falta de dois termos semelhantes à frase apresentada, correspodentes a dois terços da codificação. Se fosse incluída mais uma

frase contendo as palavras *mapa* como segundo termo ou *palavras* como terceiro, certamente a frase seria reconhecida dentro da classe das frases sem nexo.

As frases de reconhecimento para **o mapa de quatro palavras**, são as seguintes:

```
ler mapa de palavras
ver arquivo de frases
ver mapa de palavras
treinar arquivo de palavras
treinar mapa de reconhecimento
verificar mapa de palavras
verificar arquivo de frases
abra arquivo de reconhecimento
abra mapa de palavras
visualizar arquivo de reconhecimento
visualizar mapa de frases
abra arquivo de palavras
arquivo de mapa reconhecimento
reconhecimento arquivo de visualizar
de frases treinar mapa
palavras editar arquivo de
```

Estas frases podem ter sua classificação visualizada na figura 5.15.

Os diversos casos exemplares citados nos mapas anteriores também existem neste mapa. O caso de frase sem nexo ser reconhecida como coerente ocorre com a frase *arquivo de mapa reconhecimento* sendo semelhante a *ver mapa de reconhecimento*, dada a falta de outro padrão treinado com palavras que coincidam em alguma das posições. Ocorre também o caso de eqüidistância, com a frase de reconhecimento *ver arquivo de frase* ser próxima a *ver mapa de frases* e *ler arquivo de frases*.

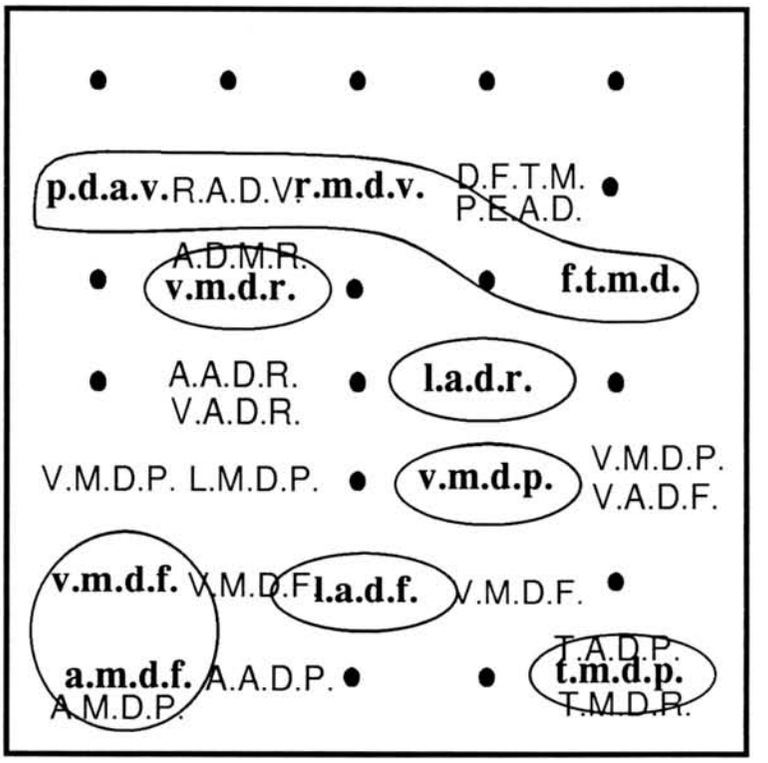


FIGURA 5.15 - **Reflexo:** Um bom treinamento deriva de num bom reconhecimento.

Os demais exemplos foram coerentes, com frases sem nexo sendo atribuídas às frases treinadas, e as frases com palavras de uma mesma classe sendo interpretadas como sendo equivalentes. Alguns exemplos desta última característica são as frases de reconhecimento *abra mapa de palavras* e *abra arquivo de palavras* terem sido dadas como semelhantes à frase *abra mapa de frases*, uma vez que as palavras *frases* e *palavras* pertencem a uma mesma classe, o mesmo ocorrendo às palavras *arquivo* e *mapa*. Há ainda outros reconhecimentos coerentes, como *abra arquivo de reconhecimento* estar próximo à frase *ler arquivo de reconhecimento*, e o padrão *verificar mapa de palavras* ser semelhante ao *visualizar mapa de palavras*.

Estes foram, enfim, os dados resultantes de uma nova abordagem de organização dos elementos de entrada do protótipo. Este bom desempenho demonstra o cuidado necessário na organização e escolha dos padrões a serem treinados, de modo que abranja o máximo possível o conjunto de reconhecimento. A tabela 5.2 mostra o percentual do desempenho obtido.

TABELA 5.2 - **Otimização:** Com o devido planejamento, é possível um melhor desempenho global.

pals. na frase	treinadas	testadas	c. frasal (%)	c. semântica (%)
2	12	24	96	87,5
3	6	7	100	71,4
4	10	16	100	93,7

A partir da observação da tabela 5.2, vê-se que o desempenho, se comparado ao da tabela 5.1, teve um aumento extremamente sensível. Salienta-se o aumento da coerência semântica para as frases de duas e quatro palavras, apesar do percentual ter sido reduzido para as frases de três palavras. Isso demonstra a necessidade de uma melhor avaliação dos padrões treinados e de sua distinção em classes.

5.2 Exemplo2: Controle de Toca-discos a *Laser*

Este é apenas um exemplo hipotético de utilização do reconhecimento semântico propiciado pelo protótipo deste trabalho. Ele consiste no controle de um toca-discos *laser*, que possui os comandos:

1. tocar o disco
2. avanço de faixa do disco;
3. retrocesso de faixa;
4. parar o disco;
5. acionar a pausa estando o disco em rotação;
6. saltar para uma faixa qualquer.

Para permitir a execução dos comandos propostos, utilizou-se o seguinte vocabulário:

```
um                  oito
tocar               pausa
dois                dezoito
disco               nove
tres                o
parar               dez
quatro              onze
avancar             doze
cinco               de
e                   dezenove
seis                treze
pare                para
sete                vinte
toque               faixa
quatorze            dezesseis
va                  dezessete
quinze              a
retroceder
```

Treinamento de Palavras

Pressupondo uma pré-classificação semântica dada pelo meio, parte-se do pressuposto que existem dez conjuntos semânticos de palavras:

- Classe 1: os numerais *um, dois, tres, quatro, cinco, seis, sete, oito, nove, dez, onze, doze, treze, quatorze, quinze, dezesseis, dezessete, dezoito, dezenove, vinte.*
- Classe 2: os comandos *toque, tocar* para acionar o disco.
- Classe 3: os comandos *pare, parar* para encerrar a execução do disco.

- Classe 4: o comando *pausa.*
- Classe 5: o verbo *vá*, que indica um salto de faixa.
- Classe 6: o comando *avançar.*
- Classe 7: as preposições *de, para.*
- Classe 8: o comando *retroceder.*
- Classe 9: os substantivos *disco, faixa.*
- Classe 10: os artigos *o, a* e a conjunção *e.*

Estas palavras foram então codificadas através do programa *tranp* para serem utilizadas como padrão de entrada no programa *tremapa*, de aprendizado e construção do mapa de palavras. Os parâmetros de aprendizado utilizados foram o raio inicial $\sigma = 4.0$, o valor de adaptação $\epsilon = 0.7$ e um tempo máximo $t = 50000$ e um mapa retangular de dimensão 10 x 10.

Para obtenção dos resultados, é necessária uma nova aplicação do programa *tranp* sobre o conjunto de palavras a reconhecer, e a utilização das palavras codificadas como entrada no programa *recmapa.* O mapa gerado é apresentado na figura 5.16.

Cabem aqui algumas observações sobre este resultado gerado, tais como:

1. a proximidade entre as palavras de uma mesma classe, com algumas exceções;
2. a distribuição coerente das classes dentro do mapa, sem sobreposições.

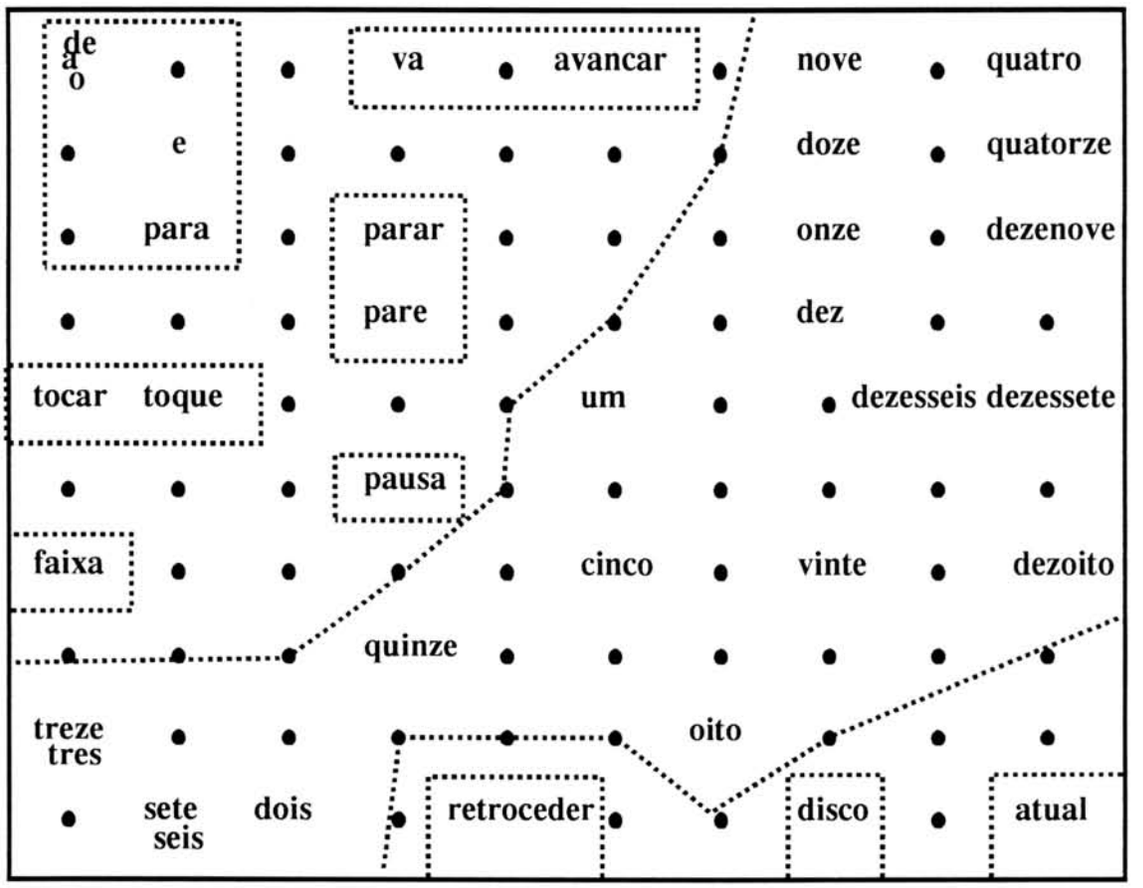


FIGURA 5.16 - **Mapa de Vocabulário:** Distribuição das palavras conforme sua classe.

5.2.1 Treinamento de Comandos Frasais

Definido o mapa de palavras, é necessário a definição de comandos frasais e suas respectivas classes semânticas. Como a organização das frases é um conjunto de dados mais complexo que as palavras, torna-se necessário sua separação por tamanho (número de palavras), criando-se um mapa para cada conjunto de frases de mesmo tamanho. Estes conjuntos foram divididos em seis classes, distribuídas em quatro mapas, respeitando o tamanho das frases. A ordem por classe, respeitando as funções definidas anteriormente para este exemplo, é a seguinte:

- Classe 1: *toque, tocar disco, toque o disco, tocar o disco.*
- Classe 2: *avancar, avancar faixa, avancar a faixa.*

- Classe 3: *retroceder, retroceder faixa, retroceder a faixa.*
- Classe 4: *parar, pare o disco.*
- Classe 5: *pausa.*
- Classe 6: *faixa um, faixa dois, faixa quatro, faixa cinco, faixa nove, faixa dez, faixa onze, faixa treze, faixa quatorze, faixa quinze, faixa dezesseis, faixa vinte, faixa vinte e um, faixa vinte e dois, faixa vinte e quatro, faixa vinte e nove, toque a faixa um, toque a faixa dois, toque a faixa quatro, toque a faixa cinco, toque a faixa nove, toque a faixa dez, toque a faixa onze, toque a faixa quatorze, toque a faixa quinze, toque a faixa dezesseis, toque a faixa vinte, va para a faixa um, va para a faixa dois, va para a faixa quatro, va para a faixa cinco, va para a faixa nove, va para a faixa dez, va para a faixa treze, va para a faixa quatorze, va para a faixa quinze, va para a faixa dezesseis, va para a faixa vinte.*

Assim, as frases de um mesmo conjunto foram então codificadas através do programa *tranf* para serem utilizadas como entrada no programa *tremapa*, de aprendizado do mapa de frases. Os parâmetros de aprendizado utilizados para todos os conjuntos foram raio inicial $\sigma = 4.0$, o valor de adaptação $\epsilon = 0.7$ e os demais foram definidos em função do número de componentes:

- mapa de uma palavra: tempo máximo $t = 10000$ e mapa retangular 4 x 4;
- mapa de duas palavras: tempo máximo $t = 50000$ e mapa retangular 10 x 10;
- mapa de três palavras: tempo máximo $t = 20000$ e mapa retangular 7 x 7;
- mapa de quatro palavras: tempo máximo $t = 10000$ e mapa retangular 6 x 6;

- mapa de cinco palavras: tempo máximo $t = 10000$ e mapa retangular 6 x 6;

Novamente, para a obtenção dos resultados, é necessária a utilização do programa *tranf* sobre os conjunto de frases a reconhecer, e a utilização das frases codificadas como entrada no programa *recmapa*. Os mapas resultantes são apresentados a seguir.

Para o conjunto de **uma palavra** tem-se os seguintes padrões:

```
toque
avancar
retroceder
parar
pausa
```

os quais foram organizados no mapa da figura 5.17.

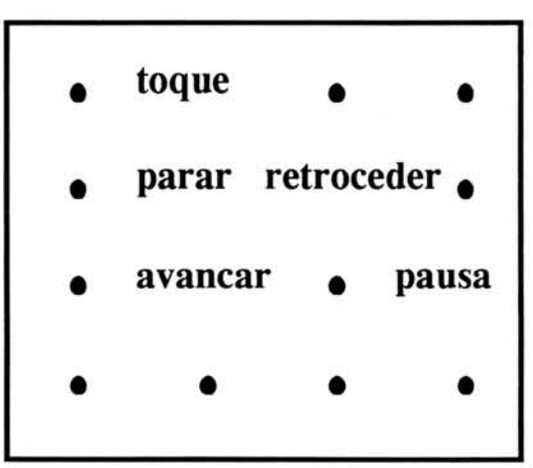


FIGURA 5.17 - **Frases de Uma Palavra:** comandos básicos do toca-discos.

Para o conjunto de **duas palavras** temos os seguintes padrões:

```
retroceder faixa
avancar faixa
```

```
tocar disco
faixa um
faixa dois
faixa quatro
faixa cinco
faixa nove
faixa dez
faixa onze
faixa treze
faixa quatorze
faixa quinze
faixa dezesseis
faixa vinte
```

os quais foram organizados no mapa da figura 5.18.

O conjunto de **três palavras** é formado pelos padrões:

```
toque o disco
retroceder a faixa
parar o disco
avancar a faixa
tocar o disco
pare o disco
```

que estão classificados no mapa da figura 5.19.

O conjunto de **quatro palavras** é composto pelos padrões:

```
faixa vinte e um
faixa vinte e dois
faixa vinte e quatro
```

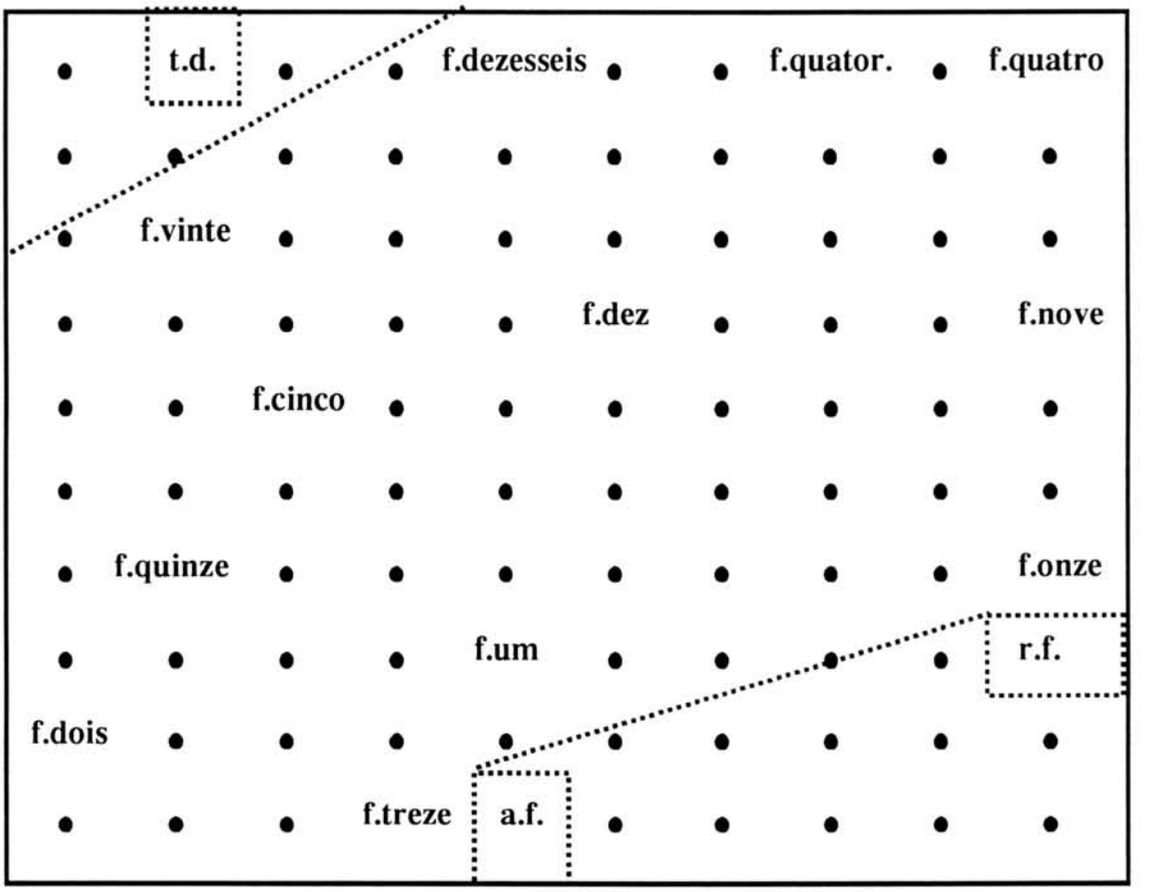


FIGURA 5.18 - **Frases de Duas Palavras:** padrões de aprendizado com diversos numerais.

```
faixa vinte e nove
toque a faixa um
toque a faixa dois
toque a faixa quatro
toque a faixa cinco
toque a faixa nove
toque a faixa dez
toque a faixa onze
toque a faixa quatorze
toque a faixa quinze
toque a faixa dezesseis
toque a faixa vinte
```

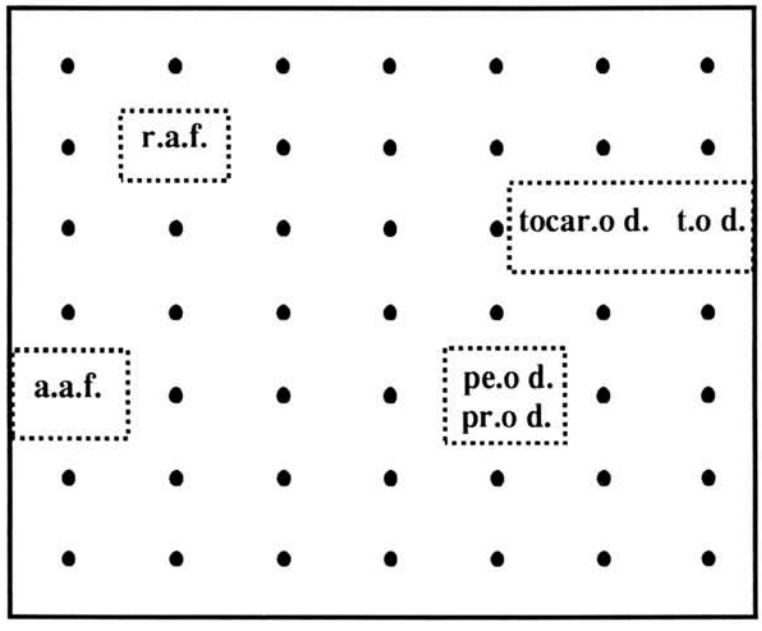


FIGURA 5.19 - **Frases de Três Palavras:** comandos com frases compostas.

que estão classificados no mapa da figura 5.20.

O conjunto de **cinco palavras** possui os seguintes padrões:

```
va para a faixa um
va para a faixa dois
va para a faixa quatro
va para a faixa cinco
va para a faixa nove
va para a faixa dez
va para a faixa treze
va para a faixa quatorze
va para a faixa quinze
va para a faixa dezesseis
va para a faixa vinte
```

que são apresentados no mapa da figura 5.21.

Há algumas observações importantes sobre os resultados gerados:

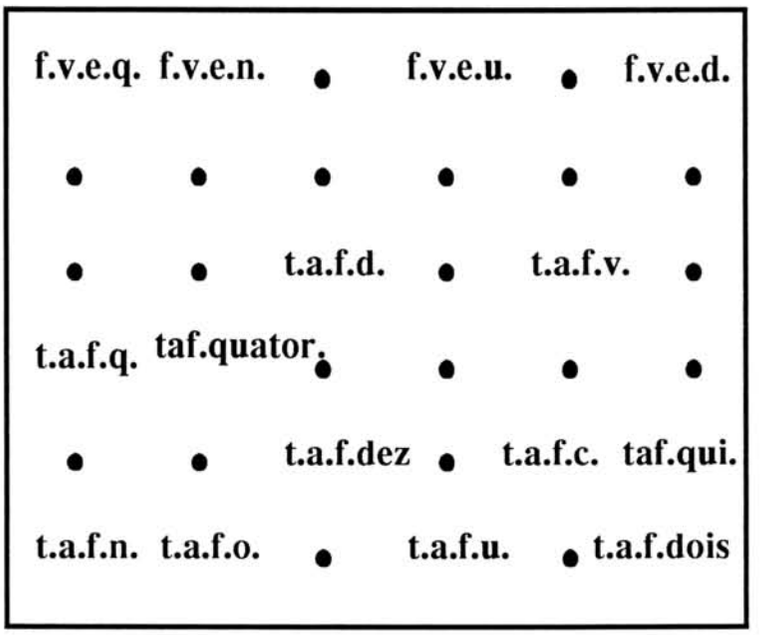


FIGURA 5.20 - **Frases de Quatro Palavras:** diferenciação automática entre os comandos com a palavra *faixa* e com a palavra *toque*.

1. a proximidade das frases é proporcional à proximidade das palavras no mapa de palavras;

2. nem sempre há uma proximidade entre as palavras de uma mesma classe, podendo ser mais próxima a outra classe;

3. existe a organização de grupos de frases com mesmas palavras, ou palavras semelhantes, e que possuem mesma posição dentro da frase.

5.2.2 Reconhecimento de Frases Não Treinadas

De modo a testar-se novamente a capacidade de generalização do protótipo, verifica-se o *grau de acerto* deste segundo exemplo, o qual dará margem para uma posterior comparação dos desempenhos de reconhecimento.

Para este segundo exemplo foram apresentadas ao protótipo o seguinte conjunto de frases:

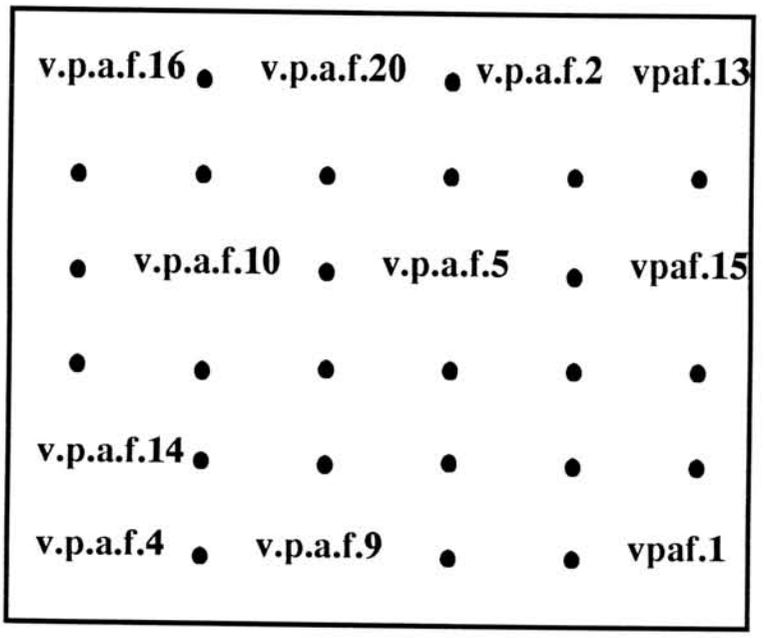


FIGURA 5.21 - **Frases de Cinco Palavras:** numerais aproximam-se conforme sua distribuição no mapa de palavras.

- **para frases de duas palavras:**

```
retroceder disco
avancar disco
tocar faixa
faixa tres
faixa seis
faixa sete
faixa oito
faixa doze
faixa dezessete
faixa dezoito
faixa dezenove
```

Comparando-se com o mapa original, como se vê na figura 5.22, percebe-se a distância entre as frases treinadas e as não-treinadas. As frases de teste estão em letras maiúsculas.

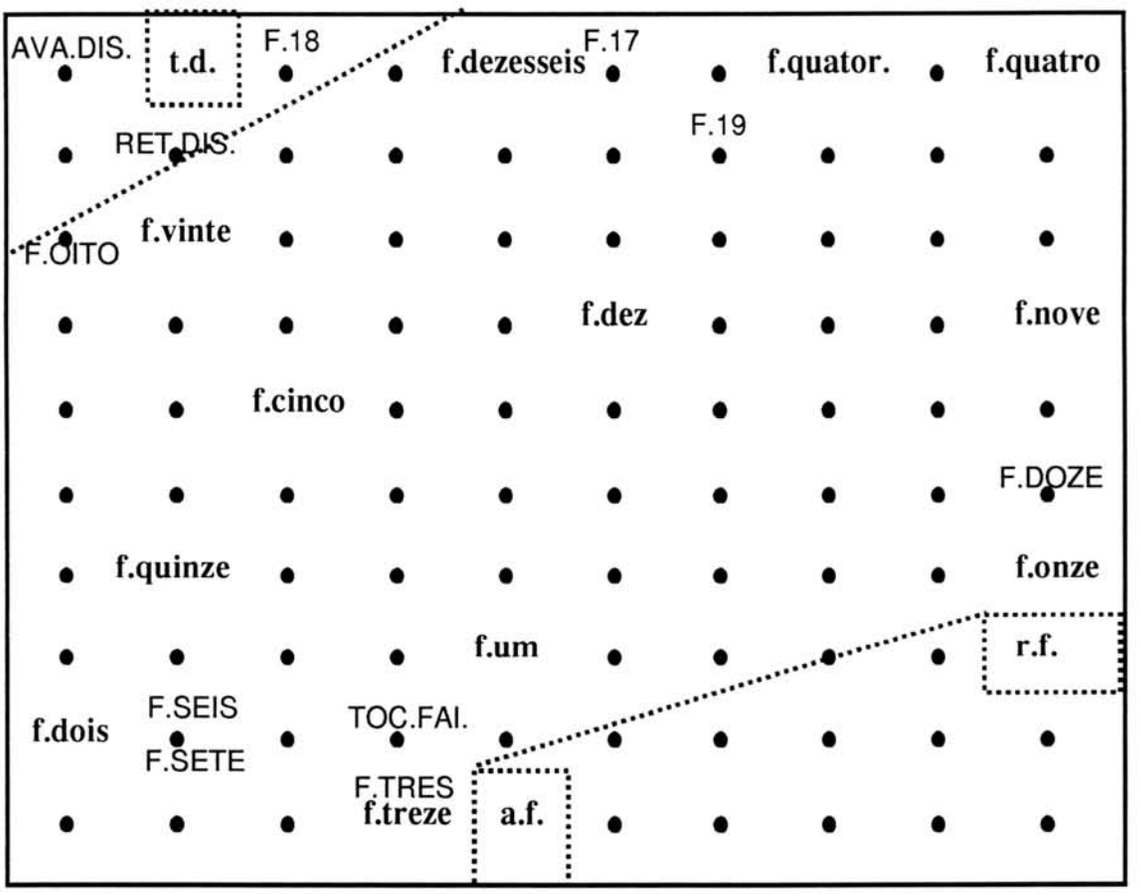


FIGURA 5.22 - **Aproximação:** frases não aprendidas têm aproximação através das palavras próximas no mapa de palavras.

- **para frases de três palavras:**

```
toque a faixa
retroceder o disco
parar a faixa
avancar o disco
tocar a faixa
pare a faixa
```

Mais uma vez, é feita a comparação entre o mapa treinado e as palavras a reconhecer na figura 5.23.

- **para frases de quatro palavras:**

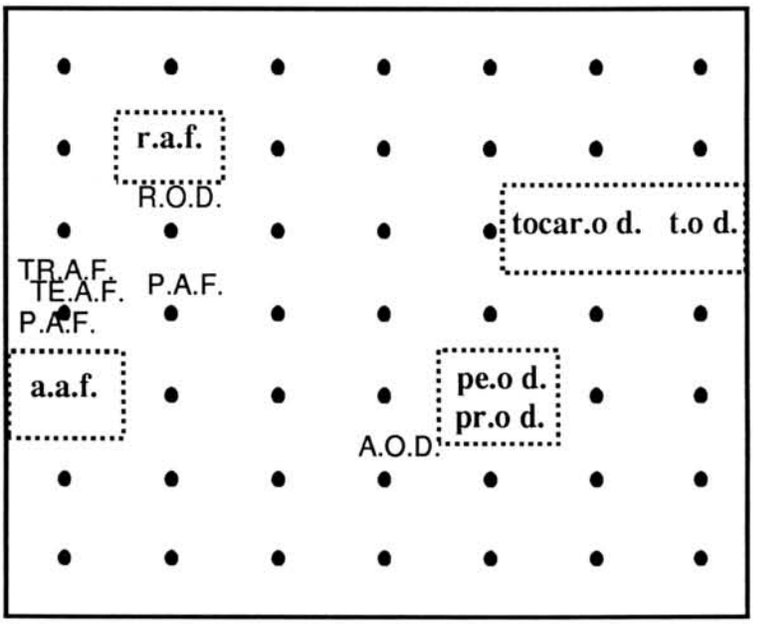


FIGURA 5.23 - **Confusão Frasal:** algumas frases aproximam-se pelo número de caracteres semelhantes.

```
faixa vinte e tres
faixa vinte e cinco
faixa vinte e seis
faixa vinte e sete
faixa vinte e oito
toque a faixa tres
toque a faixa seis
toque a faixa sete
toque a faixa oito
toque a faixa doze
toque a faixa treze
toque a faixa dezessete
toque a faixa dezoito
toque a faixa dezenove
```

As frases de quatro palavras são comparadas na figura 5.24.

- **para frases de cinco palavras:**

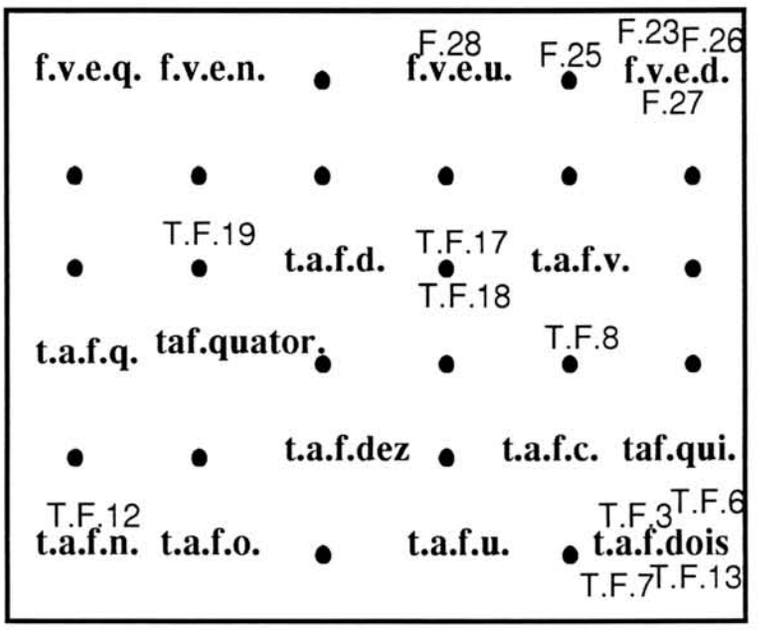


FIGURA 5.24 - **Frases Indicadas:** apesar das frases de teste não terem sido treinadas com o numeral, elas são aproximadas daquelas de numeral próximo no mapa de palavras.

```
va para a faixa tres
va para a faixa seis
va para a faixa sete
va para a faixa oito
va para a faixa onze
va para a faixa doze
va para a faixa dezessete
va para a faixa dezoito
va para a faixa dezenove
```

A visualização deste conjunto pode ser feita comparando-se com o mapa original, como na figura 5.25. As frases de teste estão em letras maiúsculas.

Comparando os resultados, obtemos a tabela 5.3, onde visualiza-se o grau de acerto relativo ao reconhecimento semântico

Com base nos resultados obtidos, pode-se observar:

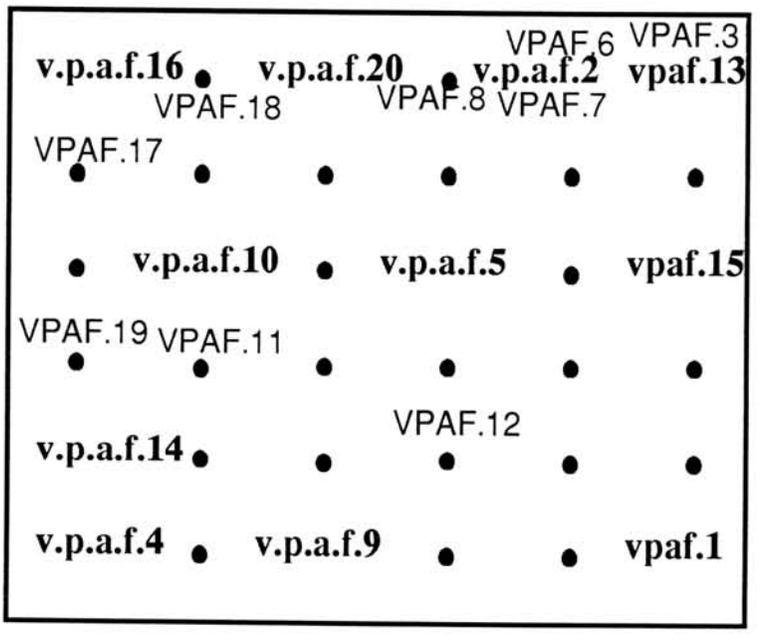


FIGURA 5.25 - **Aproximação:** mais uma vez o mapa de palavras define a aproximação das frases no mapa de frases.

TABELA 5.3 - **Resultados:** verificação do grau de acerto das frases de teste.

pals. na frase	treinadas	testadas	c. frasal (%)	c. semântica (%)
2	15	11	100	81,8
3	6	6	100	50
4	15	14	100	28,5
5	11	9	100	33,3

1. um maior número de caracteres (letras) semelhantes em mesma posição dentro de uma frase é fator determinante de aproximação de significado;

2. existe aproximação entre frases com determinado número de palavras comuns;

3. a composição (ordem de disposição) das palavras na frase treinada e na frase de teste é determinante para um bom resultado;

4. *a aproximação entre as frases treinadas e de teste têm como fator principal sua proximidade no mapa de palavras.*

5.2.3 Otimização do Treinamento

Para uma boa utilização do protótipo, cabe ao usuário a responsabilidade de organizar coerentemente a classificação semântica das palavras e, posteriormente, das frases. Desta maneira possibilita-se um efetivo reconhecimento de comandos frasais, com as devidas interpretações a nível semântico.

Como os índices de acerto quanto à coerência semântica dos exemplos anteriores (veja tabela 5.3) não foram satisfatórios, tornou-se necessária a alteração na concepção das palavras utilizadas. Verificou-se a inutilidade dos numerais que compunham o mapa de palavras apresentado na figura 5.16, e foram retiradas, então, as classes 1, 5 e 7 e a conjunção *e*. Outra alteração realizada foi a colocação das palavras *disco* e *faixa* em classes distintas.

Os parâmetros de treinamento ficaram inalterados. O mapa de palavras resultante está está na figura 5.26.

A distribuição das classes neste mapa de palavras apresenta-se uniforme e coerente, apresentando clareza nas relações entre os membros das classes. Essas indicações demonstram um posterior reconhecimento de frases provavelmente também com clareza de relacionamentos.

Com a alteração do mapa de palavras, também foi necessária a mudança dos mapas de frases. No caso, foram alterados os mapas de frases de duas ou mais palavras, que tiveram algumas frases extraídas. Desta forma, foram também alteradas as dimensões dos mapas, acompanhando a redução do número de padrões.

O mapa de frases de duas palavras ficou com os seguintes padrões:

```
retroceder faixa
avancar faixa
toque disco
faixa tocar
```

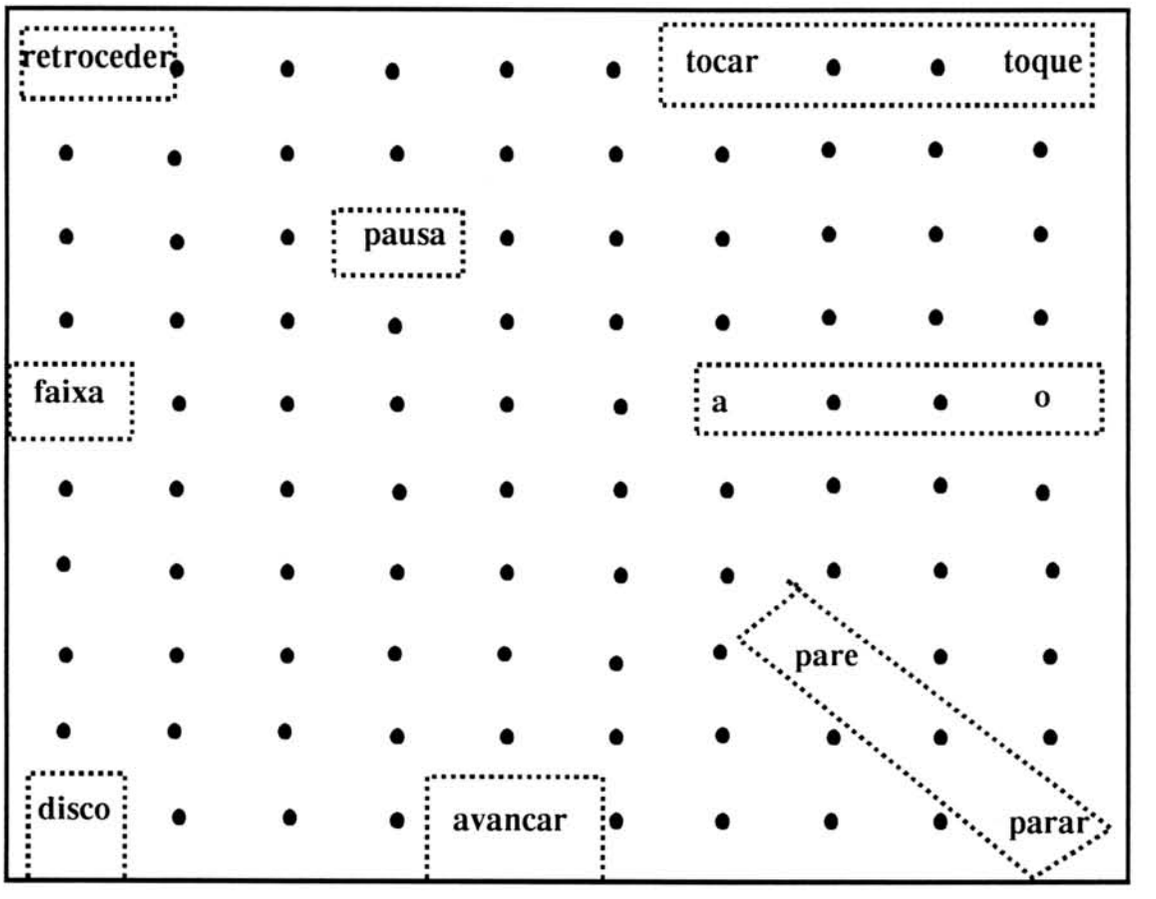


FIGURA 5.26 - **Otimização:** o enxugamento de palavras excessivas é fundamental para uma boa organização.

```
parar disco
disco avancar
```

Observa-se que neste conjunto de frases foram inseridas algumas com palavras invertidas, para a formação de uma classe de frases sem nexo.

Para um mapa de dimensões 10 x 10, tempo $t = 10000$, foram obtidos os campos semânticos visualizados na figura 5.27.

Como o número de padrões é pequeno e as classes são unitárias, não há maiores problemas de classificação, afora a distância entre as frases sem nexo, esta justificada pela distância existente no mapa de palavras.

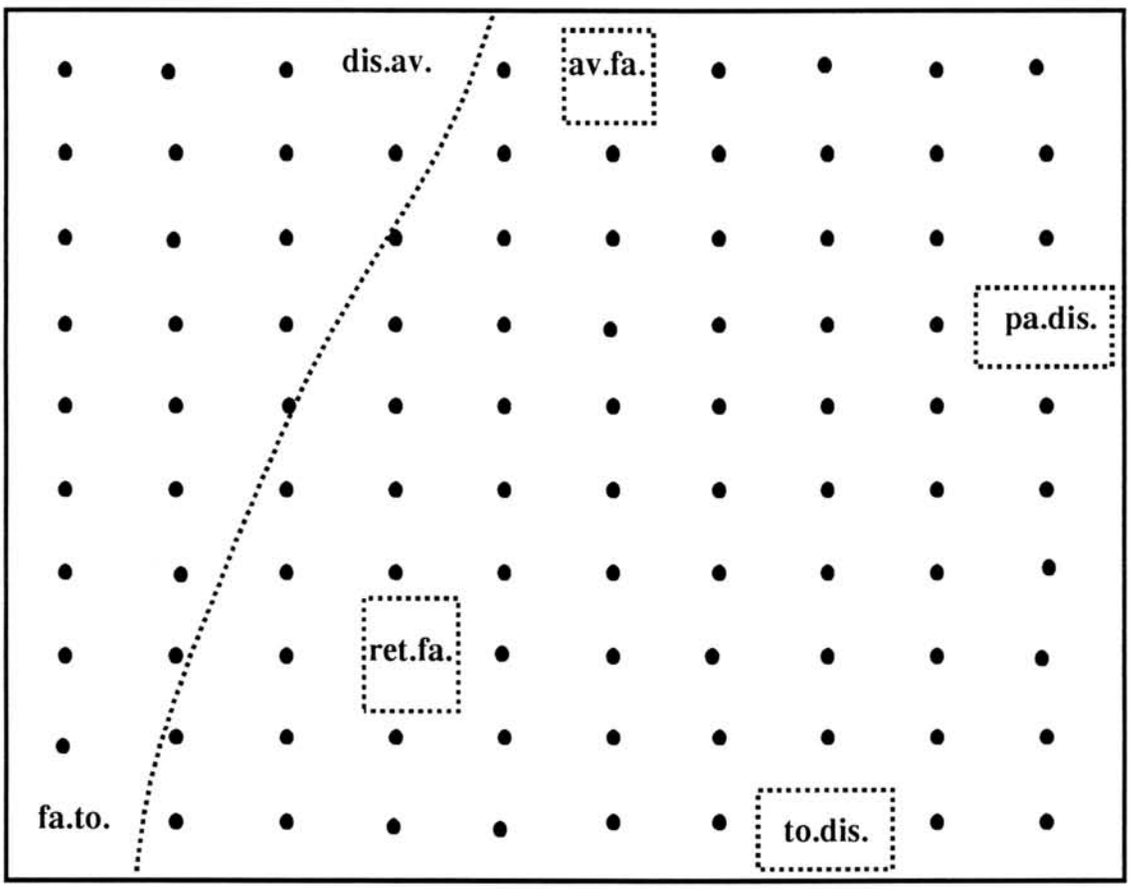


FIGURA 5.27 - **Distribuição:** A mesma dimensão do mapa com menos padrões facilita a distinção entre as classes.

Por fim, o mapa de frases de três palavras foi treinado com as seguintes frases:

```
toque o disco
retroceder a faixa
disco o parar
avancar a faixa
pare o disco
faixa a tocar
```

O diferencial deste conjunto de padrões está no fato de haver padrões com os elementos que costumamente são no início da frase estão no final. Isto foi realizado visando o reconhecimento de uma classe de frases sem nexo.

O mapa resultante, de dimensões 8 x 7, tempo de aprendizado $t = 50000$, é apresentado na figura 5.28.

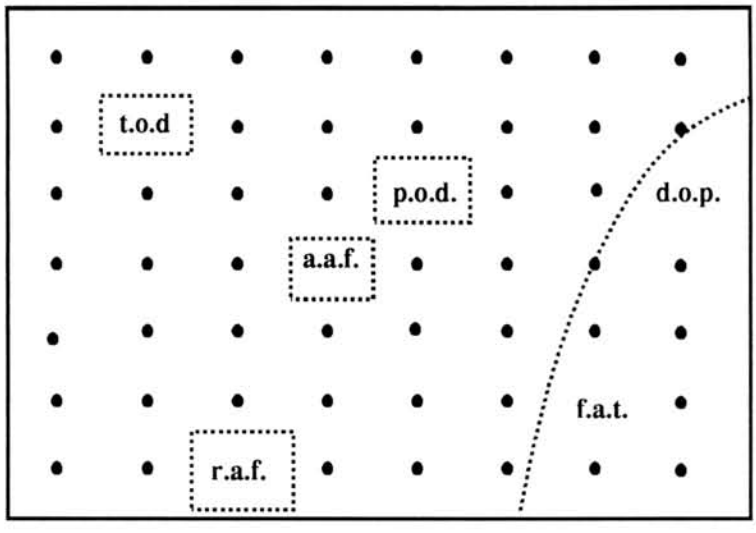


FIGURA 5.28 - **Distinção:** As classes são distribuídas conforme a sua definição e a orientação do mapa de palavras.

Como esperado, a distribuição das classes ocorreu uniformemente, de modo que há uma clara distinção entre os membros de uma e outra.

Como os mapas de *quatro* e *cinco* palavras continham apenas indicações de numerais, eles foram suprimidos do conjunto de testes.

5.2.3.1 Resultados do Reconhecimento

Para que se possa ter confiança nas alterações dos padrões e das classes treinadas para os comandos do toca-discos, é necessária a verificação através do reconhecimento de padrões não-treinados. Como as frases treinadas neste novo conjunto foram modificadas, também os padrões de reconhecimento são diversos daqueles apresentados na seção 5.2.2. Desta forma, a seguir são mostrados os novos padrões de reconhecimento e seus respectivos mapeamentos para cada um dos mapas de palavras.

Para o reconhecimento no mapa de duas palavras foram elaboradas as seguintes frases:

```
retroceder disco
avancar disco
tocar disco
disco retroceder
pare disco
faixa parar
```

A distribuição destes padrões pode ser observada na figura 5.29.

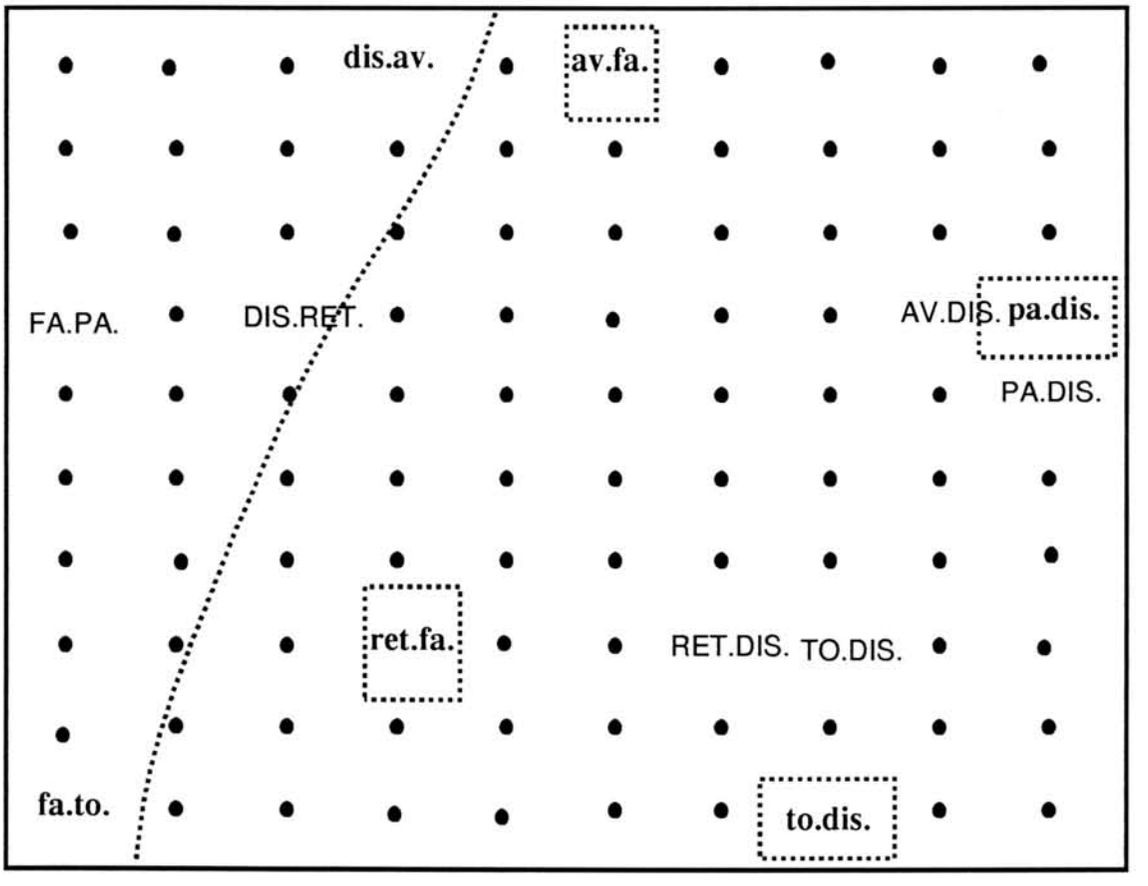


FIGURA 5.29 - **Aproximação:** As frases não-treinadas aproximam-se das classes pré-definidas mais compatíveis.

No mapa da figura 5.29, é possível verificar que os padrões não-treinados foram distribuídos de forma coerente com sua disposição no mapa de palavras. Por

outro lado, semanticamente ficou a desejar a ambigüidade da frase *retroceder disco*, disposta entre as frases treinadas *retroceder faixa* e *tocar disco*.

Para o mapa de três palavras, foram elaborados os seguintes padrões:

```
toque a faixa
retroceder a disco
parar a faixa
disco o retroceder
avancar o disco
parar o disco
tocar a faixa
faixa o toque
tocar o disco
```

O reconhecimento destas frases está mapeado na figura 5.30.

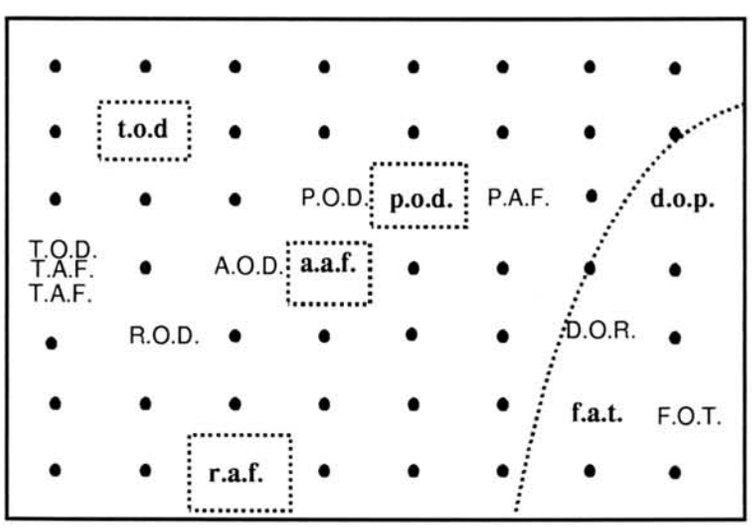


FIGURA 5.30 - **Coerência:** O reconhecimento é bem definido quando as classes também são distintas.

Neste reconhecimento do mapa de frases de três palavras, observa-se que todas as frases, exceto uma, tiveram uma distribuição semanticamente coerente com as classes pré-definidas. A frase que obteve ambigüidade foi *parar o disco*, que está

a meio caminho entre as frases *avancar a faixa* e *pare o disco*. Resultado este que se atribui à proximidade recíproca das palavras componentes no mapa de palavras: *avancar* é próxima a *parar*; *o* é próximo a *a*; *disco* é próximo a *faixa*. Desta forma, pela própria definição das palavras, toda a frase possui uma proximidade de relações, sendo, portanto, verificadas como semelhantes.

Por fim, após novos conjuntos de padrões de treinamento e reconhecimento, pode-se observar um ganho significativo com o planejamento das classes semânticas. Neste caso específico dos comandos para controle do toca-discos, foi necessário o enxugamento de palavras excessivas para a obtenção de um desempenho satisfatório. Cabe aqui lembrar que o desempenho obtido no reconhecimento é fruto da tentativa de aproximação de significado de frases não-treinadas, o que, para qualquer ser humano, é algo passível de muitos erros. Uma pessoa naturalmente aproxima o significado de uma frase ou palavra por ela desconhecida.

Os resultados finais deste segundo conjunto de reconhecimento está apresentado na tabela 5.4, que mostra o percentual do desempenho obtido.

TABELA 5.4 - **Ganho:** Um bom planejamento de classes semânticas proporciona um reconhecimento semântico eficaz.

pals. na frase	treinadas	testadas	c. frasal (%)	c. semântica (%)
2	6	6	100	83,3
3	6	9	100	88,8

A tabela 5.4 apresentada é uma demonstração clara da necessidade de planejamento quanto à definição de quais classes abrigarão quais frases. Uma decisão errada proporciona todo um conjunto de reconhecimento com problemas.

Como se observa na tabela 5.4, o ganho de desempenho ocorreu na totalidade do conjunto de treinamento, o que representa uma melhoria excepcional se comparado com os resultados anteriores da tabela 5.3. Esta melhoria no desempenho demonstra que, se houver um melhoramento ainda maior do projeto de classificação inicial e uma utilização de outras ferramentas de decisão, será possível a utilização

prática desta tecnologia para reconhecimento de padrões frasais a níveis ótimos de acerto.

6 CONCLUSÕES

A Representação da Linguagem

Pode-se afirmar que a representação da linguagem humana no computador, visando uma comunicação mais amigável entre o homem e a máquina, é uma das fronteiras da Ciência da Computação nos dias de hoje. De fato, a linguagem não é um resultado trivial do desenvolvimento das capacidades cognitivas do homem e, deste modo, também não é passível de uma modelagem trivial. Como analisado em trabalhos anteriores ([MÜL 93] [MÜL 95]) e no capítulo introdutório do presente trabalho, conclui-se que a modelagem do processo de aquisição da linguagem será ainda por muitos anos um motivador de pesquisas, tal sua complexidade. Neste sentido, o presente trabalho procurou identificar as características do processo de aquisição da linguagem passíveis de modelagem computacional. Como resultado desta pesquisa conclui-se que simplesmente inexiste uma formalização deste processo, conforme analisado na seção 1.2.2, e portanto faz-se necessária a execução desta etapa para que se atinja uma coerente representação computacional.

Embora não exista uma representação formal do processo de desenvolvimento da linguagem, há muitos trabalhos que procuram estabelecer caminhos para a comunicação homem-máquina. Dentre estes trabalhos, foram citados no capítulo 2 alguns pertencentes à IA, tendo sido ressaltados os sistemas de nível subsimbólico no capítulo 3, os quais envolvem as RNAs, abordagem central do presente trabalho.

Analisando-se as construções dos sistemas de RNAs para reconhecimento de linguagem, verifica-se a grande preocupação com métodos de pesquisa em banco de dados, ou de pergunta-resposta. Comparando-se a origem e função da linguagem humana (ver [MÜL 95]) com o que é proposto pelos sistemas analisados no capítulo 3, ver-se-á um abismo e uma incompatibilidade entre eles, o que prova a falta de

objetivos mais concretos e de uma formalização para este tipo de desenvolvimento computacional.

O desenvolvimento da linguagem humana deu-se em função das expressões de ações necessárias a sua sobrevivência, de modo a interagir com o meio-ambiente, adaptar-se a ele e adaptá-lo às suas necessidades. Neste sentido, o computador - independente de sua falta de emotividade ou intencionalidade - ainda está muito distante de ser programado para sua própria auto-suficiencia, de ser um organismo de interação com o meio.

Não é assim que um computador é visto atualmente, ao contrário, ele é projetado e programado apenas como uma máquina de obtenção de informações. Por outro lado, se for analisada a necessidade de comunicação com o computador, verificar-se-á a interação com uma máquina que adquire, processa e apresenta dados. Neste contexto, o computador poderia alterar o meio com os dados apresentados, e o meio poderia alterar seus dados internos. Nada disso, porém, influencia (com algumas exceções) a construção do computador ou sua continuidade de operação. As excessões a este caso podem ser citadas como sendo os programas de tolerância a falhas, que trabalham pela manutenção da continuidade do processamento. A necessidade do computador de *proteger-se* contra eventuais danos a seu funcionamento não é previsto em projetos, afinal, ele não é visto como uma máquina auto-regulável, mas sim como uma entidade que necessita da programação explícita passo-a-passo.

Enquanto o computador for construído (e reconstruído) com uma visão de programador, e não de usuário, ele estará sujeito a ser apenas uma máquina de obtenção de informações. Obtenção esta realizada de uma maneira pouco ou nada humanizada.

Alternativas de Representação

Na tentativa de ultrapassar a fronteira da representação, enquanto não se encontra uma formalização adequada à linguagem, buscam-se alternativas para auxiliar na compreensão do processo. Aqui está sendo tratada a alternativa relativa às RNAs, embora saiba-se que apenas esta metodologia não é suficiente para a modelagem do processo de aquisição e desenvolvimento da linguagem, conforme o MPL analisado na seção 1.2.1. A aplicação de RNAs aqui apresentada pretende a busca da compreensão e formalização do processamento da linguagem humana, e não uma (utópica) simulação deste processamento.

Assim como há tentativas de realização da implementação de regras lingüísticas por Sistemas de Linguagem Natural, através de sistemas de IA simbólica, aqui são tratadas relações semânticas existentes na linguagem, tarefa esta executada por sistemas de IA subsimbólica, as RNAs. Neste contexto, o presente trabalho insere-se na representação das relações semânticas da linguagem, dentro de uma abordagem inédita, onde o usuário do protótipo define as relações de acordo com a semântica de aplicação do vocabulário. A partir das definições de campos semânticos de palavras e frases, obtém-se um sistema capaz de reconhecer frases por aproximação de *contextos semelhantes*. Por *contextos semelhantes* entendam-se frases de testes com palavras e organização sintática semelhantes ou em mesmos campos semânticos às das frases treinadas. Sua utilização eficaz ficou provada com os dois exemplos dados no capítulo 5, os quais serão analisados a seguir.

Um Caminho para A Representação da Linguagem

Conforme comentado anteriormente, o computador é visto como uma máquina de obtenção de informações, e não como um sistema passível de auto-

regulação. O protótipo aqui apresentado tem a intenção de determinar o computador como um sistema passível de treinamento sobre o qual é feita uma tomada de decisões, visando uma operação interativa com o usuário. Interação esta que possibilitará, num próximo momento, o comando frasal falado pelo ser humano. Com isto será possível a realização de comandos por voz para o controle de artefatos com eletrônica embarcada, tais como computadores, terminais de consulta bancária, eletrodomésticos, veículos, etc.

A interface apresentada no apêndice 1 é um exemplo da interação possibilitada pelo uso das RNAs no reconhecimento de comandos frasais. A uma dada frase, a interface responde na forma da execução de um comando, que foi identificado através de uma consulta ao mapa de frases. Caso não seja uma frase coerente com aquelas treinadas, será tentada uma aproximação, cujos resultados de testes encontram-se na seção 5.1.

Resultados e Aperfeiçoamentos

Os resultados obtidos são bons o suficiente para uma análise do que será necessário evoluir. Nos testes do capítulo 5, foi demonstrado que devem ser desenvolvidas técnicas de distinção de classes de palavras e de frases dentro de um determinado contexto, que levem a um correto treinamento de padrões. Estas técnicas deverão ter suas características alicerçadas no meio-ambiente do contexto em questão, uma vez que o meio é o melhor agente de definição de classes semânticas, conforme analisado na seção 1.2.

Por outro lado, não basta a distinção de classes semânticas, mas também sua melhor distribuição e organização no mapa de palavras. Para tanto, uma alternativa viável é a utilização do modelo Linear Vector Quantization (LVQ), que é uma evolução do Mapa Auto-Organizável aqui utilizado, e que permite a aproximação no mapa de elementos de uma mesma classe. Uma vez tendo classes bem defini-

das, acabam-se os problemas de ambigüidade verificados, bem como os demais erros causados por distorções nas posições do mapa de palavras.

Afora os alguns erros proporcionados pela má distribuição das palavras, pode-se constatar a importante contribuição da capacidade de generalização do modelo de rede neural utilizado, permitindo a aproximação por semelhança sintática e semântica. Isso pode ser verificado no fato de que sete dos doze mapas treinados terem pemitido resultados com acerto acima de 70%. E destes sete, cinco foram obtidos após a otimização das classes, ou seja, todos os mapas otimizados tiveram acerto acima de 70%, e quatro deles acima de 80%. Isso significa um desempenho muito bom se for acrescentado o fato de que estes índices de acerto de reconhecimento são de *frases não-treinadas, ou seja, de frases não existentes no conjunto de padrões frasais.*

Por sua capacidade de generalização, a modelagem atende às necessidades da efetivação de um reconhecimento semântico de padrões de frases. Uma vez que existam campos bem definidos no mapa de palavras e no de frases, a aproximação realizada pelo reconhecimento de frases ocorre normalmente.

Perspectivas

O protótipo apresentado neste trabalho permite uma ampliação e melhoramento de seu desempenho prático utilizando-se ferramentas que permitam a determinação dos classes semânticas. Somente então será possível afirmar-se a existência de uma auto-classificação semântica.

As referidas ferramentas seriam esquematizações de situações vivenciadas, bem como as noções de crenças e valores associados a estas situações. Sem a atribuição de uma escala de importância aos fatos aos quais a linguagem refere-se não é possível a determinação de seu significado. O presente trabalho não trata deste

tema, utilizando uma determinação de significado supervisionada pelo usuário. A tendência atual de construção de sistemas para reconhecimento de linguagem vai no sentido da união dos paradigmas simbólico e subsimbólico. Desta forma, seriam necessárias ferramentas simbólicas que complementassem o contexto deste trabalho.

O protótipo desenvolvido poderia ainda ter pequenas mas significantes ampliações, tais como:

- reconhecimento de palavras com letras trocadas, e, em conseqüência, frases com palavras contendo letras trocadas;
- utilizar o modelo LVQ para refinamento das classes semânticas e, conseqüentemente, melhor índice de reconhecimento de frases inexistentes;
- acoplamento de um sistema de reconhecimento de voz, para utilização através da linguagem falada.

Com estes acréscimos citados e a junção com ferramentas de definição de características do meio para a realização da classificação semântica, talvez atinja-se o sistema ideal para ser utilizado na concepção de um computador com uma real interface entre o meio e a máquina, possibilitando a esta uma verdadeira comunicação.

Apesar desta situação ser a ideal, ainda são necessárias muitas etapas no sentido de formalização do processo de aquisição da linguagem para que se alcance esse estágio de comunicação. Por outro lado, é atualmente possível a realização de sistemas controlados por voz, com o reconhecimento de comandos frasais. A tecnologia atualmente existente, juntamente com o desenvolvimento do protótipo do presente trabalho, permite a concepção de sistemas de eletrônica embarcada para comando por voz. Isto, se desenvolvido no sentido de facilitar a utilização de equipamentos eletrônicos, poderá proporcionar bem estar e uma melhor qualidade de vida às pessoas. Estas não necessitarão adaptar-se às máquinas e sim o contrário.

Enfim, espera-se de que este trabalho possa ser uma contribuição à ampliação de pesquisas na área da comunicação homem-máquina através da evolução de sistemas de RNAs. Sabe-se que este é apenas um passo no sentido da evolução da visão que todos têm acerca do computador e de sua forma de utilização. Esta fronteira da Computação ainda está por ser vislumbrada e ultrapassada.

ANEXO 1 INTERFACE

Visão Geral

A interface construída serviu de plataforma de testes para a visualização dos resultados do protótipo construído, bem como para verificação de uma utilização plausível do mesmo.

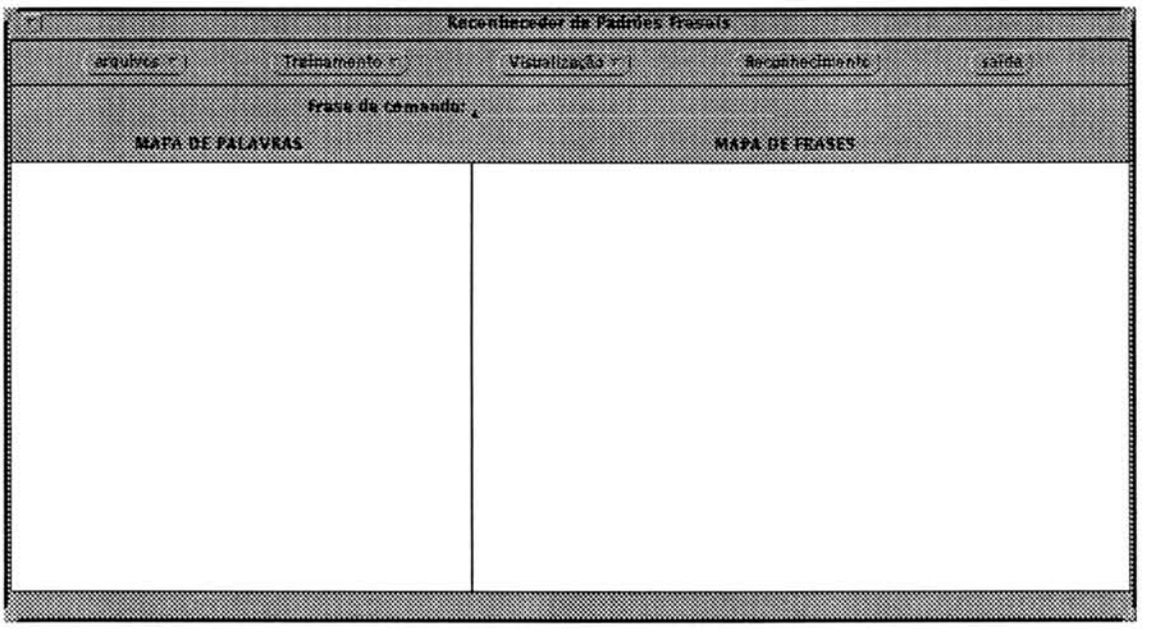
Construída no ambiente de janelas Openwindows versão 3.0, através do utilitário Openwindows Developer's Guide versão 1.1, a interface permite uma fácil utilização do protótipo, através das seguintes características:

- leitura facilitada dos arquivos de palavras e frases a treinar ou reconhecer;
- treinamento de palavras e frases;
- visualização de mapas de palavras e frases;
- reconhecimento de arquivos de frases;
- reconhecimento de comandos frasais da interface.

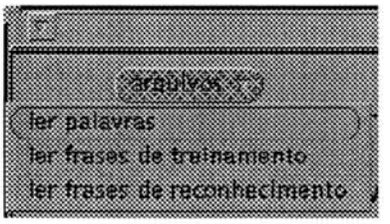
A interface pode ser visualizada na figura a seguir, onde se pode observar a organização de seus recursos, que serão especificados posteriormente.

Leitura

O primeiro passo para utilização da interface é a definição dos arquivos a serem utilizados. Os demais recursos não podem ser utilizados sem o registro do caminho e nome dos arquivos relacionados com eles. Assim, pode ser definido



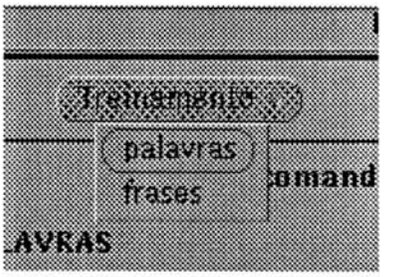
o arquivo de palavras, de frases de treinamento e frases de reconhecimento (veja figura).



A leitura do arquivo de palavras é fundamental para o funcionamento da interface, pois sem ela não é possível a utilização de qualquer outra função. Uma vez feita a leitura, será possível treinar e visualizar o mapa de palavras, além de permitir o treinamento de frases.

Treinamento

Um vez definidos os arquivos, pode-se treinar o arquivo de palavras ou o arquivo de frases (veja figura). Para o treinamento deverão estar editados os arquivos segundo o formato exigido pelo protótipo (ver anexo 3).



Caso sejam treinadas frases, necessariamente terá de ser explicitado o arquivo de palavras, além do arquivo de frases. Isso é necessário para que o protótipo leia o posicionamento das palavras no mapa resultante do treinamento das palavras. Observa-se então a necessidade do treinamento de palavras ser anterior ao das frases.

O treinamento através da interface realiza a chamada automática dos módulos do protótipo, realizando a seqüência necessária da leitura dos arquivos à apresentação dos resultados do mapa. A codificação das palavras é realizada pelo programa *tranp*, o qual colocará o código num arquivo de extensão .cod e .rec. Feito isto, é chamado o programa de treinamento em si, o *tremapa*, que gera, após o tempo determinado pelo arquivo de configuração, um arquivo de extensão .pes.

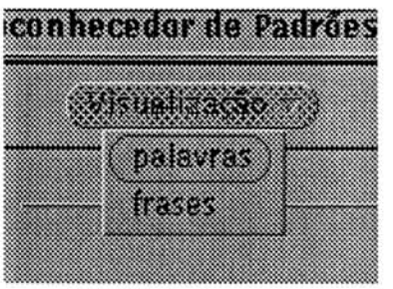
Atenção: *o arquivo de configuração deverá ser editado pelo usuário antes do treinamento!*

Uma vez gerado o arquivo de pesos, o programa *recmapa* realiza o reconhecimento dos padrões originais, para posterior visualização. Esta visualização é realizada automaticamente ao final do processo e apresentada na seção do Mapa de Palavras. O mesmo processo é feito para o treinamento de frases, com exceção do programa *tranf* utilizado no lugar do programa *tranp*.

Visualização

Uma vez feita a leitura dos arquivos, é possível a visualização da distribuição das palavras ou frases no mapa de Kohonen. Como se vê na figura a seguir,

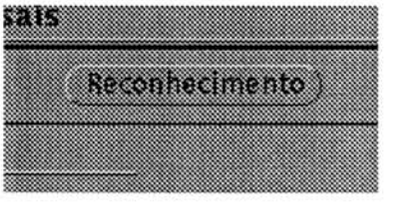
pode-se acionar a visualização dos mapas. Cada palavra ou frase é colocada num ponto de um plano bidimensional, onde se pode verificar a proximidade (ou não) dos padrões de uma mesma classe.



Para a visualização, é lido o arquivo de extensão .res, gerado pelo programa *recmapa*. As palavras são apresentadas integralmente e as frases são truncadas, sendo apresentadas apenas as iniciais de cada letra.

Reconhecimento

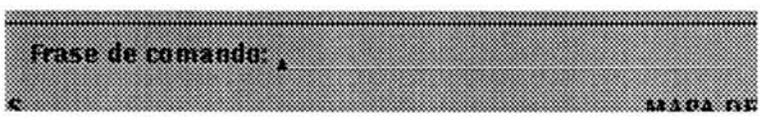
Quando torna-se necessária a verificação de uma frase externa ao conjunto de treinamento, utiliza-se a opção de reconhecimento, como mostrada na figura a seguir. Para tanto, é necessária a edição das frases a reconhecer nos mesmos moldes dos arquivos de treinamento, com a diferença de não necessitar de definição da numeração de classe, a qual é zerada.



Como resultado, são apresentadas as iniciais das frases lidas, dispostas no plano cartesiano. *Caso haja sobreposição de padrões, somente um deles irá ser apresentado.*

Reconhecimento de Comandos Frasais

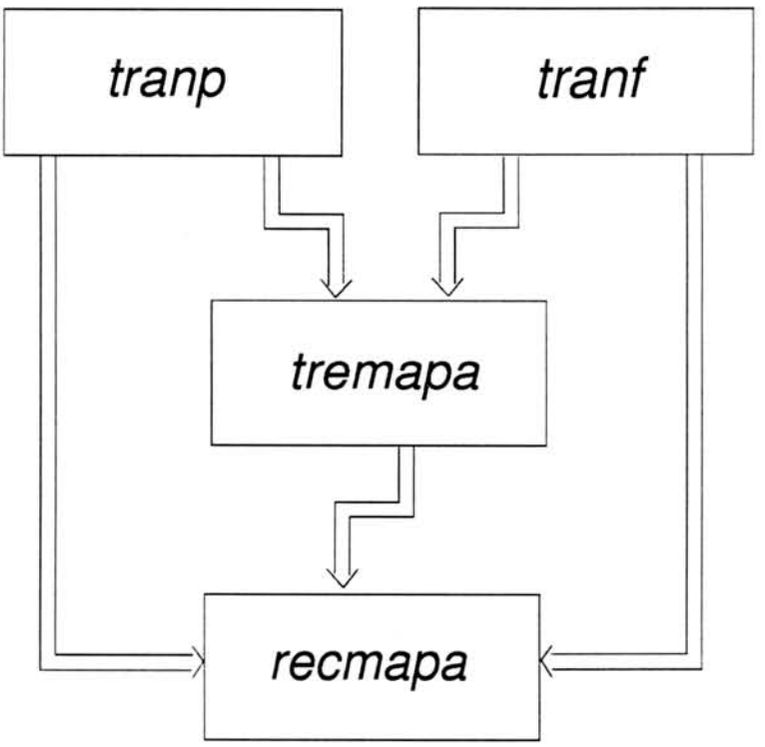
A própria interface do protótipo apresentado é uma aplicação do mesmo. Isso porque ela dispõe de uma linha de comando para o recebimento de comandos na forma de frase (veja figura). A interface possui arquivos com frases pré-definidas (f2, f3...) que possibilitam a utilização das demais funções apresentadas neste capítulo.



Ao ser lida a frase na linha de comando, é criado um arquivo para reconhecimento da frase. Dependendo do tamanho da frase, é escolhido um dos mapas: f2, para frases de duas palavras, f3 para três e assim por diante. Após a tentativa de reconhecimento, será apresentado o comando que mais se aproximar da resposta do mapa. Obviamente, dependendo da frase, não será possível o reconhecimento correto de função em todos os casos. Por outro lado, uma frase escrita corretamente sempre será reconhecida corretamente, executando a função solicitada.

ANEXO 2 MÓDULOS DO PROTÓTIPO

Como apresentado na seção 4.3, o protótipo é composto de quatro módulos básicos, como pode ser observado na figura a seguir. Há dois módulos destinados à codificação de padrões, um para o treinamento dos mapas e outro para o reconhecimento de padrões.



Codificação de Padrões

Os módulos de codificação de padrões têm por objetivo o pré-processamento das entradas do mapa de Kohonen. Isso é uma etapa extremamente importante do processamento, visto que a organização deste modelo de rede neural realiza uma distribuição probabilística das entradas, sendo a composição destas primordial para uma boa classificação. O pré-processamento, portanto, deve levar em conta uma

composição de entradas tal que realmente possibilite a classificação para um posterior reconhecimento com sucesso.

O primeiro módulo - denominado *tranp* - destina-se à atribuição de valores às letras de uma palavra. Estes valores são normalizados entre 0 e 1 e compõem a codificação de cada palavra que será treinada. Juntamente com os valores de cada letra, é acrescentada uma codificação - também normalizada - que corresponde ao campo semântico da palavra em questão. Esta codificação é derivada da orientação do próprio usuário, que deve atribuir um valor de 1 a 10 indicando o campo semântico de cada palavra. *O número de campos semânticos foi limitado a 10, para efeito de implementação.* Ao final, tem-se uma composição de um binômio onde se encontram características da palavra com a associação a seu campo semântico. Este pré-processamento permitirá a distribuição no mapa de Kohonen das características dos padrões com o mesmo campo semântico, ao mesmo tempo que os diferencia através de suas características internas, ou seja, da codificação das letras ou palavras. Outros detalhes do pré-processamento de palavras foram discutidos na seção 4.4.1.

O segundo módulo - *tranf* - permite a codificação numérica das frases, baseando-se no resultado do mapa de palavras. O programa lê as frases e o respectivo número de classe do arquivo editado pelo usuário e as codifica conforme o resultado do mapa de palavras e a classe apresentada. *Como no módulo anterior, o número de campos semânticos foi limitado a 10, para efeito de implementação.* Forma-se, como visto no primeiro módulo, um binômio formado pela representação das características do relacionamento entre palavras resultante do mapa de palavras e pelo código da classe. Como no módulo anterior, este também normaliza os valores entre 0 e 1, permitindo uma melhor adequação ao modelo de rede neural utilizado.

Caso a codificação apresentada seja voltada ao reconhecimento, todo campo de valores de definição de classe é preenchido com números que expressam uma aproximação àqueles valores definidos para as dez classes semânticas possíveis.

Isso permite a aproximação no mapa sem a necessidade da indicação da classe semântica da frase. Mais detalhes veja no apêndice 3.

Com esta organização, espera-se a distribuição no mapa das relações existentes entre as palavras, influenciadas pela definição das classes, resultando numa distribuição de frases conforme a semelhança das palavras (e sua posição na frase) e a classe definida. Outras características deste módulo foram apresentadas na seção 4.4.2.

Ambos os módulos lêem arquivos ASCII contendo palavras ou frases. No caso da codificação de palavras, o usuário deve colocar o número de padrões existentes no arquivo, o número de palavras no arquivo e as palavras juntamente com seu número de classe. No caso de frases, coloca-se no arquivo o número de frases do arquivo, o número máximo de palavras por frase, a frase juntamente com o número de palavras na frase e seu valor de classe. Mais detalhes sobre a edição destes arquivos, veja no apêndice 3.

Treinamento

O módulo de treinamento - *tremapa* - consiste na implementação do modelo de rede neural *Mapa Auto-Organizável* desenvolvido por Kohonen, conforme descrito na seção 4.2. Neste módulo, o modelo foi adaptado para permitir até 21 neurônios na camada de entrada e 200 na camada de saída. Assim, a codificação está limitada a 16 letras por palavra e 8 palavras por frase.

O programa desenvolvido lê primeiramente o arquivo de configuração, do qual recebe os valores para os parâmetros do modelo neural. São eles:

- dimensão das abscissas do mapa;
- dimensão das ordenadas do mapa;

- dimensão do vetor de entradas;
- tamanho inicial do raio de cálculo;
- coeficiente de aprendizado;
- tempo máximo;
- número de padrões a ler do arquivo editado pelo usuário.

Após lidos estes valores, é acessado o arquivo de padrões a treinar, já pré-processados por um dos módulos de codificação.

Ao final do processamento dos dados fornecidos, este módulo gera um arquivo de pesos com os coeficientes utilizados pelo mapa de Kohonen. Para obtenção dos resultados, é necessária a utilização do módulo de reconhecimento *recmapa*.

Reconhecimento

O módulo de reconhecimento - *recmapa* - tem por finalidade a obtenção dos resultados do módulo de treinamento. Após lido o arquivo de configuração, é acessado o arquivo de pesos do mapa correspondente, o arquivo com o código para reconhecimento, e é então gerado um arquivo de resultados (posição no mapa). O arquivo com o código de reconhecimento segue os mesmos passos do arquivo com o código de treinamento, ou seja, o usuário deve editar o arquivo com as palavras ou frases a reconhecer e utilizar um dos módulos de codificação para transformá-lo em valores, os quais serão compilados por este módulo. O resultado final é um arquivo com a posição no mapa da palavras ou frase apresentada. O arquivo de resultados possui as posições do mapa na mesma ordem em que os padrões de reconhecimento estão editados.

A composição básica deste módulo consiste num segmento do modelo de Kohonen, voltado apenas para a obtenção dos resultados, baseado no neurônio vencedor. Mais detalhes do modelo veja na seção 4.2.

ANEXO 3 LISTAGEM

Programa *tremapa*

```
/*{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}

Programa tremapa.c      -  dissertacao de mestrado  -  Daniel Muller

Objetivo: este programa utiliza o metodo SOM de RNAs para a
classificacao dos padroes semanticos de palavras e frases.

Funcionamento: necessita de um arquivo de padroes no formato

#numero_do_padrao peso_do_padrao
valor1 valor2 ... valor_nent

Ex.:
#1 0.000000
0.09 0.09 0.01 0.09 0.09
0.106600 0.101400 0.104000 0.106600 0.090000 0.090000 0.090000

#2 0.000000
etc...

Esta organizacao pode ser provida automaticamente pelo programa
tranp e tranf.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}*/
```

```
#include <stdio.h>
#include <string.h>
#include <math.h>


#define PESA 200
#define DIMX 200
#define ENT 21

double Tmax = 0.0;      /* tempo: numero de epocas */

char arqpad[20];      /* arquivo de padroes */

int nent,      /* numero de entradas */
 dimx, dimy, dimt,         /* dimensoes */
    npad;            /* numero de padroes */

long local;        /* posicao de arquivo */

double peso[PESA],        /* peso da amostra */
       pesos[DIMX][ENT],  /* pesos "codebook" */
       ent[ENT];     /* entradas "code vector" */


/*********************************************************************
*  Rotina: randomico                                                 *
*  Funcao: gerar valores aleatorios                                  *
*  Recebe: nada                                                      *
```

```
*  Retorna: valores randomicos                                          *
************************************************************************/

double randomico()
{
 return(rand()/100001.0);
}

/************************************************************************
*  Rotina: abrearq                                                      *
*  Funcao: abre arquivo de padroes                                      *
*  Recebe: nada                                                         *
*  Retorna: nada  /  ponteiro de arquivo                                *
************************************************************************/

void abrearq()
{
 FILE *arq;

 if((arq = fopen(arqpad,"rt")) == NULL)
  {
fprintf(stderr,"\nArquivo nao encontrado.\n");
exit(-1);
  }
 local = ftell(arq);
 fclose(arq);
}


/************************************************************************
```

```
*  Rotina: learq                                                    *
*  Funcao: ler arquivo de padroes                                   *
*  Recebe: numero do padrao                                         *
*  Retorna: nada  / padroes lidos                                   *
*********************************************************************/

void learq(num)
int num;
{
 FILE *arq;
 char ca;
 int i, n = -1, sai = 1;

 if((arq = fopen(arqpad,"rt")) == NULL)
  {
fprintf(stderr,"\nArquivo de padroes nao encontrado.\n");
exit(-1);
  }
 while(sai)
  {
   do ca = getc(arq); while(ca != '#');
   fscanf(arq,"%d",&n);
   fscanf(arq,"%lf",&peso[num]);
   if(n > num) { if (fseek(arq,local,0) != 0)  exit(-1); }
    else if(n == num) sai = 0;
  }

 for(i = 0; i < nent; i++) fscanf(arq,"%lf",&ent[i]);

 fclose(arq);
```

```
}

/*********************************************************************
*  Rotina: inipes                                                    *
*  Funcao: inicializacao randomica de pesos                          *
*  Recebe: nada                                                      *
*  Retorna: nada / pesos inicializados                               *
*********************************************************************/

void inipes()
{
 register int i, j;

 for(i = 0; i < dimt; i++)
  for(j = 0; j < nent; j++)
if(j%2==0) pesos[i][j] = randomico();
 else pesos[i][j] = -randomico();
}



/*********************************************************************
*  Rotina: gravapes                                                  *
*  Funcao: grava arquivo de pesos                                    *
*  Recebe: nada                                                      *
*  Retorna: nada                                                     *
*********************************************************************/

void gravapes(arqpes)
```

```
char *arqpes;
{
 FILE *arq;
 int i, j;

 arq = fopen(arqpes,"w");

 for(i = 0; i < dimt; i++)
  for(j = 0; j < nent; j++)
   fprintf(arq,"%lf ",pesos[i][j]);

 fclose(arq);
}


/*********************************************************************
*  Rotina: disret                                                    *
*  Funcao: calculo da distancia retangular                           *
*  Recebe: dois pontos cartesianos                                   *
*  Retorna: modulo da diferenca                                      *
*********************************************************************/

double disret(x1,y1,x2,y2)
double x1, x2, y1, y2;
{
 double dif, dist;

 dif = x1 - x2;
 dist = dif * dif;
 dif = y1 - y2;                 /* modulo da diferenca  */
```

```
 dist += dif * dif;              /* (distancia retangular) */
 dist = sqrt(dist);
 return(dist);
}

double dis(ind1,ind2)
int ind1, ind2;
{
 int dif, dist;

 dif = ind1 - ind2;               /* modulo da diferenca  */
 dist = dif * dif;               /* (distancia vetorial) */
 dist = sqrt(dist);
 return(dist);
}

/*********************************************************************
*  Rotina: gaussiana                                                 *
*  Funcao: calculo da funcao gaussiana                               *
*  Recebe: fator alfa, raio, coordenadas, tempo                      *
*  Retorna: nada / altera pesos                                      *
*********************************************************************/

void gaussiana(alfa, raio, x1, y1)
double alfa, raio, x1, y1;
{
 double x2, y2, dist, nucleo;
 register int i, j;

 for(i = 0; i < dimt; i++)
```

```
  {
x2 = (double)(i % dimx);
y2 = (double)(i) / (double)(dimx);

dist = disret(x1,y1,x2,y2);
nucleo = alfa * exp(-dist * dist / (2.0 * raio * raio));
   for(j = 0; j < nent; j++)
     pesos[i][j] += nucleo * (ent[j] - pesos[i][j]);
  }
}


/*********************************************************************
*  Rotina: ganhador                                                  *
*  Funcao: encontrar o neuronio vencedor                             *
*  Recebe: nada                                                      *
*  Retorna: nodo vencedor                                            *
*********************************************************************/

int ganhador()
{
 register int i, k;
 double dif0 = 1e30,
        dif, di, ind;

   for(i = 0; i < dimt; i++)
    {
     dif = 0.0;

     for(k = 0; k < nent; k++)
```

```
      {
       di = pesos[i][k] - ent[k];
 dif += di * di;
 if(dif > dif0) break;
      }

     if(dif < dif0)
      {
       dif0 = dif;
       ind = i;
      }
    }
 return(ind);
}

/*********************************************************************
*  Rotina: ensina_koh                                                *
*  Funcao: algoritmo-base de aprendizado de mapas de caracteristica *
*  Recebe: fator alfa, raio, tamanho, tempo                          *
*  Retorna: nada / mapa ensinado                                     *
*********************************************************************/
void ensina_koh(alfa, raio)
double alfa, raio;
{
 double decraio, decalfa,
  med = 0.5/raio,
        tmp, x, y;
 int ind,
  np = 1;
```

```
 double tempo, t = 1.0;

 for(tempo = 0.0; tempo < Tmax; tempo++)
  {
if(t == 10000) { printf("# "); t = 1; } else t++;
   if(np > npad) np = 1;  /* para leitura de padroes */
learq(np);
np++;

tmp = (double)(tempo) / (double)(Tmax);
decraio = raio * pow(med,tmp);
decalfa = alfa * (Tmax - tempo) / (double) Tmax;

if (peso[np-1] > 0.0)
 decalfa = 1.0 - pow((1.0 - decalfa), peso[np-1]);

ind = ganhador();
x = (double)(ind % dimx);
y = (double)(ind) / (double)(dimx);
gaussiana(decalfa,decraio,x,y);
  }
}

/*********************************************************************
*  Rotina: aprende                                                   *
*  Funcao: le dados para aprendizado                                 *
*  Recebe: nada                                                      *
*  Retorna: nada / padroes aprendidos                                *
*********************************************************************/
```

```
void aprende(argv)
char *argv[];
{
 double raio, alfa;
 FILE *arq;

 if((arq = fopen(argv[3],"rt")) == NULL)
  {
printf("\n dimensao X: ");
scanf("%d",&dimx);

printf("\n dimensao Y: ");
scanf("%d",&dimy);

printf("\n Dimensao vetor de entradas: ");
scanf("%d",&nent);

printf("\n raio: ");
scanf("%lf",&raio);

printf("\n alfa: ");
scanf("%lf",&alfa);

printf("\n tempo: ");
scanf("%lf",&Tmax);

printf("\n numero de padroes: ");
scanf("%d",&npad);
  }
  else
```

```
  {
fscanf(arq,"%d",&dimx);
fscanf(arq,"%d",&dimy);
fscanf(arq,"%d",&nent);
fscanf(arq,"%lf",&raio);
fscanf(arq,"%lf",&alfa);
fscanf(arq,"%lf",&Tmax);
fscanf(arq,"%d",&npad);
  }

 dimt = dimx * dimy;

 abrearq();
 inipes();
 ensina_koh(alfa,raio);
 gravapes(argv[2]);
}


/*************************************************************/

void main(argc,argv)
int argc;
char *argv[];
{
 nent = 21;
 npad = 10;              /* inicializacoes default */
 dimx = 7;
 dimy = 7;
 dimt = dimx * dimy;
```

```
 if(argc < 3)
  {
    fprintf(stderr,"Uso: tremapa <arq.padroes> <arq.pesos>
<arq.config>\n");
    return;
  }
 printf(":::: Mapa para reconhecimento semantico - aprendizado
 ::::");
 sprintf(arqpad,argv[1]);
  aprende(argv);
}
```

Programa *tranf*

```
/*{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}

Programa tranf.c     -  dissertacao de mestrado  -  Daniel Muller

Objetivo: este programa realiza a codificacao numerica de frase
baseado nos resultados do mapa de palavras.

Funcionamento: necessita de um arquivo de padroes no formato
```

```
$numero_de_palavras   maior_tamanho_frase

numero_palavras_frase1 campo_da_frase1  frase1
numero_palavras_frase2 campo_da_frase2  frase2
...

Ex.:
$2 4
4 2 grave arquivo de edicao
2 1 abra janela

Este arquivo deve ser editado diretamente pelo usuario. Ha ainda a
necessidade de um arquivo que contenha os resultados do mapa de
palavras no formato:

palavra1 posicao1x posicao1y
palavra2 posicao2x posicao2y
...

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{*/

#include <stdio.h>

#define NMAX 200
#define TPAL 10

typedef struct
 {
  char pal[20];
```

```
  int x,
      y;
 } MODULO;

MODULO mod[NMAX];
int campo[NMAX], np;



/*************************************************************/

void poenum(num,arqsai)
int num;
FILE *arqsai;
{
 switch(num)
  {

case 1: fprintf(arqsai,"%.2f %.2f %.2f %.2f %.2f ",
0.09,0.09,0.09,0.09,0.07); break;
case 2: fprintf(arqsai,"%.2f %.2f %.2f %.2f %.2f ",
0.09,0.09,0.09,0.07,0.09); break;
case 3: fprintf(arqsai,"%.2f %.2f %.2f %.2f %.2f ",
0.09,0.09,0.07,0.09,0.09); break;
case 4: fprintf(arqsai,"%.2f %.2f %.2f %.2f %.2f ",
0.09,0.07,0.09,0.09,0.09); break;
case 5: fprintf(arqsai,"%.2f %.2f %.2f %.2f %.2f ",
0.07,0.09,0.09,0.09,0.09); break;
case 6: fprintf(arqsai,"%.2f %.2f %.2f %.2f %.2f ",
0.09,0.09,0.09,0.07,0.07); break;
```

```
case 7: fprintf(arqsai,"%.2f %.2f %.2f %.2f %.2f ",
0.09,0.09,0.07,0.09,0.07); break;
case 8: fprintf(arqsai,"%.2f %.2f %.2f %.2f %.2f ",
0.09,0.07,0.09,0.09,0.07); break;
case 9: fprintf(arqsai,"%.2f %.2f %.2f %.2f %.2f ",
0.07,0.09,0.09,0.09,0.07); break;
case 10: fprintf(arqsai,"%.2f %.2f %.2f %.2f %.2f ",
0.09,0.09,0.07,0.07,0.07); break;

default: fprintf(stderr,"Campo semantico invalido.\n");
exit(-1);
  }
 fprintf(arqsai,"\n");
}


/*************************************************************/

void poenum2(arqsai)
FILE *arqsai;
{
 fprintf(arqsai,"0.080 0.080 0.080 0.080 0.080 \n");
}


/*************************************************************/

void learq1(argv)
char *argv[];
{
```

```
 FILE *arqent;

 if((arqent = fopen(argv[3],"rt")) == NULL)
  {
fprintf(stderr,"Erro no arquivo de padroes 'pa.ven'.\n");
exit(-1);
  }

 np = -1;
 while(np < NMAX || !(feof(arqent)))
  {
np++;
fscanf(arqent,"%s",mod[np].pal);   /* le o arquivo com os
padroes */
fscanf(arqent,"%d",&mod[np].x);
fscanf(arqent,"%d",&mod[np].y);
  }
 fclose(arqent);
}


/*************************************************************/

void learq2(argv)
char *argv[];
{
 FILE *arqent;
 FILE *arqsai;
 int i, j, k, l, n = 1,
  nf, /* num. frases */
```

```
  maior,  /* maior tamanho de frase */
  npal, /* num. palavras (por frase) */
  classe;

 char c, pal[20];

 float xx, yy;


 if((arqent = fopen(argv[1],"rt")) == NULL)
  {
fprintf(stderr,"Erro no arquivo origem.\n");
exit(-1);
  }

 if((arqsai = fopen(argv[2],"wt")) == NULL)
  {
fprintf(stderr,"Erro no arquivo destino.\n");
exit(-1);
  }

  /* le o arquivo com as frases */

fscanf(arqent,"%c",&c);
fscanf(arqent,"%d",&nf);
fscanf(arqent,"%d",&maior);

 if (argv[4] != NULL) if (strcmp(argv[4],"-r") == 0)
  fprintf(arqsai,"$%d \n\n",nf); /* indica quantas frases estao
codificadas */
```

```
for (i = 0; i < nf; i++)
 {
  fscanf(arqent,"%d",&npal);
  fscanf(arqent,"%d",&classe);
  if (argv[4] == NULL)
{
 fprintf(arqsai,"%s","#");
 fprintf(arqsai,"%d %f\n",n++,0.0);
}

if (argv[4] == NULL) poenum(classe,arqsai);   /* coloca
codigo da classe semantica */
         else if (strcmp(argv[4],"-r") == 0) poenum2(arqsai);


for(j = 0; j < npal; j++)
{
 fscanf(arqent,"%s",pal);
 for(k = 0; k < np; k++)
if (strcmp(pal,mod[k].pal) == 0)
  {
xx =  mod[k].x / 100.0;
yy =  mod[k].y / 100.0;
fprintf(arqsai,"%.3f ",xx);
fprintf(arqsai,"%.3f ",yy);
k = np;
  }
                        else if (k == np-1)
      {
```

```
fprintf(stderr,"Palavra '%s' fora
do vocabulario.\n",pal);
exit(-1);
        }

}
 if (npal < maior)
  for(l = 0; l < maior-npal; l++) fprintf(arqsai,
"0.090 0.090 ");

 fprintf(arqsai,"\n\n");
}
 fclose(arqsai);
 fclose(arqent);
}


/*************************************************************/

void main(argc,argv)
int argc;
char *argv[];
{
 int i;

 if(argc < 4)
  {
   fprintf(stderr,"Uso: tranf <arq.origem> <arq.destino> <res.mapa>
[-r]\n");
return;
```

```
  }

 for(i = 0; i < NMAX; i++) campo[i] = 0;  /* inicializa campos */
 learq1(argv);
 learq2(argv);
}
```

Programa *tranp*

```
/*{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}

Programa tranp.c     -  dissertacao de mestrado  -  Daniel Muller

Objetivo: este programa realiza a codificacao numerica de palavras
baseado na porcentagem de pontos no bitmap de cada letra.

Funcionamento: necessita de um arquivo de padroes no formato

$numero_de_palavras

campo_da_palavra1  palavra1
campo_da_palavra2  palavra2
...

Ex.:
```

```
$3
2 abra
1 de
5 arquivo
...
Este arquivo deve ser editado diretamente pelo usuario.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}*/

#include <stdio.h>

#define NMAX 200
#define TPAL 10

FILE *arqsai;
char pa[NMAX][20];
int campo[NMAX];
float cod[NMAX][28];
int nt;


/************************************************************/

void pal()
{
 int i, w;

 for(i = 0; i < nt; i++)
  for(w = 0; w < TPAL; w++)
 switch(pa[i][w])
```

```
  {
/* minusculas */
case 'a': cod[i][w] = 0.1066; break;
case 'b': cod[i][w] = 0.1014; break;
case 'c': cod[i][w] = 0.0650; break;
case 'd': cod[i][w] = 0.1092; break;
case 'e': cod[i][w] = 0.0676; break;
case 'f': cod[i][w] = 0.0598; break;
case 'g': cod[i][w] = 0.0806; break;
case 'h': cod[i][w] = 0.0832; break;
case 'i': cod[i][w] = 0.0312; break;
case 'j': cod[i][w] = 0.0572; break;
case 'k': cod[i][w] = 0.0910; break;
case 'l': cod[i][w] = 0.0494; break;
case 'm': cod[i][w] = 0.0936; break;
case 'n': cod[i][w] = 0.0858; break;
case 'o': cod[i][w] = 0.1144; break;
case 'p': cod[i][w] = 0.0728; break;
case 'q': cod[i][w] = 0.0962; break;
case 'r': cod[i][w] = 0.1040; break;
case 's': cod[i][w] = 0.0754; break;
case 't': cod[i][w] = 0.0546; break;
case 'u': cod[i][w] = 0.0780; break;
case 'v': cod[i][w] = 0.0884; break;
case 'w': cod[i][w] = 0.0900; break;
case 'x': cod[i][w] = 0.0624; break;
case 'y': cod[i][w] = 0.0468; break;
case 'z': cod[i][w] = 0.1118; break;
/* MAIUSCULAS */
case 'A': cod[i][w] = 0.1066; break;
```

```
case 'B': cod[i][w] = 0.1014; break;
case 'C': cod[i][w] = 0.0650; break;
case 'D': cod[i][w] = 0.1092; break;
case 'E': cod[i][w] = 0.0676; break;
case 'F': cod[i][w] = 0.0598; break;
case 'G': cod[i][w] = 0.0806; break;
case 'H': cod[i][w] = 0.0832; break;
case 'I': cod[i][w] = 0.0312; break;
case 'J': cod[i][w] = 0.0572; break;
case 'K': cod[i][w] = 0.0910; break;
case 'L': cod[i][w] = 0.0494; break;
case 'M': cod[i][w] = 0.0936; break;
case 'N': cod[i][w] = 0.0858; break;
case 'O': cod[i][w] = 0.1144; break;
case 'P': cod[i][w] = 0.0728; break;
case 'Q': cod[i][w] = 0.0962; break;
case 'R': cod[i][w] = 0.1040; break;
case 'S': cod[i][w] = 0.0754; break;
case 'T': cod[i][w] = 0.0546; break;
case 'U': cod[i][w] = 0.0780; break;
case 'V': cod[i][w] = 0.0884; break;
case 'W': cod[i][w] = 0.0900; break;
case 'X': cod[i][w] = 0.0624; break;
case 'Y': cod[i][w] = 0.0468; break;
case 'Z': cod[i][w] = 0.1118; break;
/* se nao for letra, "zera" */
default: cod[i][w] = 0.09;
  }
}
```

```
/*****************************************************************/

void escreve(indice)
int indice;
{
 int i;
 for(i = 0; i < TPAL; i++) fprintf(arqsai,"%f ",cod[indice][i]);
 fprintf(arqsai,"\n");
}



/*****************************************************************/

void poenum(num)
int num;
{
 switch(num)
  {
  case 1: fprintf(arqsai,"%f %f %f %f %f ",
0.09,0.09,0.09,0.09,0.01); break;
  case 2: fprintf(arqsai,"%f %f %f %f %f ",
0.09,0.09,0.09,0.01,0.09); break;
  case 3: fprintf(arqsai,"%f %f %f %f %f ",
0.09,0.09,0.01,0.09,0.09); break;
  case 4: fprintf(arqsai,"%f %f %f %f %f ",
0.09,0.01,0.09,0.09,0.09); break;
  case 5: fprintf(arqsai,"%f %f %f %f %f ",
```

```
0.01,0.09,0.09,0.09,0.09); break;
  case 6: fprintf(arqsai,"%f %f %f %f %f ",
0.09,0.09,0.09,0.01,0.01); break;
  case 7: fprintf(arqsai,"%f %f %f %f %f ",
0.09,0.09,0.01,0.09,0.01); break;
  case 8: fprintf(arqsai,"%f %f %f %f %f ",
0.09,0.01,0.09,0.09,0.01); break;
  case 9: fprintf(arqsai,"%f %f %f %f %f ",
0.01,0.09,0.09,0.09,0.01); break;
  case 10: fprintf(arqsai,"%f %f %f %f %f ",
0.09,0.09,0.01,0.01,0.01); break;
  default: fprintf(stderr,"Campo semantico invalido.\n"); exit(-1);
  }
 fprintf(arqsai,"\n");
}


/*************************************************************/

void learq(argv)
char *argv[];
{
 char cifrao;
 FILE *arqent;
 int i, j;

 for(j = 0; j < TPAL; j++)
  for(i = 0; i < NMAX-3; i++)
   pa[i][j] = NULL;
```

```
 if((arqent = fopen(argv[1],"rt")) == NULL)
  {
fprintf(stderr,"Erro no arquivo origem.\n");
exit(-1);
  }
 fscanf(arqent,"%c",&cifrao);
 fscanf(arqent,"%d",&nt);


 for(i = 0; i < nt; i++)
  {
fscanf(arqent,"%d",&campo[i]);
fscanf(arqent,"%s",pa[i]);   /* le o arquivo com as frases */
  }
 fclose(arqent);
}

/*************************************************************/

void gravarq(argv)
char *argv[];
{
 int i, n = 1;

 if((arqsai = fopen(argv[2],"wt")) == NULL)
  {
fprintf(stderr,"Erro no arquivo destino.\n");
exit(-1);
  }
```

```
 if (argv[3] != NULL) if (strcmp(argv[3],"-r") == 0)
  fprintf(arqsai,"$%d \n\n",nt); /* indica quantas palavras estao
codificadas */

 for(i = 0; i < nt; i++)  /* faz a codificacao */
  {
if (argv[3] == NULL)
 {
  fprintf(arqsai,"%s","#");
  fprintf(arqsai,"%d %f\n",n++,0.0);
    }
poenum(campo[i]);
escreve(i);
   fprintf(arqsai,"\n");
  }
 fclose(arqsai);
}



/*************************************************************/

void main(argc,argv)
int argc;
char *argv[];
{
 int i;

 if(argc < 3)
  {
```

```
fprintf(stderr,"Uso: tranp <arq.origem> <arq.destino>
[-r]\n");
return;
  }

 for(i = 0; i < NMAX; i++) campo[i] = 0;  /* inicializa campos */
 learq(argv);
 pal();
 gravarq(argv);
}
```

Programa *recmapa*

```
/*{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}

Programa recmapa.c     -  dissertacao de mestrado  -  Daniel Muller

Objetivo: este programa utiliza o metodo SOM de RNAs para o
reconhecimento dos padroes semanticos de palavras e frases.

Funcionamento: necessita de um arquivo de padroes no formato

$numero_de_padroes

valor1 valor2 ... valor_nent
```

```
valor1...

Ex.:
$1

0.09 0.09 0.01 0.09 0.09
0.106600 0.101400 0.104000 0.106600 0.090000 0.090000 0.090000

Esta organizacao pode ser provida automaticamente pelo programa
tranp e tranf.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}*/

#include <stdio.h>
#include <string.h>
#include <math.h>



#define DIMX 200
#define ENT 21
#define NP 40

int nent,     /* numero de entradas */
 dimx, dimy, dimt,        /* dimensoes */
 nt,     /* numero de padroes */
 ven[NP];     /* vetor de vencedores */

double pesos[DIMX][ENT], /* pesos "codebook" */
 ent[NP][ENT];     /* entradas "code vector" */
```

```
/*************************************************************************
*  Rotina: ganhador                                                      *
*  Funcao: encontrar o neuronio vencedor                                 *
*  Recebe: nada                                                          *
*  Retorna: nada / nodos vencedores                                      *
*************************************************************************/

void ganhador()
{
 register int i, k, l;
 double dif0, dif, di, ind;

 for(l = 0; l < nt; l++)
  {
dif0 = 1e10;
for(i = 0; i < dimt; i++)
 {
  dif = 0.0;
  for(k = 0; k < nent; k++)
{
 di = pesos[i][k] - ent[l][k];
 dif += di * di;
 if(dif > dif0) break;
}
  if(dif < dif0)
{
 dif0 = dif;
 ind = i;
```

```
}
 }
ven[l] = ind;
  }
}
```

```
/*********************************************************************
*  Rotina: mostra                                                    *
*  Funcao: apresenta o resultado                                    *
*  Recebe: nada                                                      *
*  Retorna: nada  / resultado                                        *
*********************************************************************/

void mostra(argv)
char *argv[];
{
 FILE *arq, *arqp;
 int i, x1, y1, ifim, d;
 char pal[NP][20], c;

 if (argv[5] != NULL)
  {
if((arqp = fopen(argv[5],"rt")) == NULL)
 {
fprintf(stderr,"Erro no arquivo de palavras.\n");
exit(-1);
 }
fscanf(arqp,"%c",&c);
```

```
fscanf(arqp,"%d",&d);
for(i = 0; i < nt; i++)
 {
  fscanf(arqp,"%d",&d);
  fscanf(arqp,"%s",pal[i]);
 }
fclose(arqp);
  }

 ganhador();

 if((arq = fopen(argv[2],"wt")) == NULL)
  {
fprintf(stderr,"Erro na abertura do arquivo de respostas.\n");
exit(-1);
  }

 for(i = 0; i < nt; i++)
  {
x1 = ven[i] % dimx;
y1 = ven[i] / dimx;
if (argv[5] != NULL) fprintf(arq,"%s ",pal[i]);
fprintf(arq,"%d %d\n\n",x1,y1);
  }
 fclose(arq);
}


/*********************************************************************
*  Rotina: learq                                                     *
```

```
*  Funcao: ler arquivo de padroes                                    *
*  Recebe: nada                                                      *
*  Retorna: nada  / padroes lidos                                    *
*********************************************************************/

void learq(argv)
char *argv[];
{
 FILE *arq;
 char cifrao;
 int i, j;

 if((arq = fopen(argv[1],"rt")) == NULL)
  {
fprintf(stderr,"Erro no arquivo de reconhecimento.\n");
exit(-1);
  }
 fscanf(arq,"%c",&cifrao);
 fscanf(arq,"%d",&nt);

 for(i = 0; i < nt; i++)      /* le o arquivo com os valores */
 {
  for(j = 0; j < nent; j++)
fscanf(arq,"%lf",&ent[i][j]);
 }
 fclose(arq);
}


/*********************************************************************
```

```
*  Rotina: lepes                                                    *
*  Funcao: ler arquivo de pesos                                     *
*  Recebe: nada                                                     *
*  Retorna: nada  / padroes lidos                                   *
*********************************************************************/

void lepes(argv)
char *argv[];
{
 FILE *arq;
 int i, j;

  if((arq = fopen(argv[4],"rt")) == NULL)
{
 printf("\n Dimensao X: ");
 scanf("%d",&dimx);

 printf("\n Dimensao Y: ");
 scanf("%d",&dimy);

 printf("\n Dimensao vetor de entradas: ");
 scanf("%d",&nent);

}
else
{
 fscanf(arq,"%d",&dimx);
 fscanf(arq,"%d",&dimy);
 fscanf(arq,"%d",&nent);
 fclose(arq);
```

```
}

 if((arq = fopen(argv[3],"rt")) == NULL)
  {
fprintf(stderr,"\nArquivo de pesos nao encontrado.\n");
exit(-1);
  }

 dimt = dimx * dimy;

 for(i = 0; i < dimt; i++)
  for(j = 0; j < nent; j++)
fscanf(arq,"%lf",&pesos[i][j]);

 fclose(arq);
}


/*****************************************************************/

void main(argc,argv)
int argc;
char *argv[];
{
 if(argc < 5)
  {
  fprintf(stderr,"Uso: recmapa <arq.rec.> <arq.dest.> <arq.pes>
<arq.cfg> [arq.pal.]\n");
  return;
  }
```

```
 nent = 13;
 dimx = 10;
 dimy = 16;
 dimt = dimx * dimy;

 lepes(argv);
 learq(argv);
 mostra(argv);
}
```

BIBLIOGRAFIA

[AMA 72] AMARI, S. Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements. **IEEE Transactions on Computers**, New York, v. C-21, p. 1197-1206, Nov. 1972.

[ANB 75] ANDERSON, B. F. **Cognitive Psychology - The Study of Knowing, Learning and Thinking.** New York: Academic Press, 1975.

[AND 68] ANDERSON, J. A. A Memory Model Using Spatial Correlation Functions. **Kybernetik**, [S.l.], n. 5, p. 113-119, June 1968.

[BEH 92] BEHAR, P. A. **Psicologia Cognitiva & Aprendizagem Computadorizada.** Porto Alegre: CPGCC da UFRGS, 1992. (Trabalho Individual n. 299).

[CHA 80] CHANGEUX, J. Genetic Determinism and Epigenisis of the Neuronal Network: Is There a Biological Compromise between Chomsky and Piaget? In: PIATTELLI-PALMARINI, M. **Language and Learning**: The Debate between Jean Piaget and Noam Chomsky. Cambridge: Harvard University, 1980.

[CLA 90] CLARK, H. H.; CLARK, E. V. **Psychology and Language**: An Introduction to Psycholinguistics. New York: Harcourt Brace Jovanovich, 1977. 608p.

[COR 93] CORBALLIS, M. C. A Gesture in the right direction? In: DUNBAR, R. I. M. Coevolution of neocortical size, group size and language in humans. **Behavioral and Brains Sciences**, Cambridge, n. 16, p. 697, 1993.

[CRI 86] CRICK, F.; ASANUMA, C. Certain Aspects of the Anatomy and Physiology of the Cerebral Cortex. In: RUMELHART, D. E.;

MCCLELLAND, J. L. **Parallel Distributed Processing**: Explorations in the Microstructure of Cognition. Cambridge: MIT Press, 1986.

[DEN 91] DENIS, F. A. R. M.; MACHADO, R. J. **O Modelo Conexionista Evolutivo.** Rio de Janeiro: IBM - Centro Tecnológico, 1991. (Relatório Técnico CCR-128).

[DUN 93] DUNBAR, R. I. M. Coevolution of neocortical size, group size and language in humans. **Behavioral and Brains Sciences**, Cambridge, n. 16, p. 681-735, 1993.

[EBE 90] EBERHART, R.; DOBBINS, R. **Neural Networks PC Tools**: A Practical Guide. San Diego: Academic Press, 1990. 414p.

[ELM 91] ELMAN, J. L. Distributed representations, simple recurrent networks, and gramatical structure. **Machine Learning**, Dordrecht, Netherlands, n. 7, p. 195-225, 1991.

[FAR 89] FARIA, A. R. **O pensamento e a linguagem da criança segundo Piaget.** São Paulo: Ática, 1989. 80p.

[FUK 75] FUKUSHIMA, K. Cognitron: A Self-Organizing Multilayered Neural Netword. **Biological Cybernetics**, [S.l.], n. 20, p. 121-136, 1975.

[GRO 69] GROSSBERG, S. Embedding Fields: A Theory of Learning with Physiological Implications. **Journal of Mathematical Psychology**, [S.l.], n. 6, p. 209-239, 1969.

[HAR 89] HARRIS, C. J.; ELMAN, J. L. Representing variable information with simple recurrent networks. In: ANNUAL CONFERENCE OF THE COGNITIVE SCIENCE SOCIETY, 11., 1989, [S.l.]. **Proceedings....** Hillsdale: Erlbaum, 1989. p. 435-642.

[HEB 49] HEBB, D. O. **The Organization of Behavior.** New York: Wiley, 1949.

[HEC 90] HECHT-NIELSEN, R. **Neurocomputing.** Reading: Addison-Wesley, 1990. 433p.

[HER 91] HERTZ, J.; KROGH, A. S.; PALMER, R. G. **Introduction to the theory of neural computation.** Reedwood: Addison-Wesley, 1991. 327p.

[HIN 90] HINTON, G. E. Connectionist Learning Procedures. In: **Machine Learning - Paradigms and Methods.** Cambridge: MIT Press, 1990.

[HOP 82] HOPFIELD, J. J. Neural Networks and Physical Systems with Emergent Colective Computacional Abilities. **Proceedings of the National Academy of Sciences**, [S.l.], n. 79, p. 2554-2558, 1982.

[INH 80] INHELDER, B. Language and Knowledge in a Construtivist Framework. In: PIATTELLI-PALMARINI, M. **Language and Learning**: The Debate between Jean Piaget and Noam Chomsky. Cambridge: Harvard University, 1980.

[JAC 86] JACKSON, P. **Introduction to Expert Systems.** Wokingham: Addison-Wesley, 1986. p. 2-11.

[JAI 90] JAIN, A. N.; WEIBEL, A. H. Incremental Parsing by Modular Recurrent Connectionist Networks. In: TOURETZKY, D. S. (Ed.) **Advances in Neural Information Processing Systems 2.** San Mateo: Morgan Kaufmann, 1990. p. 364-371.

[JAM 1890] JAMES, W. **Psychology - Briefer Course.** New York: Holt, 1890.

[JUN 93a] JUNG, C. G. O Homem Arcaico. In: **Civilização em Transição.** Petrópolis: Vozes, 1993. p. 65.

[JUN 93b] JUNG, C. G. O Problema Psíquico do Homem Moderno. In: **Civilização em Transição.** Petrópolis: Vozes, 1993. p. 74-93.

[JUN 93c] JUNG, C. G. Sobre o Inconsciente. In: **Civilização em Transição.** Petrópolis: Vozes, 1993. p. 9-32.

[KOH 72] KOHONEN, T. Correlation matrix memories. **IEEE Transactions on Computers.**, New York, v. C-21, p 353-359, Nov. 1972.

[KOH 77] KOHONEN, T. **Associative Memory.** Berlin: Springer-Verlag, 1977.

[KOH 84] KOHONEN, T. **Self-Organization and Associative Memory.** Berlin: Springer-Verlag, 1984.

[KOH 90] KOHONEN, T. The Self-Organizing Map. **Proceedings of the IEEE**, New York, v. 78, n. 9, p. 1464-1480, Sept. 1990.

[KLA 86] KLAHR, P.; WATERMAN, D. A. **Expert Systems - Techniques, Tools and Applications.** Reading: Addison-Wesley, 1986. p.3-9.

[LAN 89] LANGE, T. E.; DYER, M. G. High-Level Inferencing in a Connectionist Network. **Connection Science,** [S.l.], n. 1, p. 181-127, 1989.

[LIP 87] LIPPMANN, R. P. An Introduction to Computing with Neural Networks. **IEEE ASSP Magazine**, New York, v. 3, n. 4, p. 4-22, Apr. 1987.

[LUR 87] LURIA, A. R.; YUDOVICH, F. Z. **Linguagem e Desenvolvimento Intelectual na Criança.** Porto Alegre: Artes Médicas Sul, 1987.

[MAC 89] MACHADO, R. J.; ROCHA, A. F. **Handling Knowledge In High Order Neural Networks**: The Combinatorial Neural Model. Rio de Janeiro: IBM - Centro Tecnológico, 1989. (Relatório Técnico CCR-076).

[MAC 92] MACHADO, R. J.; FERLIN, C.; ROCHA, A. F. **Combining Semantic and Neural Networks in Expert Systems.** Rio de Janeiro: IBM - Centro Tecnológico, 1992. (Relatório Técnico CCR-128).

[MCC 86] McCLELLAND, J. L.; RUMELHART, D. E.; HINTON, G. E. The Appeal of Parallel Distributed Processing. In: RUMELHART, D. E., MCCLELLAND, J. L.. **Parallel Distributed Processing**: Explorations in the Microstructure of Cognition. Cambridge: MIT Press, 1986. p. 3-44.

[MCC 86a] McCLELLAND, J. L.; KAWAMOTO, A. H. Mechanisms of sentence processing: Assigning roles to constituents. In: Rumelhart, D. E., McClelland, J. L.. **Parallel Distributed Processing**: Explorations in the Microstructure of Cognition. Cambridge: MIT Press, 1986. p. 272-325.

[MCW 43] McCULLOCH, W. S.; PITTS, W. A Logical Calculus of Ideas Immanent in Nervous Activity. **Bulletin of Mathematical Biophysics**, [S.l.], n. 5, p. 115-133, 1943.

[MII 90] MIIKKULAINEN, R. A Distributed Feature Map Model of The Lexicon. In: ANNUAL CONFERENCE OF THE COGNITIVE SCIENCE SOCIETY, 12., 1990, [S.l.]. **Proceedings....** Hilsdale: Erlbaum, 1990. p. 447-454.

[MII91] MIIKKULAINEN, R. **DISCERN**: A Distributed Artificial Neural Network Model of Script Processing and Memory. San Diego: University of California, 1991. PhD Thesis.

[MII93] MIIKKULAINEN, R. **Subsymbolic Natural Language Processing**: An Integrated Model of Scripts, Lexicon, and Memory. Cambridge: MIT Press, 1993. 391p.

[MIN 68] MINSKY, M. **Semantic Information Processing.** Cambridge: MIT Press, 1968.

[MIN 69] MINSKY, M.; PAPERT, S. **Perceptrons.** Cambridge: MIT Press, 1969.

[MOU 90] MOURA, M. L. S. Aquisição de Linguagem: Compreensão e Produção. **Arquivos Brasileiros de Psicologia**, Rio de Janeiro, v. 42, n. 3, p. 50-57, jan/ago 1990.

[MÜL 93] MÜLLER, D. N. **Um Estudo Comparativo Entre As Redes Neurais Artificiais e o Processo de Aquisição da Linguagem Humana**. Porto Alegre: CPGCC-UFRGS, 1993. (Trabalho Individual n. 359).

[MÜL 95] MÜLLER, D. N. **Requisitos para A Reprodução Computacional de Processos Linguísticos A Nível Subsimbólico.** Porto Alegre: CPGCC-UFRGS, 1995. (Trabalho Individual n. 473).

[NEU 58] NEUMANN, J. V. **The Computer and The Brain.** New Haven: Yale University, 1958.

[PES 90] PESSOA, L. A.; REIS, A. Avaliação da representação de schemata em modelos conexionistas. **Revista Brasileira de Computação**, Rio de Janeiro, v. 5, n. 4, abr./jun. 1990.

[PIA 70] PIAGET, J. **O Nascimento da Inteligência na Criança.** Rio de Janeiro: Zahar, 1970.

[PIA 71] PIAGET, J. **A Formação do Símbolo na Criança**: Imitação, jogo e sonho, imagem e representação. Rio de Janeiro: Zahar, 1971. 370p.

[PIA 76] PIAGET, J. **A Equilibração das Estruturas Cognitivas**: O Problema Central do Desenvolvimento. Rio de Janeiro: Zahar, 1976.

[PIA 78] PIAGET, J. A Linguagem e As Operações Intelectuais. In: Problemas de Psicologia Genética. In: **Piaget**. São Paulo: Abril, 1978. p. 264-271. (Os Pensadores).

[PIA 80] PIAGET, J. Schemes of Action and Language Learning. In: PIATTELLI-PALMARINI, M. **Language and Learning**: The Debate between Jean Piaget and Noam Chomsky. Cambridge: Harvard University, 1980. p. 164-167.

[PIA 89] PIAGET, J. **A Linguagem e O Pensamento da Criança.** São Paulo: Martins Fontes, 1989. 212p.

[PIM 80] PIATTELLI-PALMARINI, M. **Language and Learning**: The Debate between Jean Piaget and Noam Chomsky. Cambridge: Harvard University, 1980.

[PIN 84] PINA, F. H. Estudio Semantico/Estrutural de las Frases de Dos Palabras. **Anales de Pedagogía**, Murcia, n. 2, p. 249-261, 1984.

[ROS 58] ROSENBLATT, F. The Perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, [S.l.] n. 65, p. 386-408, 1958.

[RUM 86] RUMELHART, D. E.; MCCLELLAND, J. L. **Parallel Distributed Processing**: Explorations in the Microstructure of Cognition. Cambridge: MIT Press, 1986.

[RUM 86a] RUMELHART, D. E.; MCCLELLAND, J. L. On Learning the Past Tenses of English Verbs. In: RUMELHART, D. E., MCCLELLAND, J. L. **Parallel Distributed Processing**: Explorations in the Microstructure of Cognition. Cambridge: MIT Press, p. 216-271, 1986.

[ROC 92] ROCHA, A. F. et al. A Neural Net for Extracting Knowledge from Natural Language Data Bases. **IEEE Transactions on Neural Networks**, New York, v. 3, n. 5, p. 819-828, Sept. 1992.

[SAN 71] SANDSTRÖM, C. I. **A Psicologia da Infância e da Adolescência.** Rio de Janeiro: Zahar, 1971. 288p.

[SCH 90] SCHRETER, Z. Connectionism: A Link Between Psychology and Neuroscience? In: STENDER, J.; ADDIS, T. **Symbols versus Neurons?** [S.l.]: IOS Press, 1990. p. 152-173.

[SEJ 87] SEJNOWSKY, T. J.; ROSENBERG, C. R. Parallel Networks that Learn to Pronounce English Text. **Complex Systems.** [S.l.], n. 1, p. 145-168, 1987.

[SER 89] SERVAN-SCHREIBER, D.; CLEEREMANS, A.; McCLELLAND, J. L. Learning sequential structure in simple recurrent networks. In: TOURETZKY, D. S. (Ed.). **Advances in Neural Information Processing Systems 1**. San Mateo: Morgan Kaufmann, 1989. p. 643-652.

[SER 91] SERVAN-SCHREIBER, D.; CLEEREMANS, A.; McCLELLAND, J. L. Graded state machines: The representation of temporal contingences in simple recurrent networks. **Machine Learning**, Dordrecht, Netherlands, n. 7, p. 161-194, 1991.

[SMO 86] SMOLENSKY, P. Information Processing in Dynamical Systems: Foundations of Harmony Theory. In: RUMELHART, D. E.; MCCLELLAND, J. L. **Parallel Distributed Processing**: Explorations in the Microstructure of Cognition. Cambridge: MIT Press, 1986. p. 195-281.

[SMO 90] SMOLENSKY, P. Tensor Product Variable Binding and Representation of Symbolic Structures in Connectionist Systems. **Artificial Inteligence**, [S.l.], v. 46, n. 1-2, p. 159-216, Nov. 1990.

[SOU 88] SOUCEK, B.; SOUCEK, M. **Neural and Massively Parallel Computers - The Sixth Generation.** New York: John Wiley & Sons, 1988.

[STJ 90] St. JOHN, M. F.; MCCLELLAND, J. L. Learning and Applying Contextual Constraints in Sentence Comprehension. **Artificial Intellingence**, [S.l.], n. 46, p. 217-258, 1990.

[STJ 92] St. JOHN, M. F. The story gestalt: A model of knowledge-intensive processes in text comprehension. **Cognitive Science**, [S.l.], n. 16, p. 271-306, 1992.

[WID 60] WIDROW, B.; HOFF, M. E. Adaptive switching circuits. **1960 IRE WESCON Convention Record.** New York: IRE, 1960. p. 96-104.

[WID 90] WIDROW, B.; LEHR, M. A. 30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation. **Proceedings of the IEEE**, New York, v. 78, n. 9, p. 1415-1442, Sept. 1990.

[WIL 76] WILLSHAW, D. J.; MALSBURG, C. How Patterned Neural Connections Cam Be Set Um by Self-Organization. **Proceedings of the Royal Society of London**, London, n. 194, p. 431-445, 1976.

[VYG 91] VYGOTSKY, L. S. **Pensamento e Linguagem.** São Paulo: Martins Fontes, 1991. 132p.

Informática UFRGS

CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Reconhecimento Semântico Através de Redes Neurais Artificiais.

por

Daniel Nehme Müller

Dissertação apresentada aos Senhores:

Prof. Dr. Raul Sidnei Wazlawick (UFSC)

Prof. Dr. Dante Augusto Couto Barone

Prof. Dr. Paulo Martins Engel

Vista e permitida a impressão.
Porto Alegre, 02/10/96.

Prof. Dr. Philippe Olivier Alexandre Navaux,
Orientador.

p/ Prof. Flávio Rech Wagner
Coordenador do Curso de Pós Graduação
em Ciência da Computação - CPGCC
Instituto de Informática - UFRGS