UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA ABORDAGEM CONEXIONISTA PARA RESOLUÇÃO DE ANÁFORAS PRONOMINAIS

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO

Itamar Leite de Oliveira

Florianópolis, Fevereiro de 1997

Uma Abordagem Conexionista para Resolução de Anáforas Pronominais

ITAMAR LEITE DE OLIVEIRA

Esta Dissertação foi julgada para a obtenção do título de

MESTRE EM CIÊNCIA DA COMPUTAÇÃO

e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação

Prof. Raul Sidnei Wazlawick, Dr.
Orientador

Prof. Murilo Silva de Camargo, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Raul Sidnei Wazlawick, Dr. (Presidente)

Prof. Vera Lúcia Strube de Lima, Dr.

Prof. Fernando Mendes Azevedo, Dr.

Prof. Luiz Fernando Jacintho Maia, Dr.

Resumo

Neste trabalho foram implementadas redes neurais artificiais visando a resolução de um fenômeno lingüístico conhecido como referência anafórica. Foram resolvidas referências anafóricas pronominais com apenas dois pronomes pessoais, a saber: **ele** e **ela**.

No primeiro experimento, a entrada para a rede corresponde a segmentos de texto compostos de duas sentenças. Na segunda sentença o sujeito é sempre o pronome **ele** ou **ela**. A primeira sentença fornece o contexto para determinar a quem o pronome da segunda sentença está-se referindo. Utilizou-se uma rede recorrente simples para determinar a referência corretamente.

No segundo experimento foi implementado um modelo composto de duas redes neurais: uma rede recorrente simples (*Parser*) e uma rede direta (Segmentador). Estas redes são treinadas e testadas simultaneamente. Com este modelo é possível resolver o mesmo problema do primeiro experimento com segmentos de texto compostos de um número arbitrário de sentenças.

Palavras Chaves

Redes Neurais Artificiais, Processamento de Linguagem Natural (PLN), PLN Conexionista, Processamento Distribuído Paralelo, Representações Conexionistas, Representação Distribuída.

Abstract

This work presents two connectionist approaches to solve a linguistic phenomena called anaphoric reference, using only two pronoun: **he** and **she**.

In the first experiment, the network input was composed by text segments with two sentences. In the second sentence, the subject is always the pronoun **he** or **she**. The first sentence provides the context to establish "to who" the second sentence's pronoun refers. A Simple Recurrent Network (SRN) was used to determine the correct reference.

In the second experiment a model with two neural networks was developed. The networks were: a simple recurrent network (parser) and a feedforward network (segmenter). These networks were trained simultaneously. With this model was possible to solve the problem stated for the first experiment with text segments of an arbitrary size (with a random amount of sentences).

Keywords

Artificial Neural Networks, Natural Language Processing (NLP), Connectionist NLP, Parallel Distributed Processing, Connectionist Representations, Distributed Representations.

Agradecimentos

Aos meus pais pelo apoio e incentivo durante toda a minha vida.

Ao orientador Prof. Raul Sidnei Wazlawick pelas idéias e dedicação ao longo de todo o tempo dispensado na elaboração deste trabalho.

Aos colegas, professores e funcionários da UFSC que, direta ou indiretamente, contribuíram para a realização deste trabalho.

A CAPES pela ajuda financeira concedida através de uma bolsa de estudos.

Em especial à Maristela Correa Meller pelo carinho, paciência e incentivo nos bons e maus momentos que sempre surgem na realização de um trabalho como este.

Sumário

Lista de Figuras

Lista de Tabelas

1. Introdução

A referência anafórica é um fenômeno lingüístico em que um pronome ou um sintagma nominal (NP; veja cap. 3) em uma sentença está se referindo a alguém ou a um objeto previamente mencionado no texto. O problema então é saber a quem o pronome ou o sintagma estão se referindo, uma vez que podem existir vários objetos ou pessoas mencionadas até o momento em que se encontra a referência. Tem-se assim um problema de ambigüidade. No presente trabalho procurou-se resolver tal problema de encontrar o objeto ou a pessoa referenciada utilizando-se da tecnologia de redes neurais artificiais. Serão tratadas apenas referências anafóricas pronominais em um pequeno texto composto de sentenças justapostas. Os pronomes utilizados são os pronomes pessoais **ele** e **ela**. Não foi feito um estudo sobre a complexidade do uso das mesmas estruturas para tratar pronomes oblíquos, como: **o, a, os, as, lhe** e **lhes** e nem um levantamento para saber se os pronomes **ele** e **ela** são mais importantes.

A maioria dos sistemas que lidam com Processamento de Linguagem Natural (PLN) são sistemas desenvolvidos dentro da abordagem simbólica da Inteligência Artificial (IA) ou Inteligência Artificial Simbólica (IAS). Neste trabalho será resolvido o problema de referência mencionado acima dentro da abordagem conexionista da IA ou Inteligência Artificial Conexionista (IAC).

Um sistema com a filosofia da IAC possui todas as características inerentes as Redes Neurais Artificiais (RNA; cap. 2). Pode-se citar algumas características, entre outras, que justificam o uso de RNA na resolução de problemas, não apenas em PLN:

a) Generalização;
b) Robustez e degradação progressiva;
c) Processamento distribuído e maciçamente paralelo.

Atualmente, ainda é uma questão em aberto se apenas a abordagem conexionista ou a abordagem simbólica são poderosas o suficiente como uma base geral para tarefas cognitivas de alto nível como PLN [SMO88, DYE91].

Serão descritos nos próximos capítulos alguns conceitos e técnicas utilizados em processamento de linguagem natural dentro das duas abordagens. Existe ainda uma terceira abordagem, que é a hipótese do sistema conexionista híbrido [WER95] que não será discutido neste trabalho; para maiores detalhes veja [WER95, WAP85, DYE91, LEE91, HEN89].

1.1 Objetivos

O objetivo principal deste trabalho, como já foi dito, é a resolução de anáforas em um pequeno texto composto de sentenças justapostas, mas mesmo assim pode-se destacar, além deste, outros objetivos:

a) Verificar a adeqüabilidade de RNA na resolução de anáforas pronominais;
b) Implementar algumas técnicas de processamento de linguagem natural conexionista;
c) Estudar o treinamento simultâneo de duas redes neurais;

d) Dar à própria rede o controle da execução do modelo;
e) Fazer a rede aprender a regra da resolução do problema proposto.

1.2 Estrutura do trabalho

O presente texto é dividido em 6 capítulos. Nos 3 primeiros capítulos são descritos os principais conceitos e técnicas de Redes Neurais Artificiais, Processamento de Linguagem Natural e Processamento de Linguagem Natural Conexionista.

No capítulo 2 serão vistos os principais conceitos e componentes de uma Rede Neural Artificial, como: atuais paradigmas de redes, topologia, aprendizado, algoritmos de treinamento entre outros tópicos.

No capítulo 3 serão apresentados alguns conceitos relacionados ao processamento de linguagem natural. Neste capítulo as técnicas descritas estão dentro da filosofia da IAS. Neste capítulo serão introduzidos alguns conceitos, como: análise sintática, semântica e pragmática; gramáticas, *parsing*, linguagens formais e roteiros.

No capítulo 4 será discutida a abordagem conexionista para a resolução de problemas em processamento de linguagem natural. Desta abordagem, surgiu o ramo do conexionismo conhecido como Processamento de Linguagem Natural Conexionista. Neste capítulo são discutidos: Problema de tempo em linguagem, Processamento Distribuído Paralelo (PDP) e Representações Conexionistas.

Nos demais capítulos (5 e 6) serão descritas as aplicações desenvolvidas para o tratamento de referências anafóricas. No capítulo 5 é abordada apenas uma rede para analisar segmentos de texto compostos de apenas duas sentenças. No capítulo 6 é discutido um modelo composto de duas RNA treinadas e executadas simultaneamente para resolver o mesmo problema.

2. Redes Neurais Artificiais

2.1 Introdução

Uma Rede Neural Artificial (RNA)[1] é um modelo matemático de processamento de informações inspirado no sistema nervoso biológico, local onde são processadas as informações capturadas do ambiente pelos órgãos sensoriais. Uma RNA é composta de um grande número de elementos de processamento (neurônios), altamente conectados, trabalhando em conjunto para a resolução de um problema específico. As RNAs se utilizam de um processo de aprendizado, e são projetadas para aplicações específicas. O aprendizado em sistemas biológicos (também em RNAs) envolve o ajustamento das conexões sinápticas que existem entre os neurônios.

RNAs estão sendo aplicadas para um número cada vez maior de problemas reais de considerável dificuldade. Uma de suas vantagens mais importantes está na resolução de problemas que são complexos para as tecnologias convencionais – problemas que não têm uma solução algorítmica ou para os quais uma solução algorítmica é muito difícil de ser encontrada. Em geral, por causa de sua inspiração no cérebro humano, RNAs são aplicadas na resolução de problemas que as pessoas são eficientes em resolver, mas os computadores não são.

Em princípio, RNAs podem representar qualquer função computável, isto é, elas podem fazer tudo que um computador digital normal pode fazer. Especialmente algo que pode ser representado como um mapeamento entre espaços vetoriais poderá ser aproximado com precisão arbitrária por uma RNA [RHW86].

As características que tornam a metodologia de redes neurais interessante, do ponto de vista da solução de problemas, são as seguintes [BIT96]:

a) Capacidade de "aprender" através de exemplos e de generalizar este aprendizado de maneira a reconhecer instâncias similares que nunca haviam sido apresentadas como exemplo;
b) Bom desempenho em tarefas mal definidas, em que falta o conhecimento explícito sobre como encontrar uma solução;
c) Não conhecimento a respeito de eventuais modelos matemáticos do domínio de aplicação;
d) Elevada imunidade ao ruído, isto é, o desempenho de uma rede neural não entra em colapso em presença de informações falsas ou ausentes.

Os domínios de aplicações da técnica de redes neurais artificiais mais comuns são:

a) Sistemas Especialistas;
b) Reconhecimento de padrões (análise de cenas, reconhecimento de voz, etc.);
c) Processamento de sinais;
d) Previsões (variações de cargas elétricas, cotações de bolsa de valores, etc.);
e) Diagnóstico de falhas;
f) Controle de processos;

[1] Outros nomes podem ser encontrados na literatura, como: redes conexionistas, processamento distribuído paralelo (PDP), redes adaptativas e computação coletiva.

g) Linguagem Natural.

2.2 Motivação Biológica

Os computadores convencionais (computadores de Von-Neuman) e o cérebro humano resolvem tarefas de maneiras diferentes. Os computadores superam o cérebro humano ao realizarem cálculos matemáticos precisos. O cérebro humano, por sua vez, tem habilidades para reconhecer padrões (visuais, por exemplo) e coerência em associá-los, que excedem em muito a capacidade dos computadores. Então seria interessante encontrar uma maneira de simular essas vantagens do cérebro num computador. Isto permitiria aos computadores resolverem tarefas que eles não efetuam eficientemente. Estas tarefas são resolvidas pelo cérebro graças a sua representação distribuída e processamento maciçamente paralelo. Estima-se que mais de 100 bilhões de neurônios biológicos são conectados entre si de maneira altamente complexa de modo que grande parte do cérebro possa contribuir para o processamento de um sinal [SJT88]. Ainda que a funcionalidade do cérebro seja pouco conhecida, um modelo simples foi desenvolvido. A idéia é tentar simular as habilidades de aprendizado e generalização do cérebro desenvolvendo uma rede neural artificial.

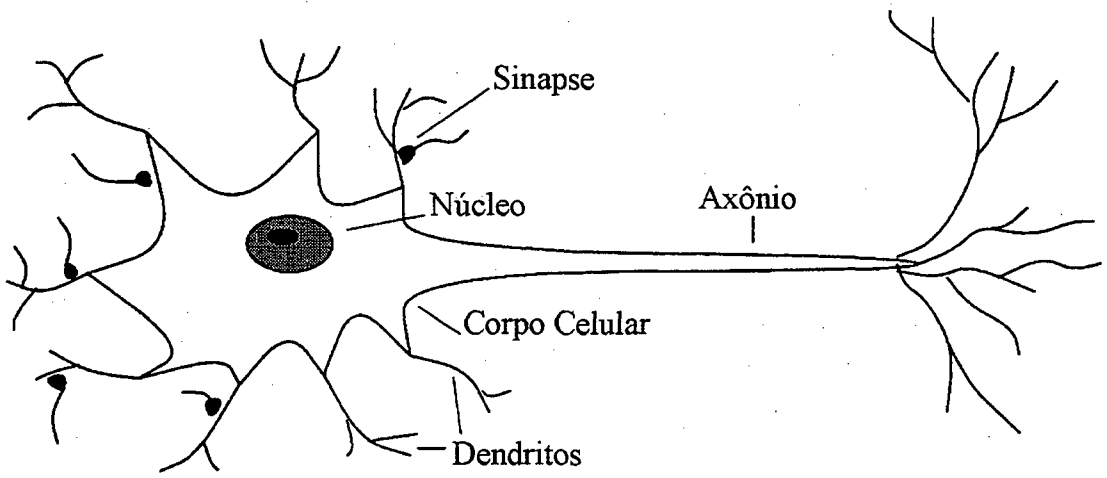

Figura 2-1 : Desenho esquemático do neurônio biológico.

Uma rede neural biológica é tradicionalmente uma rede de neurônios biológicos que podem ser encontrados em cérebros humanos e animais. A Figura 2-1 mostra um desenho esquemático de um neurônio biológico. De maneira extremamente simplificada, um neurônio biológico é formado por um corpo celular ou soma que contém o núcleo da célula, diversos dendritos através dos quais a célula recebe impulsos elétricos reagindo inibitoriamente[2] ou excitatoriamente[3], e um axônio, através do qual impulsos elétricos são enviados a outras células. A propagação de um impulso elétrico ao longo de um dendrito ou de um axônio se dá através da alteração da concentração de íons em ambos os lados da membrana [KSV96]. As interligações

[2] Diminuindo a força do estímulo.
[3] Ampliando a força do estímulo.

entre neurônios são efetuadas através de sinapses, que são pontos de contato[4] entre dendritos e axônios, controlados por impulsos elétricos e por reações químicas devido aos neurotransmissores.

2.3 Modelos Computacionais de Neurônios

Para construir RNAs um modelo de neurônio tem que ser estabelecido. O modelo de neurônio artificial é inspirado no neurônio biológico. Cabe ressaltar que o modelo artificial, atualmente, está muito distante do neurônio biológico.

2.3.1 O Modelo de McCulloch e Pitts

A estrutura do neurônio artificial proposta por McCulloch e Pitts [McP43] é baseada no neurônio biológico. O neurônio de McCulloch e Pitts é de natureza binária, ou seja, suas entradas e a saída são 0s ou 1s. O modelo pode ser descrito intuitivamente da seguinte maneira: se a soma ponderada dos sinais de entrada de um neurônio ultrapassar um determinado limite de disparo, então a saída será 1, senão será 0. As entradas do neurônio também são binárias.

Formalmente, este funcionamento pode ser descrito da seguinte maneira: Considere o i-ésimo neurônio de uma rede neural com n neurônios, este neurônio é caracterizado pelo valor y_i que é a saída do neurônio (chamado ativação do neurônio que corresponde à taxa média dos disparos dos potenciais de ação do neurônio biológico) e pelo valor o_i chamado nível de ativação do neurônio (que corresponde ao potencial de ação do neurônio biológico). No modelo de McCulloch e Pitts, o nível de ativação é definido da seguinte maneira:

$$o_i = \sum_{j=1}^{m} w_{ij} y_j \qquad (i = 1..n) \tag{2-1}$$

onde $w_{ij} \in \Re$ é o peso atribuído à entrada do neurônio i cuja origem é a saída do neurônio j (y_j). Este peso simula a sinapse entre dois neurônios. A saída do neurônio i é dada por $y_i = f(o_i)$. A função f, chamada de função de ativação ou de transferência adotada neste modelo é a função degrau:

$$f(o_i) = \begin{cases} 0 & \text{se } o_i \leq \alpha \\ 1 & \text{se } o_i > \alpha \end{cases} \tag{2-2}$$

onde α é o limite de disparo.

Mesmo com este modelo, McCulloch e Pitts foram capazes de provar que uma rede neural é equivalente a uma máquina de Turing, logo capaz de calcular qualquer função computável [McP43].

---

[4] Na verdade, os dendritos e axônios não têm um contato físico. Quando o potencial de ação atinge a terminação axônica, os neurotransmissores são liberados através da membrana pré-sináptica e estes então iniciam a despolarização da membrana pós-sináptica.

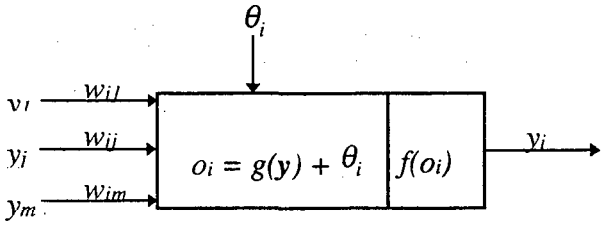

Figura 2-2 : Desenho esquemático do modelo geral de neurônio.

2.3.2 O Modelo Geral de Neurônio

O modelo de McCulloch e Pitts pode ser generalizado. Assim obtemos o modelo geral de neurônio[5] [BAR96a] com os seguintes aspectos (veja Figura 2-2):

a) A função de transferência *f*, que determina a saída de um neurônio, é generalizada e passa a ser uma função limitada qualquer (veja Figura 2-3). Essa função deve ser não linear, diferenciável e monotônica.

O valor de saída da função é usado como um estímulo de entrada para outros neurônios artificiais numa rede de neurônios ou é a saída da rede, se estes forem os neurônios de saída. A função de transferência para unidades ocultas (veja seção 2.4) é necessária para introduzir não linearidade à rede. Sem não linearidade, as unidades ocultas não tornam uma rede neural mais eficiente que o Perceptron[6] [ROS58, ROS62]. A não linearidade é essencial, porque uma rede composta de unidades lineares sempre pode ser reduzida a uma rede equivalente de uma camada que efetua a mesma transformação entrada/saída [WAS89]. Entretanto, é a não linearidade que faz das redes neurais multicamadas métodos eficientes para resolverem problemas complexos.

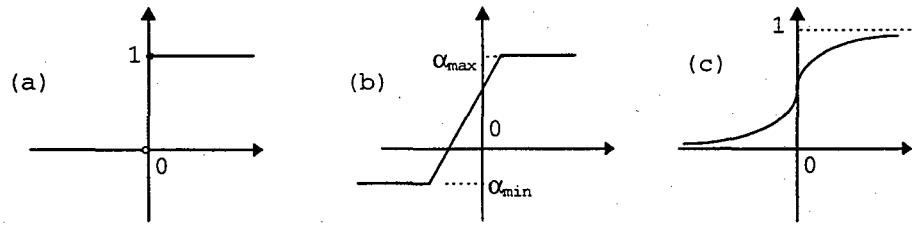

Figura 2-3 : Gráfico de algumas funções de transferência mais utilizadas em neurônios artificiais; a)função degrau; b)função semilinear; c)função logística ou sigmóide.

b) O nível de ativação passa a ser definido como uma função qualquer *g* das entradas do neurônio:

$$o_i = g(x_1, x_2, \ldots, x_m).$$

c) É introduzido um valor de polarização $\theta \in \Re$, de modo que a saída do neurônio passa a ser calculada por $y_i = f(o_i + \theta_i)$; valor abaixo do qual a saída é nula.

---

[5] Neste trabalho será descrito apenas o modelo geral de neurônio estático.

[6] Foi o primeiro modelo de redes neurais. Proposto por Rosenblatt em 1958. Consiste de uma rede de duas camadas formadas por neurônios binários.

Observações:

a) Na maioria dos modelos, a função g é a soma ponderada, como no modelo de McCuloch e Pitts, embora existam modelos onde é utilizado o produto, o mínimo, ou o máximo dos valores da entrada.

b) As funções f mais utilizadas, além da função degrau (Figura 2-3a), são:

-Função semilinear (Figura 2-3b):

$$f(x)=\begin{cases}0 & \text{se } x<\alpha_{min}\\ mx+1 & \text{se } \alpha_{min}<x<\alpha_{max}\\ f_{max} & \text{se } x>\alpha_{max}\end{cases} \tag{2-3}$$

-Função sigmóide (Figura 2-3c):

$$f(x)=\frac{1}{1+e^{-x}} \tag{2-4}$$

-Função tangente hiperbólica:

$$f(x)=\frac{1-e^{-x}}{1+e^{-x}} \tag{2-5}$$

c) A partir deste ponto o neurônio artificial será referido apenas como unidade de processamento ou, simplesmente, unidade; e a função g será a soma ponderada das entradas. O valor de polarização θ_i será considerado como uma entrada adicional y_0 com valor sempre igual a 1 e terá um peso correspondente w_{i0} (veja Figura 2-4).

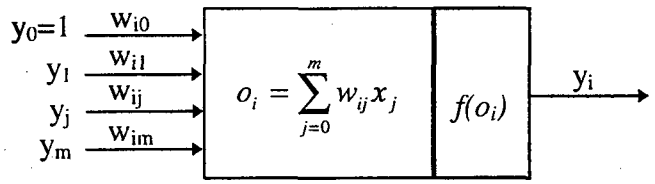

Figura 2-4 : Unidade de processamento (neurônio artificial) com m valores de entrada e com $\theta_i = y_0 = 1$ com $i = 1..n$.

2.4 Topologia de Redes Neurais Artificiais

Uma vez definido o neurônio é possível estudar as propriedades de redes de neurônios interconectados. RNAs podem ser vistas como um grafo orientado e rotulado, no qual as unidades de processamento são nós e os arcos orientados (com pesos) são conexões [JMM96].

Uma rede neural artificial pode conter três tipos de unidades de processamento:

a) De entrada : São todas as unidades que recebem um estímulo do meio externo (neurônios sensoriais dos seres vivos).
b) De saída : São as unidades que podem modificar, de alguma forma, o mundo exterior (neurônios motores).
c) Ocultas : São as unidades que não são nem de entrada e nem de saída, são internas à rede.

As unidades ocultas são importantes do ponto de vista matemático, pois é possível provar que a classe de problemas conhecidos como não linearmente separáveis só possuem solução através de redes que possuem, no mínimo, uma camada oculta [MIP69].

A camada que recebe a entrada é chamada de camada de entrada e não efetua qualquer operação além da distribuição do sinal de entrada para a próxima camada. A saída da rede é gerada pela camada de saída. Quaisquer outras camadas são chamadas camadas ocultas, porque elas são internas à rede e não têm contato direto com o ambiente externo. O número de camadas ocultas é determinado antes do treinamento[7], podendo ter zero ou mais camadas ocultas. A Figura 2-5 ilustra estas definições.

Uma topologia bem adequada é altamente importante para a rede encontrar uma solução para um problema. Redes que são muito pequenas não são capazes de aprender o conjunto de treinamento ou o aprendem parcialmente. Redes que são muito grandes, por outro lado, podem não ter uma boa generalização após terem sido treinadas. A topologia ótima de uma rede é a que tem menor complexidade e que permite à rede aprender o conjunto de treinamento e fazer uma boa generalização com o conjunto de teste. Tal topologia, no entanto, pode ser difícil de se encontrar. Não existe um procedimento que determina qual será a topologia ótima para resolver um determinado problema, o que se utiliza na prática é testar diferentes topologias até encontrar uma que resolva o problema satisfatoriamente.

Posto que RNAs podem ser vistas como um grafo e que possuem tipos de unidades diferentes, pode-se identificar algumas topologias de redes dependendo de como o fluxo das informações é conduzido através da rede [HAY94].

a) Rede direta

Redes diretas (também conhecidas como redes *feedforward*) são redes nas quais o grafo não possui laços. Assim, o fluxo dos dados das unidades de entrada para as unidades de saída é em um único sentido (veja Figura 2-5). As redes são organizadas em camadas: Camada de entrada (formadas pelas unidades de entrada), Camadas ocultas (formadas pelas unidades ocultas) e Camada de Saída (formada pelas unidades de saída). Estas redes são as mais utilizadas atualmente.

[7] Se for utilizado o aprendizado não construtivo (seção 2.5 Treinamento de RNAs).

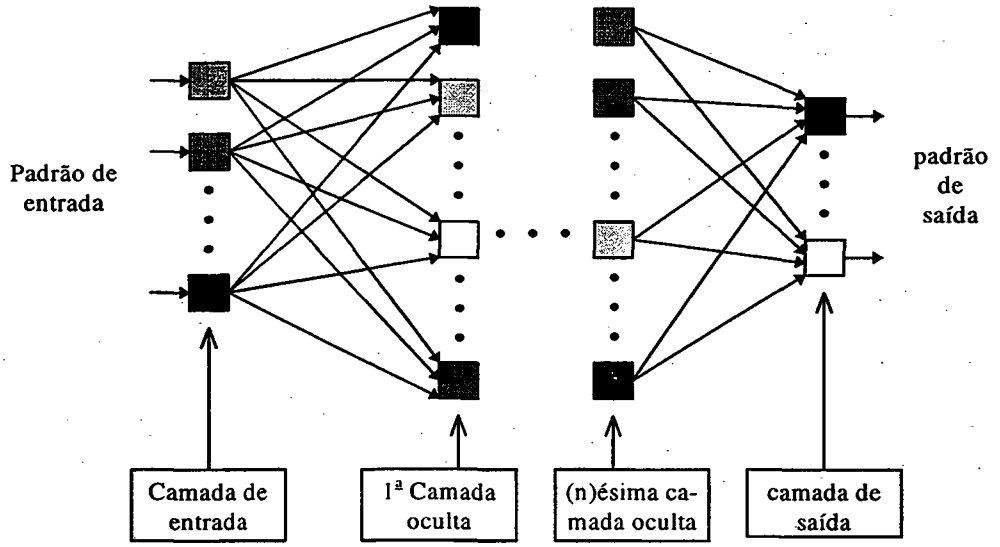

Figura 2-5 : Rede neural direta inteiramente conectada com n camadas ocultas. Os tons de cinza representam os valores de ativação em um intervalo limitado: valor mínimo (branco) a um valor máximo (preto).

b) Rede com Ciclos

Se a saída de um neurônio de uma rede neural é uma entrada para ele mesmo ou para algum neurônio da camada precedente, esta rede é chamada de rede com ciclos (fluxo dos dados nas duas direções) ou rede recorrente, veja Figura 2-6.

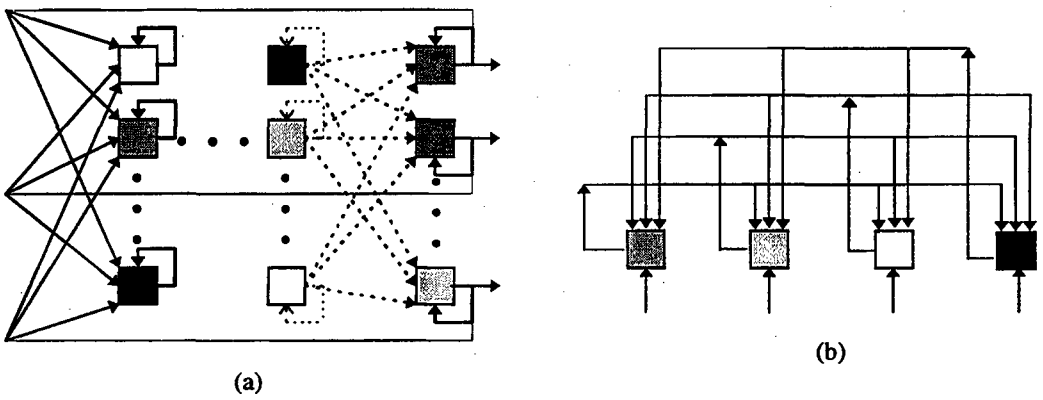

Figura 2-6 : Redes Neurais com ciclos; a) recorrente multicamada; b)Rede de Hopfield.

2.5 Treinamento de Redes Neurais Artificiais

A habilidade de aprender é fundamental à inteligência. O processo de aprendizado no contexto de RNAs pode ser visto como o problema de atualização dos pesos das conexões de modo que uma rede possa efetuar eficientemente uma tarefa específica. A rede deve ajustar os pesos das conexões de acordo com os padrões de treinamento disponíveis. O desempenho é

melhorado progressivamente, ajustando-se os pesos na rede iterativamente. A habilidade das RNAs de aprender automaticamente através de exemplos, ao invés de seguir um conjunto de regras específico, torna-as atrativas.

O treinamento de uma RNA é realizado de modo que a aplicação de um conjunto de entrada produza um conjunto de saída (saída da rede). Cada entrada ou saída (conjunto de valores) é referenciado como um vetor. O treinamento é realizado aplicando os vetores de entrada seqüencialmente, enquanto os pesos da rede são ajustados de acordo com um procedimento pré-determinado. Durante o treinamento, os pesos da rede gradualmente convergem para valores de tal modo que o vetor de entrada produza um vetor de saída cada vez mais próximo do vetor desejado (saída alvo).

De todas as características de RNAs, talvez a sua habilidade de aprender seja a mais interessante. A equação que especifica esta mudança é chamada de lei de aprendizado, ou regra de aprendizado.

2.5.1 Paradigmas de Aprendizado

Pode-se categorizar o aprendizado em 2 tipos distintos: Aprendizado supervisionado e aprendizado não supervisionado.

a) Aprendizado supervisionado

Algoritmos de treinamento supervisionado requerem que sejam apresentados à rede o vetor de entrada e o seu correspondente vetor de saída desejado. Estes vetores são chamados pares de treinamento. Uma rede é treinada com um número finito de pares de treinamento. Um vetor de entrada é aplicado à rede, a saída da rede é calculada e comparada à saída desejada, a diferença (erro) é enviada de volta à rede e os pesos são alterados de acordo com um algoritmo que procura minimizar o erro. Os vetores do conjunto de treinamento são aplicados seqüencialmente, calcula-se o erro e os pesos são ajustados para cada vetor, até o erro para o conjunto de treinamento inteiro atingir um valor mínimo aceitável. A rede resultante deverá ser capaz de generalizar (dar uma boa resposta) quando forem apresentados casos não encontrados no conjunto de treinamento. Na seção 2.5.3.1 é descrito um algoritmo de treinamento supervisionado.

Distinguem-se duas classes de algoritmos de aprendizado supervisionado: não construtivo e construtivo. Por não construtivos, entendem-se os algoritmos para os quais a topologia da rede tem que ser determinada antes do treinamento, enquanto no aprendizado construtivo o próprio algoritmo automaticamente determina a topologia da rede.

b) Aprendizado não supervisionado ou auto-organização

Apesar de algum sucesso em aplicações, o aprendizado supervisionado vem sendo dito como não biologicamente plausível. É difícil de imaginar um mecanismo de treinamento no cérebro que compara a saída real com a saída desejada e volta fazendo correções na rede. Se este é o mecanismo utilizado pelo cérebro, de onde vem o padrão de saída desejado? Aprendizado não supervisionado é um modelo plausível de aprendizado no sistema biológico. Desenvolvido por Kohonen [KOH88] e muitos outros, este tipo de aprendizado não necessita da saída desejada,

e assim, não faz comparações a uma resposta ideal pré-determinada. O conjunto de treinamento consiste apenas de vetores de entrada. O algoritmo de treinamento modifica os pesos da rede para produzir vetores de saída que sejam consistentes, isto é, aplicações de vetores suficientemente similares produzirão o mesmo padrão de saída. O processo de treinamento, portanto, extrai as propriedades estatísticas do conjunto de treinamento e agrupa os vetores similares em classes. Na seção 2.5.3.2 é descrito um algoritmo de treinamento não supervisionado.

2.5.2 Regras de Modificação dos Pesos das Conexões

Os paradigmas de aprendizado discutidos acima resultam em um ajustamento dos pesos das conexões entre unidades de acordo com alguma regra de modificação. Virtualmente todas as regras de aprendizado para modelos de RNAs podem ser consideradas como variantes da primeira regra de aprendizado de Hebb [HEB49]. A idéia básica é que, se duas unidades i e j são ativadas simultaneamente, sua conexão deve ser reforçada. Se a unidade i recebe entrada de j, a primeira regra de Hebb para modificar o peso w_{ij} entre i e j é:

$$\Delta w_{ij} = \eta y_i y_j \tag{2-6}$$

onde η é uma constante de proporcionalidade representando a taxa de aprendizado e y_i e y_j são as saídas de i e j respectivamente. Outra regra comum não usa a saída obtida da unidade i, mas a diferença entre esta e a saída desejada para ajustar os pesos das conexões entre unidades:

$$\Delta w_{ij} = \eta\left(d_i - y_i\right) y_j \tag{2-7}$$

onde d_i é a saída desejada fornecida por um professor. Esta regra é freqüentemente referida como regra de Widrow-Hoff ou regra delta, utilizada no treinamento das redes Adaline e Madaline [WIH60].

2.5.3 Algoritmos de Treinamento de Redes Neurais Artificiais

Nesta seção serão descritos dois algoritmos para o treinamento de RNA: algoritmo de retropropagação, que é um algoritmo de treinamento supervisionado para treinar redes diretas (ou recorrentes - seção 2.4), e a lei de aprendizado de Kohonen, que é um método de treinamento não supervisionado, utilizado para treinar os chamados mapas auto-organizativos.

2.5.3.1 Algoritmo de Retropropagação

Minsky e Papert [MIP69] mostraram, em 1969, que redes com camadas ocultas podem superar muitas restrições[8], mas não apresentaram uma solução para o problema de como ajustar os pesos de uma rede que possui camadas ocultas. Rumelhart, Hinton e Williams [RHW86] apresentaram em 1986 um algoritmo que realiza tal tarefa[9].

---

[8] Como resolver problemas linearmente não separáveis.

[9] O algoritmo de retropropagação foi desenvolvido anteriormente por Werbos [WER74] em sua tese de doutoramento em 1974 e por Parker [PAR85] em 1985 independentemente.

Retropropagação é um método de treinamento supervisionado, por isso antes de iniciar o treinamento deve-se gerar o conjunto de pares de treinamento que o sistema deverá aprender. O treinamento consiste em apresentar os pares de treinamento repetidamente à rede e gradualmente mudar os pesos de modo que a rede eventualmente computará o mapeamento desejado.

A cada apresentação de um exemplo, o padrão de entrada é posicionado na camada de entrada e é propagado através da rede até a camada de saída. O padrão de saída da rede é comparado ao padrão de saída desejado, e um sinal de erro é formado para cada unidade de saída. Então o sinal de erro é propagado de volta através da rede até a primeira camada oculta.

Os sinais de erro são calculados para todos os pares de treinamento (uma época) e então usados para determinar o valor para o ajustamento de cada peso, de modo que o erro de saída total se reduzirá mais rápido. Em outras palavras, é calculado o gradiente do erro em relação aos pesos atuais. Um pequeno passo é feito nesta direção (os pesos são alterados ao longo do gradiente), e o gradiente é calculado novamente com os novos pesos. O processo eventualmente encontra um conjunto de pesos de maneira que a rede produza padrões de saídas corretos para cada padrão de entrada no conjunto de treinamento.

Outro conjunto de padrões, chamado de conjunto de teste, é usado para testar se a rede treinada está fazendo uma boa generalização. O conjunto de teste não pode ser um subconjunto do conjunto de treinamento, mas deve estar dentro do mesmo espaço do conjunto de treinamento. O bom desempenho do conjunto de teste indica que a rede aprendeu a tarefa de mapeamento em geral e não apenas memorizou o conjunto de treinamento.

Formalmente, um padrão de entrada do conjunto de treinamento é selecionado aleatoriamente, e cada elemento deste padrão é atribuído às saídas das unidades de entrada. Cada unidade nas camadas subsequentes calcula a soma dos seus valores de entrada, ponderada pelos pesos das conexões correspondentes (equação 2-1), e produzem um valor de saída (Figura 2-7):

$$y_i^{\,l} = f\left(o_i^{\,l}\right) = f\left(\sum_{j=0}^{m} w_{ij}^{\,(l-1)} y_j^{\,(l-1)}\right) \tag{2-8}$$

onde $y_i^{\,l}$ é a saída da unidade i na camada l, $o_i^{\,l}$ representa a soma ponderada de suas entradas, e $w_{ij}^{\,l}$ é o peso entre a unidade j na camada $(l\text{-}1)$ e a unidade i na camada l. A função sigmóide (equação 2-4) é uma escolha conveniente para a função de transferência f, porque os valores de saída estão entre 0 e 1 e tem uma derivada simples:

$$f'(o) = f(o)\left(1 - f(o)\right) \tag{2-9}$$

Uma vez que o padrão de ativação é produzido na camada de saída (camada L), ele é comparado ao padrão de saída desejado. Um sinal de erro é formado para cada unidade de saída da rede (camada L):

$$\delta_k^{\,L} = f'\left(o_k^{\,L}\right)\left(d_k - y_k^{\,L}\right) \tag{2-10}$$

onde d_k é o valor desejado para a unidade de saída k. Quando a função de transferência é a sigmóide, a equação 2-10 se reduz a:

$$\delta_k{}^L = y_k{}^L\left(1 - y_k{}^L\right)\left(d_k - y_k{}^L\right) \tag{2-11}$$

Na fase de volta (*backward*), o sinal de erro para a unidade i na camada prévia l é calculado propagando-se o erro de volta através da rede:

$$\delta_i{}^l = f'\left(o_i{}^l\right)\sum_k \delta_k{}^{(l+1)} w_{ki}{}^l \tag{2-12}$$

Propagação da entrada para a saída:

$$y_i{}^l = f\left(\sum_{j=1}^{m^{l-1}} w_{ij}{}^{(l-1)} y_j{}^{(l-1)}\right) + w_{i0}{}^{(l-1)}$$
$$= f\left(o_i{}^l\right)$$

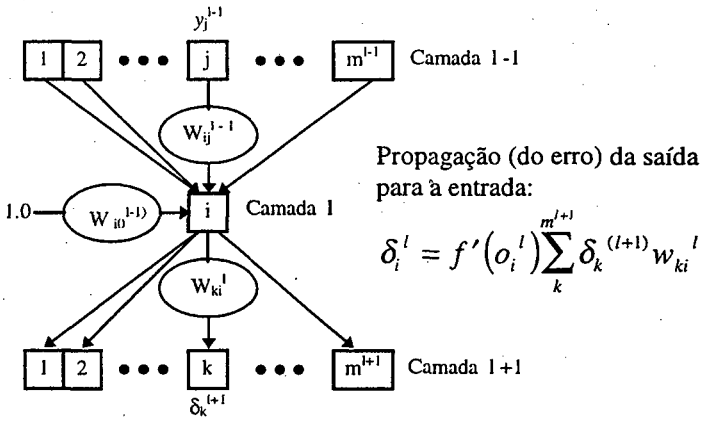

Figura 2-7 : Desenho da unidade genérica *i* de uma RNA, mostrando a propagação nos dois sentidos na rede durante o treinamento com o algoritmo de retropropagação.

ou no caso da função sigmóide,

$$\delta_i{}^l = y_i{}^l\left(1 - y_i{}^l\right)\sum_k \delta_k{}^{(l+1)} w_{ki}{}^l \tag{2-13}$$

O aprendizado consiste em mudar cada peso das conexões proporcionalmente ao sinal do erro:

$$\Delta w_{ki}{}^l = \eta \delta_k{}^{(l+1)} y_i{}^l \tag{2-14}$$

Então atualiza-se cada peso pela equação:

$$w_{ki}{}^l(t) = w_{ki}{}^l(t-1) + \Delta w_{ki}{}^l \tag{2-15}$$

onde t é o índice de apresentação (ou época).

As equações 2-11 e 2-13 fornecem um procedimento recursivo para computar os δs para todas as unidades na rede, que são então usadas para calcular as mudanças de pesos de acordo com a equação 2-14. Este procedimento constitui a regra delta generalizada [RHW86] para uma rede direta de unidades não lineares.

Se o erro para um par p de entrada/saída é definido como:

$$E^p = \frac{1}{2}\sum_k \left(d_k - y_k{}^L\right)^2 \tag{2-16}$$

Pode ser mostrado [RHW86] que $\Delta w_{ki}{}^l$ indica a direção da descida mais íngreme na superfície do erro total para este padrão:

$$-\frac{dE_p}{dw_{ki}{}^l} = \delta_i{}^{(l+1)} y_i{}^l .$$

Entretanto, se $\Delta w_{ki}{}^l$ é a média sobre o conjunto de treinamento, estes valores indicam a direção mais íngreme na superfície do erro total E para todos os padrões:

$$E = \frac{1}{2}\sum_p \sum_k \left(d_k - y_k{}^L\right)^2 \tag{2-17}$$

• **Taxa de aprendizado e *Momentum***: O procedimento de aprendizado requer que a mudança nos pesos seja proporcional a *dE/dw*. Verdadeiros métodos de gradiente descendente requerem que sejam feitos passos infinitesimais. A constante de proporcionalidade é a taxa de aprendizado η. Para propósitos práticos escolhe-se η tão grande quanto possível sem obter oscilações. Uma maneira de evitar oscilação com η grande, é fazer a alteração dos pesos dependente da alteração passada adicionando um termo chamado *momentum*:

$$\Delta w_{ki}{}^l(t) = \eta \delta_k{}^{(l+1)} y_i{}^l + \alpha \Delta w_{ki}{}^l(t-1) \tag{2-18}$$

onde t é o índice do número de apresentação de cada padrão e α é uma constante que determina o efeito da alteração dos pesos anteriores.

O papel do termo *momentum* é mostrado na Figura 2-8. Quando o termo *momentum* não é usado e a taxa de aprendizado é baixa, o algoritmo leva um longo tempo para convergir, com taxa de aprendizado alta, podem ocorrer oscilações que retardam ou impedem a convergência do algoritmo. Quando o termo *momentum* é usado, a convergência é mais rápida.

Apesar do grande sucesso do algoritmo de retropropagação, existem alguns problemas que podem surgir no treinamento. Um deles é o longo tempo no processo de treinamento que pode ser o resultado de escolher uma taxa de aprendizado e *momentum* não ótimos.

• **Paralisia da rede**. Enquanto a rede treina, os pesos podem ser ajustados para valores muito grandes. Assim a entrada para a função de transferência (soma ponderada) pode possuir valores muito altos (negativos ou positivos). Se a função de transferência for a função sigmóide, a saída da unidade será muito próxima de zero ou um. Como fica claro pelas equações 2-11 e 2-13, os ajustamentos dos pesos que são proporcionais a *y(1 - y)* serão próximos de zero, assim o processo de treinamento pode parar.

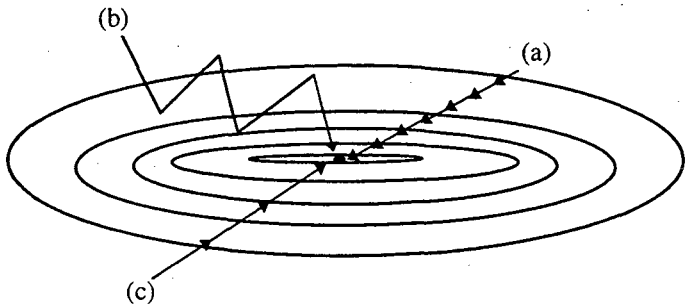

Figura 2-8 : A descida no espaço de pesos. a) para pequenas taxas de aprendizado; b)para grandes taxas de aprendizado: note as oscilações; c) com grande taxa de aprendizado, mas com termo *momentum* adicionado.

• **Mínimos locais**. Retropropagação aproxima gradiente descendente, que em princípio não é um método de otimização muito bom. Se existem mínimos locais na superfície de erro, o método corre o risco de ficar preso neste mínimo. Mínimos locais não são problemáticos na prática, por razões que não são muito bem conhecidas. Existem algumas técnicas para escapar do problema de mínimos locais: iniciar o treinamento com novos pesos iniciais é talvez o mais simples e funciona razoavelmente bem [RHW86]; iniciar com uma taxa de aprendizado maior e diminuir gradualmente durante o treinamento. Ambas funcionam de maneira similar.

2.5.3.2 Mapas Auto-Organizativos

Na seção anterior discutiu-se um algoritmo que realiza treinamento supervisionado, ou seja, o algoritmo necessita do par entrada e saída desejada *(x,d)*. Entretanto, existem problemas quando os pares entrada/saída desejada não são disponíveis. Assim , existe somente o conjunto de padrões de entrada. O treinamento é feito sem a presença de um professor externo.

Há muitos tipos de redes auto-organizativas, um dos procedimentos mais básico é o aprendizado competitivo proposto por Rumelhart e Zipser [RUZ85]. Outros exemplos são: ART, proposto por Carpenter e Grossberg [CAG87, GRO76] e o *Cognitron* de Fukushima [FUK75, FUK88]. Discutir-se-á aqui os chamados Mapas auto-organizativos [10] propostos por Teuvo Kohonen [KOH90, KOH88].

Mapas auto-organizativos, ou sistemas constituídos de diversos módulos de mapas, têm sido usados para tarefas similares àquelas que as outras redes mais tradicionais estão sendo aplicadas: reconhecimento de padrões, robótica, controle de processos e até processamento de informações semânticas. A segregação espacial de diferentes ativações e suas organizações em subconjuntos relacionados topologicamente resulta em um alto grau de eficiência em operações de redes neurais.

Neste tipo de arquitetura de RNA (Figura 2-9), células vizinhas em uma RNA competem através de seus valores de ativação, por meio de interações laterais, e desenvolvem detetores específicos de diferentes padrões de sinal; nesta categoria de RNA, o aprendizado é chamado competitivo, ou não supervisionado, ou auto-organizável. Os algoritmos de adaptação de pesos

[10] Outros nomes são: Mapa de características de Kohonen ou simplesmente Mapas de Kohonen.

no aprendizado não supervisionados são geralmente baseados em alguma forma de competição global entre as unidades.

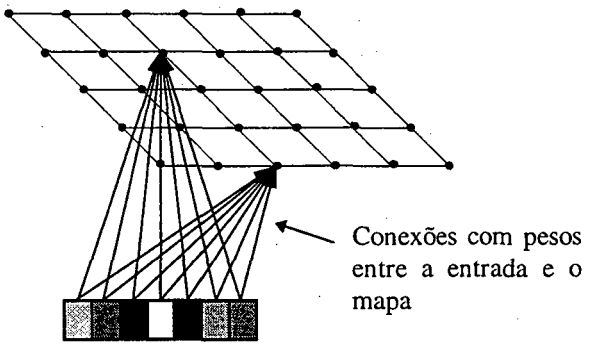

Figura 2-9 : Mapa de Kohonen de 2 dimensões. Todos os valores de entrada são conectados a todas as células, a figura mostra apenas conexões para duas células.

Mapas auto-organizativos são redes neurais artificiais com uma camada de entrada e um plano bidimensional (ou n-dimensional) com as unidades dispostas topologicamente (Figura 2-9). As células são especificamente sintonizadas para vários padrões de sinais de entrada ou classe de padrões, através de aprendizado não supervisionado. Na versão básica, somente uma célula ou grupo de células respondem à entrada corrente. A localização da resposta tende a se tornar ordenada como se algum sistema de coordenada significativo para diferentes características de entrada, estivesse sendo criado na rede. A localização espacial ou coordenada de uma célula na rede corresponde então a um domínio particular de padrões de sinais de entrada. Cada célula ou grupo local de células atuam como um codificador separado para a mesma entrada.

Considere a rede bidimensional da Figura 2-10. O arranjo das unidades pode ser hexagonal, retangular (Figura 2-11), etc. Considere também $x=(x_1, x_2, ...,x_n)^T \in \Re^n$ como sendo o vetor de entrada, que é conectado a todas as unidades da rede. O vetor peso para a unidade i será denotado por $w_i=(w_{i1}, w_{i2}, ..., w_{in})^T \in \Re^n$.

O primeiro passo será calcular a unidade "vencedora". Utiliza-se, então a distância euclidiana[11] para calcular a distância entre os pesos e a entrada. A unidade que possuir a menor distância será a unidade "vencedora", denotada por c:

$$\|x - w_c\| = \min_i \{\|x - w_i\|\}$$

Uma vez definida a unidade vencedora, deve-se atualizar os pesos (w_c) desta célula e os pesos das células vizinhas. Para isso devemos definir um conjunto de vizinhança N_c em volta da unidade c. Em cada passo de treinamento os pesos das células dentro do conjunto N_c serão atualizados e os demais pesos ficarão intactos.

[11] Com esta medida mede-se a similaridade entre os vetores $\mathbf{x}$ e $\mathbf{w_i}$. Pode-se utilizar o produto interno entre $\mathbf{x}$ e $\mathbf{w_i}$ como outra medida da similaridade entre estes vetores.

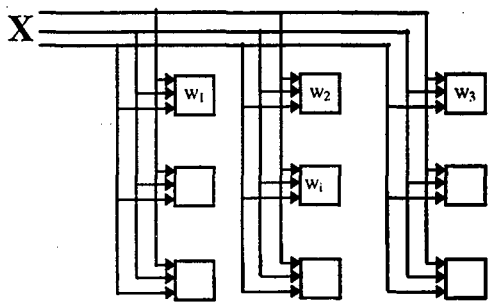

Figura 2-10 : Mapa auto-organizativo com 3 entradas e 9 unidades. O vetor $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ representa a entrada para a rede. Os vetores pesos estão representados como $\mathbf{w}_i$ que corresponde ao i-ésimo vetor peso para a i-ésima unidade.

O raio de N_c pode ser variável. Para uma boa ordenação global tem sido vantajoso experimentalmente deixar N_c grande no início e diminuir monotonicamente (Figura 2-11), podendo chegar até $N_c = \{c\}$ que, neste caso, atualiza somente os pesos da unidade vencedora [KOH90].

O processo de atualização então será:

$$w_i(t+1) = \begin{cases} w_i(t) + \eta(t)[x(t) - w_i(t)] & \text{se } i \in N_c(t), \\ w_i(t) & \text{se } i \notin N_c(t) \end{cases} \qquad (2\text{-}19)$$

onde $\eta(t)$ é a taxa de aprendizado e $0<\eta(t)<1$.

Uma notação alternativa é introduzir uma função escalar $h_{ci}(t)$:

$$w_i(t+1) = w_i(t) + \eta(t) h_{ci}(t)\left[x(t) - w_i(t)\right]$$

onde, $h_{ci}(t)=1$ se a unidade $i \in N_c$ ou $h_{ci}=0$ se $i \notin N_c$. Por outro lado a definição de h_{ci} pode ser mais geral, uma interação lateral biológica tem a forma de uma curva de sino (gaussiana). Denotando d_{ci} como a distância lateral da unidade i à unidade c, temos a função gaussiana para h_{ci}:

$$h_{ci} = \exp\left(-d_{ci}^2 / \sigma^2\right)$$

A taxa de aprendizado $\eta(t)$ e a função de vizinhança $h_{ci}(t)$ são dependentes do tempo. Para uma boa convergência é importante que $\eta(t)$ e $\sigma(t)$ diminuam com o tempo. Uma boa escolha para $\eta(t)$ e $\sigma(t)$, respectivamente, é:

$$\eta(t) = \eta_0 \exp\left(-t/\tau_1\right) \qquad \text{e} \qquad \sigma(t) = \sigma_0 \exp\left(-t/\tau_2\right)$$

As constantes η_0 e σ_0 são os valores de $\eta(t)$ e $\sigma(t)$ no tempo $t=0$ e τ_1 e τ_2 são suas respectivas constantes de tempo.

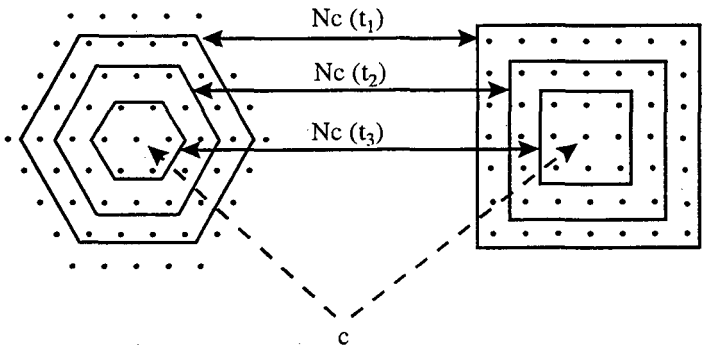

Figura 2-11 : Exemplo de vizinhança topológica $N_c(t)$, onde $t_1<t_2<t_3$.

Utilizando-se o algoritmo descrito acima pode-se encontrar *clusters* de dados primários de uma maneira organizada hierarquicamente. Em um sistema de aprendizado competitivo, um número de células comparam o mesmo sinal de entrada com seus respectivos pesos, e a célula vencedora então "ganha o direito" de atualizar seu peso e/ou os pesos das células vizinhas. Assim diferentes células aprendem diferentes aspectos da entrada.

Os mapas auto-organizativos também são aplicados em Processamento de Linguagem Natural [RIK89, SJC91, HPK95, JAM95].

3. Processamento de Linguagem Natural

3.1 Introdução

Processamento de Linguagem Natural (PLN) é um ramo da Inteligência Artificial que tem por objetivo gerar e/ou interpretar textos[12] em uma língua natural (p. ex. português, inglês, etc.) utilizando-se o computador para realizar tal tarefa.

PLN é uma área multidisciplinar envolvendo estudos nas áreas de ciência da computação, lingüística e ciências cognitivas. A pesquisa em PLN divide-se em duas sub-áreas de trabalho: Interpretação[13] e Geração.

- **Interpretação de Linguagem Natural** baseia-se em mecanismos que tentam compreender frases em alguma língua natural (LN), buscando traduzi-las para uma representação que possa ser compreendida e utilizada pelo computador. Um exemplo clássico de interpretação são as interfaces em LN para consultas em Banco de Dados.

- **Geração de LN** é o oposto da interpretação. O computador traduz uma representação interna para um texto em alguma língua. O texto de saída deve ser o mais próximo possível de um texto produzido por um falante daquela língua.

Um exemplo de aplicação de interpretação e geração de LN é a tradução automática de textos entre duas línguas L1 e L2. Na tradução automática, uma frase na língua natural L1 pode ser traduzida para uma representação interna – interpretação – e em seguida é traduzida para uma frase na outra língua L2 – geração.

3.2 Alguns Conceitos de Lingüística

Uma língua natural é formada por um conjunto infinito de orações (conjunto de frases). As frases da língua podem ser divididas em unidades básicas de sons e significados – as palavras. As palavras, por sua vez, podem ser subdivididas em unidades mínimas de sons (fonemas) e de significados (morfemas).

O morfema básico de uma palavra é chamado de raiz. A uma raiz podem-se afixar outros morfemas , por exemplo, prefixos, sufixos, vogal temática e desinência.

- **Categoria Gramatical**: Também conhecida como classes de palavras. As palavras são classificadas em categorias sintáticas de acordo com sua denotação. Uma mesma palavra pode pertencer a mais de uma categoria sintática, dependendo de sua posição dentro de uma oração. As classes mais conhecidas são: Substantivo, adjetivo, numeral, pronomes, verbos, advérbios, etc.

---

[12] A linguagem falada também faz parte de PLN. Todavia, neste texto será considerada apenas a linguagem escrita.
[13] Também conhecida como compreensão ou entendimento de uma LN.

• **Frase**: É um conjunto de palavras que possui um significado completo e serve para estabelecer uma comunicação; descrever ações, estados ou fenômenos; expressar juízos, emoções, etc.

• **Funções Sintáticas**: Sujeito, objeto, complemento, adjunto, conectivo, etc.

• **Papéis Temáticos**: Papéis semânticos comuns em formas sintáticas alternativas: agente, paciente, instrumento, etc. Veja Gramáticas de casos na seção 3.9.

• **Sintagmas**: Elemento da estrutura da oração. Grupo de palavras classificado de acordo com a categoria sintática de seu elemento núcleo. Os sintagmas nominais (NP) têm como núcleo um substantivo; os sintagmas verbais (VP) possuem um verbo como núcleo; os sintagmas preposicionais (PP) começam com uma preposição, etc. Exemplos:

NP: **Pedro, A moça, A uva seca**;
VP: **chegou tarde, escreveu uma carta, chove**;
PP: **para mim, de João a Carlos**.

3.3 Conhecimento e linguagem

Um programa de compreensão de linguagem deve ter um considerável conhecimento sobre a estrutura da língua, incluindo conhecimento sobre as palavras, como combinar as palavras em sentenças, o que as palavras significam, como os significados destas palavras contribuem para o significado da sentença, e assim por diante. Entretanto, um programa não pode simular o comportamento lingüístico sem primeiro levar em conta importantes aspectos da inteligência humana – seu conhecimento do mundo e suas habilidades de raciocínio. Por exemplo, para responder perguntas ou para participar de uma conversa, uma pessoa não somente deve saber muito sobre a estrutura da língua que está sendo utilizada, mas também deve saber sobre o mundo em geral e do emprego da conversa em particular. Assim, um sistema de linguagem natural necessitaria de métodos de codificação para usar este conhecimento de forma a produzir o comportamento apropriado. Além disso, o conhecimento da situação atual (ou contexto) representa um papel crucial na determinação de como o sistema interpreta uma sentença particular.

As diferentes formas de conhecimento têm sido tradicionalmente definidas como [WEI86]:

a) **Conhecimento fonético e fonológico** preocupam-se em como os sons são relacionados para produzir palavras. Este tipo de conhecimento é importante para sistemas de compreensão automática da fala.

b) **Conhecimento morfológico** diz respeito a como as palavras são construídas sobre seu mais básico significado, o morfema. Por exemplo, pode–se construir a palavra velozmente da raiz veloz acrescentando o sufixo mente.

c) **Conhecimento sintático** trata de como as palavras podem ser colocadas juntas para formar sentenças que parecem corretas na língua. Esta forma de conhecimento identifica como

uma palavra relaciona-se com outras (por exemplo, se uma palavra modifica a outra, ou se elas não estão relacionadas).

d) **Conhecimento semântico** trata do significado de uma palavra e como este significado se combina em sentenças para formar o significado da sentença.

e) **Conhecimento pragmático** diz respeito ao modo como as sentenças são usadas em diferentes contextos e como os contextos afetam a interpretação da sentença.

f) **Conhecimento do mundo** inclui o conhecimento em geral sobre a estrutura do mundo que o usuário da língua deve ter para manter uma conversa, e inclui o que cada usuário da língua deve ter sobre suas convicções e objetivos.

Estas definições são imprecisas, e incompletas. Por exemplo, qualquer fato particular pode incluir aspectos de diversos níveis diferentes. Entretanto, pode-se dividir este problema em três grandes fases de processamento que são baseadas nas técnicas computacionais usadas em cada fase: fase de *parsing* (incluindo tratamento morfológico), fase de interpretação semântica e fase de interpretação contextual.

3.3.1 Sintaxe, Semântica e Pragmática

Os seguintes exemplos podem ajudar a compreender a distinção entre sintaxe, semântica e pragmática. Imagine que os exemplos a seguir são frases para introduzir este capítulo que é sobre PLN (ex. adaptado de [ALL87]).

1. **Este capítulo descreve alguns conceitos básicos que são usados na construção de modelos computacionais de compreensão da linguagem natural.**

Esta sentença parece ser, para este capítulo, um início razoável. É adequada em todos os sentidos: sintático, semântico e pragmático. Entretanto, cada um dos exemplos seguintes violam um ou mais destes níveis. A sentença 2 é bem formada sintaticamente e semanticamente, mas não pragmaticamente:

2. **Rãs verdes têm grandes focinhos.**

Esta sentença não serve como a primeira sentença deste capítulo, simplesmente porque o leitor não encontraria razão para iniciar o capítulo com esta sentença. A sentença 3 é pior:

3. **Idéias verdes têm grandes focinhos.**

Enquanto a sentença 3 é obviamente mal formada pragmaticamente, ela é também semanticamente mal formada. Para ver isto, considere o argumento de que a sentença 2 é ou não verdadeira, mas não se pode fazer isso com a sentença 3. Não se pode afirmar ou negar a sentença 3 em conversações coerentes. Entretanto, a sentença 3 tem alguma estrutura e pode–se discutir o que está errado com ela. Em particular, idéias não podem ser verdes e se pudessem, elas certamente não poderiam ter grandes focinhos. A sentença 4 é totalmente mal formada:

4. **Grandes tem focinhos de idéias verdes.**

De fato, esta sentença não está inteligível; embora ela contenha as mesmas palavras da sentença 3, não tem o suficiente de uma estrutura de modo que se possa dizer o que está errado com ela. Ela é sintaticamente mal formada.

3.4 Etapas de Análise

A análise de linguagem, em geral, é dividida em 3 etapas, as quais serão descritas a seguir.

3.4.1 Análise Sintática

Na análise sintática é feita uma transformação da seqüência de palavras de entrada em estruturas que mostram como as palavras se relacionam entre si. As rejeições acontecem quando as palavras violam as regras da língua que regem a combinação de palavras. Estas regras são fornecidas pela gramática da língua.

Para cada sentença analisada, o analisador (*parser*) gera uma árvore sintática (ou mais de uma se a sentença for ambígua) segundo uma gramática específica. Um analisador mapeia um conjunto de palavras em um conjunto de padrões sintaticamente significativos. A sentença **O menino chutou a bola** gera a árvore sintática da Figura 3-1.

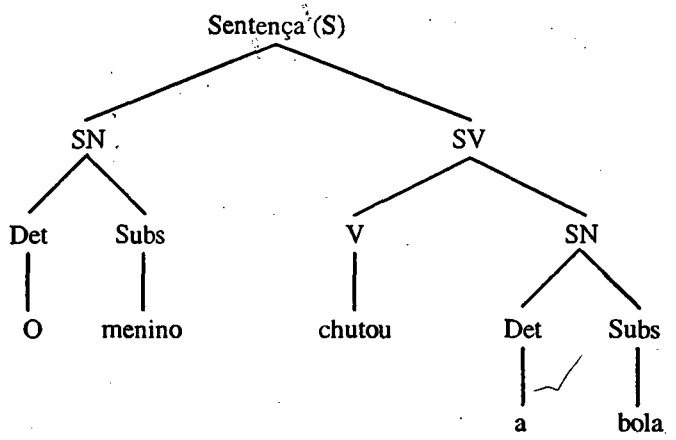

Figura 3-1 : Árvore sintática para a sentença **O menino chutou a bola.**

3.4.2 Análise Semântica

A análise semântica faz a designação de significados das estruturas criadas pelo analisador sintático. Faz-se um mapeamento entre as estruturas sintáticas e os significados no domínio de atividade. As rejeições acontecem quando esse mapeamento não é possível.

Há um problema quando o analisador sintático precisa decidir entre diversas estruturas possíveis para uma determinada frase. Nesse caso, as informações semânticas servem para decidir qual a estrutura sintática correta.

Pode-se classificar as abordagens desenvolvidas para solucionar este problema da seguinte maneira:

a) Gramáticas Semânticas;
b) Gramáticas de Casos;
c) Filtragem semântica das estruturas sintáticas geradas;
d) Desenfatizar a análise sintática e acionar o processo de conhecimento semântico e não sintático (conhecido como analisadores livres de sintaxe).

3.4.3 Análise Pragmática

Na análise pragmática faz-se a reinterpretação da estrutura que representa o que foi dito para se determinar o que realmente se quis dizer. Por exemplo, a frase **Você tem horas?** Se interpretada somente sob o ponto de vista sintático-semântico levará a uma resposta do tipo **Sim, eu tenho** ou **Não, eu não tenho**. Sob o enfoque pragmático, deve ser interpretada como uma solicitação para se informar as horas.

3.5 Gramática e *Parsing*

Uma gramática de uma língua é um esquema para especificar as sentenças permitidas naquela língua, indicando a regra sintática para combinar palavras em frases e cláusulas [BAF81]. Em programas de PLN a gramática é utilizada para analisar sentenças e ajudar a determinar seus significados, e assim obter uma resposta apropriada. Vários tipos diferentes de gramáticas são usados em programas de LN, incluindo: gramáticas de constituintes imediatos (PSG; *Phrase-Structure Grammar*), gramáticas transformacionais, gramáticas de casos, gramáticas sistemáticas e gramáticas semânticas [BAR96b].

O *parsing* faz uso de regras gramaticais e outras fontes de conhecimento para determinar as funções das palavras na sentença de entrada (uma cadeia linear de palavras) a fim de criar uma estrutura de dados conhecida como árvore de derivação. Esta estrutura ilustra algumas das relações entre palavras na sentença e pode ser usada para obter o significado da sentença.

3.6 Gramática Formal

Uma linguagem formal é definida como um conjunto de cadeias de tamanho finito tomada de um vocabulário (uma cadeia é uma seqüência de símbolos formada a partir de um alfabeto - por exemplo, uma palavra da língua portuguesa).

Formalmente, uma gramática G é definida por uma quádrupla (N, T, P, S) representando os não terminais (N), terminais (T), as regras de produções ou de reescrita (P) e o símbolo inicial (S). O vocabulário (V) representa a união dos conjuntos N e T, que não possuem elementos em comum. Cada regra de produção em P é da forma:

$$X \rightarrow Y$$

onde X e Y pertencem a V.

O formalismo gramatical pode ser classificado por sua capacidade gerativa, que é o conjunto de linguagens que uma gramática pode representar. Existem 4 tipos de gramáticas que diferem somente na forma das regras de reescrita. Esta classificação é conhecida como hierarquia de Chomsky:

Tipo 0 : Este tipo de gramática utiliza regras irrestritas. Ambos os lados das regras de reescrita podem ter qualquer número de símbolos terminais e não terminais. Estas gramáticas são equivalentes à máquina de Turing em seu poder expressivo. Exemplo de uma regra desta gramática:

$$XY \rightarrow C$$

Tais gramáticas são computáveis, mas indecidíveis.

Tipo 1 : Também conhecida como Gramática Sensível ao Contexto (GSC). A diferença da GSC para a gramática do tipo 0 é que o número de símbolos no lado direito de uma regra na GSC deve ser maior ou igual ao número de símbolos no lado esquerdo da regra e o símbolo inicial não pode aparecer no lado direito de uma regra de produção. Exemplo:

$$AXB \rightarrow AYB$$

Gramáticas do tipo 1 são decidíveis.

Tipo 2 : Chamada Gramática Livre de Contexto (GLC), é a mais popular para gramáticas de LN. A restrição é que cada regra deve ter somente um símbolo não terminal no lado esquerdo. Exemplo:

$$S \rightarrow aSb$$

Este tipo de gramática possui complexidade polinomial.

Tipo 3 : Gramática Regular (GR). Toda regra de produção para gramáticas regulares é da forma:

$$X \rightarrow aY \quad \text{ou} \quad X \rightarrow a$$

onde X e Y são símbolos não terminais e a é um terminal.

Este tipo de gramática possui complexidade linear.

3.7 Gramáticas de Constituintes Imediatos

Uma gramática de constituintes imediatos [BAR96b] (*Phrase Structure Grammar* - PSG) é uma GLC, em que os não terminais são sintagmas ou categorias gramaticais. Os terminais são palavras da língua. Assim, esta gramática fornece a estrutura gramatical de uma sentença através de seus constituintes, mostrando a hierarquia entre os mesmos. Por exemplo, uma PSG para a frase **O menino chutou a bola**, cuja árvore sintática é mostrada na Figura 3-1, será:

S → NP VP
NP → Det Subs
VP → V NP
Det → **o** | **a**
Subs → **menino** | **bola**
V → **chutou**

As gramáticas PSG são interessantes do ponto de vista computacional, porque como uma GLC, existem vários algoritmos eficientes para avaliá-las.

3.8 Redes de Transição

Uma rede de transição (RT) é utilizada em lugar de regras para definir uma linguagem. Ela é utilizada sozinha ou como um conjunto de redes de transição em uma gramática. Uma rede de transição consiste de um conjunto de estados conectados por arcos. Alguns estados são designados como estados iniciais e outros como estados finais, os demais são estados intermediários. Uma Rede de Transição Simples (RTS) consiste de uma rede cujos arcos são rotulados como palavras ou categorias gramaticais.

A Figura 3-2 mostra uma RTS para um NP do tipo: **A menina bonita**. Observe que não existe um arco para um nó anterior a este nó (não é recursiva), mas podem existir *loops* (ex. o arco (3,3)). As RTSs são equivalentes às máquinas de estados finitos.

Uma rede de transição recursiva (RTR) possui uma rede para cada símbolo não terminal da gramática, ou seja, um rótulo de um arco pode ser uma chamada para outra RT da gramática. Os arcos são rotulados com categorias gramaticais e não terminais. Veja Figura 3-3. As redes de transição recursiva são equivalentes às gramáticas livres de contexto.

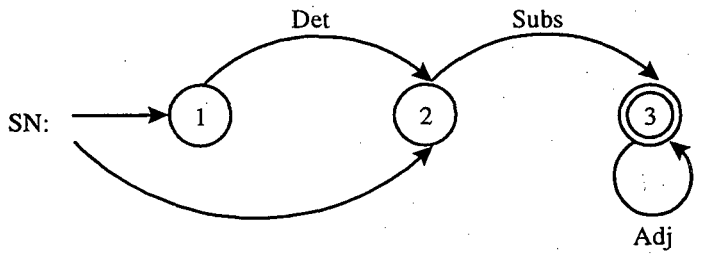

Figura 3-2 : Rede de Transição Simples.

Por último, tem-se as Redes de Transição Aumentadas (RTA) que são RTR acrescidas de condições e ações associadas aos arcos. Para "atravessar" um determinado arco, uma condição deve ser satisfeita (deve ser verdadeira). A seguir tem-se mais um exemplo adaptado de [BAR96b] para ilustrar uma RTA, que é a RTR da Figura 3-3 acrescida das ações e condições da Tabela 3-1 para a rede do NP.

Arco	Condição	Ação
1-2	—	Det := determinante Conc := conc.det
2-3	Conc ∩ Conc.det	Núcleo := substantivo Conc := Conc.det ∩ Conc.subs
1-3	—	Subs := substantivo Conc := Conc.subs

Tabela 3-1 : Condições e ações para a RTR do NP do exemplo da Figura 3-3 em um RTA.

As RTAs são equivalentes as GSCs, pois as condições dos arcos podem ser vistas como um contexto associado.

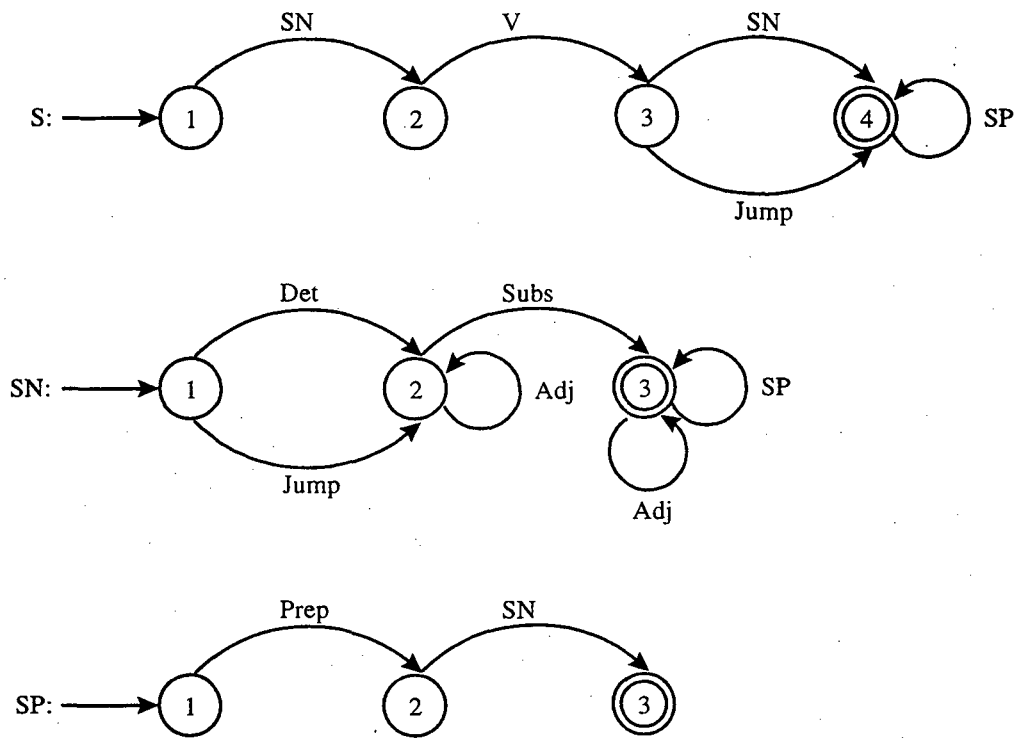

Figura 3-3 : Rede de Transição Recursiva [BAR96b].

3.9 Gramática de Casos

A gramática de casos, diferentemente das gramáticas sintáticas (RTA, GLC, etc.), atribuem papéis temáticos, ou casos, aos componentes de uma frase – papéis esses que devem ser independentes da distribuição desses componentes na estrutura sintática da frase.

O objetivo desta análise é representar o significado de uma frase. Frases com significados diferentes deverão ter representações diferentes e frases com o mesmo significado deverão resultar na mesma representação, como em:

João abriu a porta;
A porta foi aberta por João.

As duas frases possuem o mesmo agente = **João** e paciente = **porta**, ainda que elas tenham estruturas sintáticas diferentes.

3.9.1 Atribuições de Papéis Temáticos

Nesta subseção será considerado um aspecto na compreensão de sentenças que é a atribuição de papéis temáticos às constituintes de uma sentença.

Para ilustrar as dificuldades na determinação dos papéis de caso[14] em sentenças, tem-se alguns exemplos de sentenças com o verbo **quebrar**:

1. **O garoto quebrou a janela;**
2. **A pedra quebrou a janela;**
3. **A janela quebrou;**
4. **O garoto quebrou a janela com a pedra;**
5. **O garoto quebrou a janela com a cortina.**

Pode-se ver que a atribuição de casos é bem complexa. O primeiro NP das sentenças 1, 4 e 5 pode ser o agente, o instrumento na sentença 2, o paciente na sentença 3. O NP do sintagma preposicional (PP) pode ser o instrumento na sentença 4 ou o modificador na sentença 5 [McK86].

Outro exemplo mostra a ambigüidade na atribuição de casos:

6. **O garoto comeu macarrão com molho;**
7. **O garoto comeu macarrão com garfo.**

Na sentença 7 o PP é o instrumento, o que não acontece na sentença 6. Nesta o PP é o modificador do paciente.

Pelos exemplos dados pode-se observar que os significados das palavras nestas sentenças influenciam a determinação dos casos (restrições semânticas). Entretanto, a posição das constituintes nas sentenças também é importante – considere os dois casos a seguir:

8. **O vaso quebrou a janela;**
9. **A janela quebrou o vaso.**

Nestes casos deve-se levar em conta as ordens das constituintes para se determinar o caso correto. Estas restrições de ordem das constituintes variam de língua para língua, em algumas, elas são mais fortes que em outras.

Deve-se lembrar que não há um conjunto universalmente aceito no número de casos, nem uma concordância universal de atribuição de casos às constituintes de uma sentença [McK86]. A maioria dos autores utilizam aqueles introduzidos por Filmore, [FIL68].

---

[14] Serão considerados casos, papéis ou papéis de casos como palavras sinônimas.

3.10 Roteiros

De acordo com a teoria de roteiros, as pessoas organizam o conhecimento de rotinas diárias em forma de seqüência de eventos estereótipos, ou roteiros [SCA77]. Este conhecimento compartilhado torna possível fazer tarefas sociais eficientemente, como: ir a um restaurante, ir ao médico, ao cinema, fazer compras no supermercado, viajar de avião, etc. Todos os participantes nestas situações compartilham o mesmo roteiro e eles sabem exatamente qual evento ocorrerá, o que esperar dos outros participantes e o que é esperado dele. Por exemplo, o consumidor no restaurante sabe que o garçom vai trazer o *menu* e comunicar o pedido à cozinha.

ROTEIRO DE RESTAURANTE	
Trilha de restaurante a la carte	
Cadeia Causal :	**Papéis** :
Entrar	Consumidor : **José**
Sentar	Restaurante = **Galetus**
Pedir	Comida = **faisão**
Comer	Gosto = **bom**
Pagar	Gorjeta = **grande**
Dar gorjeta	
Sair	

Tabela 3-2 : Representação de uma história baseada em roteiros como uma cadeia causal de ligação de papéis.

O conhecimento de roteiros também faz a comunicação mais eficiente. Em textos narrativos, muitos detalhes não são colocados, porque eles podem facilmente ser preenchidos pelo leitor usando o conhecimento do roteiro, por exemplo:

José foi ao Galetus. José pediu faisão ao garçom. José deixou uma boa gorjeta.

Pode-se compreender a última sentença assumindo os eventos intermediários usuais no roteiro de restaurante: **O garçom trouxe o faisão para o José, José comeu o faisão, o faisão estava gostoso, José pagou o garçom, José deixou uma boa gorjeta** (porque o serviço/comida estavam bons). Em textos mais complexos existem referências a vários roteiros e a compreensão requer reconhecer e aplicar vários roteiros ao mesmo tempo. O roteiro é uma ferramenta para construção da representação completa de histórias.

Roteiros também suportam uma teoria computacional. Eles são baseados na idéia geral de esquemas como a base para representação mental [ARB89]. De acordo com a teoria de esquema as pessoas organizam o conhecimento sobre objetos familiares, situações e procedimentos em termos de protótipos ou esquemas [MII93]. Um esquema consiste de uma representação para o conhecimento comum que é compartilhado por todas as instâncias e um número de escaninhos que possuem diferentes valores para diferentes instâncias. Por exemplo, um esquema (roteiro) para uma ida ao restaurante especifica os eventos comuns (como entrar, sentar, pedir, comer, etc.) e escaninhos para consumidor, restaurante, comida, gorjeta, etc.

Um roteiro é representado como uma cadeia causal de eventos (PLN simbólica) com um número de papéis abertos [SCA77] (Tabela 3-2). Diferentes variações do mesmo roteiro são chamadas de trilhas, por exemplo, existe a trilha restaurante *a la carte*, trilha *fast-food* e café-restaurante, para o roteiro de restaurante, cada um com eventos diferentes.

Para aplicar o conhecimento de roteiros para uma história necessita-se identificar o roteiro relevante e preencher seus papéis com as constituintes da história. Este procedimento é chamado de instanciação do roteiro.

Uma vez que o roteiro foi reconhecido e os papéis foram instanciados as sentenças são comparadas com os eventos no roteiro. Os eventos que não foram mencionados na história e fazem parte da cadeia causal podem ser inferidos, bem como certos preenchedores de papéis que não foram especificados na história. Por exemplo, na história acima é plausível assumir que **José comeu o faisão** e que ele estava gostoso. A compreensão é geralmente demonstrada gerando-se uma paráfrase aumentada da história original e respondendo perguntas sobre ela.

4. Processamento de Linguagem Natural Conexionista

4.1 Introdução

Na década de 80 houve uma grande difusão da abordagem conexionista em IA, após ter sido demonstrado [RHW86] que se poderia superar as limitações de RNAs que Minsky & Papert [MIP69] demonstraram nos anos 60. Também foram mostradas várias áreas de aplicações para RNAs. Havia, entre as aplicações, uma rede para aprender o tempo passado de verbos em inglês [RUM86], demonstrando uma possibilidade inicial do uso de RNAs em tarefas de processamento de linguagem natural (PLN). Processamento de linguagem natural conexionista (PLNC) é uma subárea do conexionismo que especializou-se em tarefas de PLN, e trabalhos na área incluem *parsing* [JAW90, SEH85, JAC92], processamento de roteiros (*scripts*) [MII93], atribuição de papéis temáticos à constituintes de sentenças [McK86, MII95, MII96], aprendizagem de linguagem [FEL93, TOU91], geração [KAM90] e *parsing* híbrido (conexionista/simbólico) [WAP85].

Existem várias razões para se abordar processamento de linguagem natural dentro do paradigma conexionista da IA. Para muitos pesquisadores a abordagem conexionista fornece uma nova maneira de lidar com velhas questões [SCR92]. Por exemplo, sistemas conexionistas podem aprender através de exemplos, assim no contexto de um sistema baseado em regras, todas as regras não necessitarão ser especificadas a *priori*. Sistemas conexionistas possuem uma grande capacidade de generalização, memória endereçável ao conteúdo e são ótimos para lidarem com informações incompletas.

Os primeiros trabalhos em PLNC apareceram no início de 80 [HIN81, SCS82] e as pesquisas foram intensificadas desde o renascimento do conexionismo nos meados dos anos 80. Um artigo que influenciou o conexionismo e incentivou as pesquisas em PLNC e representações conexionistas, foi a crítica ao conexionismo feita por Fodor & Pylyshyn [FOP88]. Este artigo fez uma crítica ao conexionismo enquanto uma ferramenta para tarefas cognitivas, e a resposta a ele envolveram pesquisas mostrando como sistemas conexionistas podem realizar tarefas como representar estruturas de dados recursivas [POL90, HIN91] e codificar estruturas temporais [ELM90]. Freqüentemente, os sistemas conexionistas retêm suas propriedades de robustez, generalização e degradação progressiva.

Uma importante área para PLNC, são as representações conexionistas. Sistemas simbólicos (incluindo sistemas de PLN), geralmente utilizam estruturas recursivas para representar o conhecimento, enquanto RNAs são geralmente usadas para tarefas de baixo nível como reconhecimento de padrões, onde não são exigidas representações de estruturas complexas. No entanto, para construir sistemas em PLNC com poder comparável aos sistemas simbólicos, deve-se criar mecanismos para representar estruturas complexas. Pesquisas em representações conexionistas são altamente relevantes para PLNC, e os dois campos são discutidos juntos.

Neste capítulo serão descritos apenas modelos conexionistas distribuídos (sub-simbólico). Existem dentro de PLNC os chamados modelos locais (*localist*; veja [SUB95] que descreve alguns modelos locais e [WAP85]) e os modelos híbridos [WER95, WAP85, DYE91, LEE91, HEN89], para maiores detalhes consulte a referência.

4.2 O Problema do Tempo em Linguagem

Sabe-se que o tempo é importante em muitas das tarefas cognitivas encontradas na prática, como visão, linguagem (escrita e falada), processamento de sinal e controle motor. Nesta seção serão mostradas algumas técnicas que lidam com o problema da seqüencialidade da linguagem.

Quando se lê um texto ou ouve-se alguém, os sinais capturados (morfemas ou fonemas) através dos órgãos sensoriais humanos (olhos e ouvidos) entram de uma forma seqüencial, um após o outro. Sendo assim, um sistema de linguagem natural deveria possuir algum mecanismo para lidar com esta seqüencialidade inerente à linguagem.

4.2.1 Representação Espacial

Os primeiros modelos conexionistas para compreensão de linguagem natural usaram representações espaciais para seqüências de linguagem natural. Um destes modelos foi para aprender o tempo passado de verbos em inglês [RUM86]. Este modelo demonstrou que o tempo passado de verbos poderia ser aprendido com uma rede conexionista que não contém nenhuma regra simbólica explícita. A entrada para esta rede é um verbo e a saída é a forma passada deste verbo. Cada palavra é representada como um conjunto de 3-upla chamadas *Wichelphones*. A seqüencialidade não era representada explicitamente neste modelo, mas implicitamente na representação paralela de superposição de 3-upla de *wichelphones*.

Outro modelo, que usou uma representação paralela espacial, concentrava–se na atribuição de constituintes de sentenças a papéis temáticos [McK86]. A rede aprendeu e representou o conhecimento sintático e semântico para realizar a tarefa. A entrada é uma representação da estrutura superficial de uma sentença, isto é, para uma sentença como: **o menino quebrou a janela com uma pedra**, a entrada é uma representação semântica (codificada manualmente) para **quebrou**, **menino**, **janela** e **pedra** nesta ordem. A saída deste modelo é a representação de papéis de caso, isto é, para o evento **quebrou** o papel de agente é **menino**, o papel de paciente é **janela** e o papel de instrumento é **pedra**. Enquanto este modelo poderia lidar com diversos problemas difíceis de desambiguação estrutural e lexical, sua representação estava restrita em tamanho pelos 4 conjuntos predeterminados de unidades de estruturas superficiais (um verbo e três substantivos) e quatro conjuntos de unidades de papéis de casos (agente, paciente, instrumento e modificador). A seqüencialidade era representada implicitamente pela ordem das constituintes sintáticas da sentença que eram em número fixo, assim sentenças maiores e mais complexas não poderiam ser processadas.

4.2.2 O Princípio da Janela

Ao invés de apresentar a seqüência inteira à rede, apenas parte da seqüência (janela) torna-se a entrada para a rede. Movendo a janela através de toda a seqüência, poder–se–ia processar seqüências de comprimento variado. Esta técnica de janela deslizante foi usada na arquitetura NETtalk [SER87]. Uma janela de sete letras era deslocada sobre o texto e a tarefa da rede era produzir o fonema da letra central que correspondia à quarta das sete letras. No entanto, uma janela deslizante contendo parte de uma seqüência tinha a desvantagem de restringir o contexto seqüencial pelo comprimento da janela. Em geral estes modelos espaciais paralelos para

representar seqüências têm um poder limitado por causa das restrições no comprimento da seqüência ou no comprimento da janela.

4.2.3 Redes com Retardo de Tempo

Uma diferente classe de redes conexionistas que foi desenvolvida para o aprendizado de seqüências no processamento da fala são as redes neurais com retardo de tempo (TDNN - *Time Delay Neural Network* [WHH89]).Cada unidade TDNN recebe a entrada corrente bem como as entradas do passo anterior. No entanto, cada unidade pode manter-se a par da história dos valores das entradas, desta maneira a rede pode suportar seqüencialidade. Baseadas nestas unidades TDNN, várias arquiteturas TDNN foram construídas, por exemplo, para reconhecimento de fonemas [WHH89].

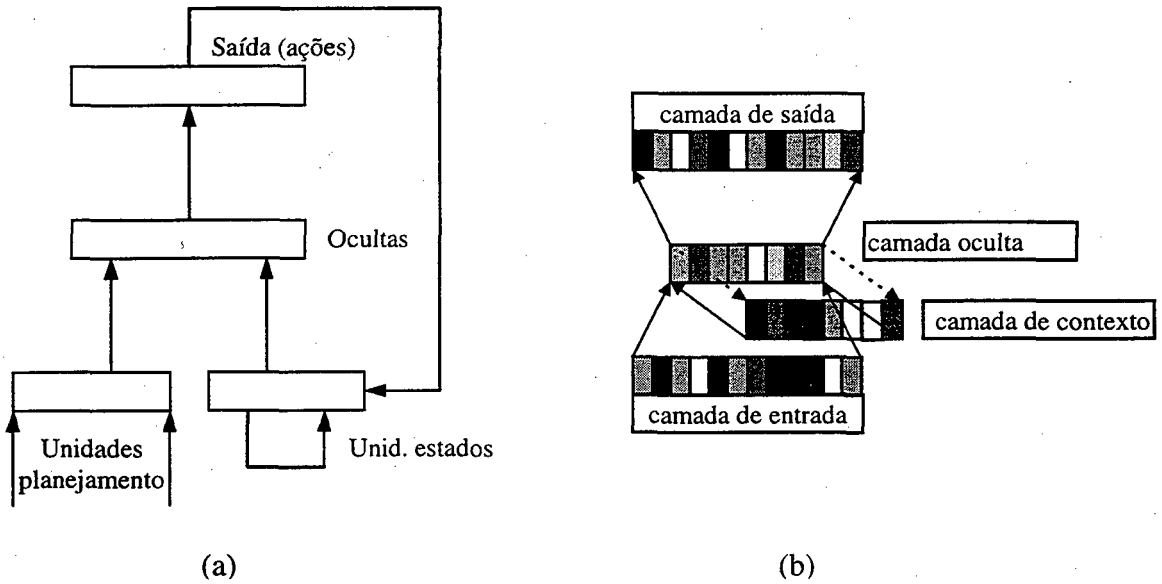

Figura 4-1 : a)Rede Recorrente de Jordan; b)Rede recorrente simples. As camadas que estão conectadas com setas cheias são inteiramente conectadas, e as que estão com linhas pontilhadas possuem conexões na base de um para um.

4.2.4 Redes Recorrentes

As redes recorrentes são mais adequadas para processamento seqüencial, por isso são as mais utilizadas. Além das redes descritas a seguir, existem outros modelos de redes neurais recorrentes que implementam linguagens regulares e livre de contexto [GHL95, TIN96, OMG96, WAK92, GMC92, ROI96].

4.2.4.1 Rede Recorrente de Jordan

Enquanto as representações espaciais e janelas deslizantes usam um comprimento fixo do contexto seqüencial, modelos recorrentes podem representar arbitrariamente seqüências longas. Jordan propôs um modelo recorrente para processar seqüências que representam planejamento, estados e ações como vetores distribuídos [JOR86]. Como ilustra a Figura 4-1a, as unidades

planejamentos e as unidades estados são as unidades de entrada da rede que são conectadas às unidades ações (as unidades de saída) via uma camada de unidades ocultas. As unidades ações são ligadas às unidades de estados como conexões recorrentes, e as unidades estados também têm conexões recorrentes para elas mesmas para representar diretamente o estado precedente. Para um certo plano esta rede gera uma seqüência correspondente de ações de saída.

4.2.4.2 Rede Recorrente Simples

Uma rede recorrente simples (SRN - *Simple Recurrent Network* [ELM90]) consiste em adicionar, a uma rede direta de 3 camadas, unidades ocultas como se fossem unidades de entrada chamadas unidades de contexto. Estas unidades são inteiramente conectadas à camada oculta do mesmo modo que a camada de entrada. Porém, a camada de contexto recebe conexões recorrentes da camada oculta, formando assim uma rede recorrente (veja Figura 4-1b). Estas unidades de contexto também são unidades ocultas porque elas interagem exclusivamente com unidades internas à rede, e não com o ambiente externo.

Com redes recorrentes pode-se processar seqüências de comprimento arbitrário. No tempo *t*, as unidades de entrada recebem o primeiro item da seqüência. As unidades de entrada e de contexto (inicialmente possuem valores igual a 0,5) [15] ativam as unidades ocultas. As unidades ocultas ativam as unidades de saída e também as unidades de contexto através das conexões recorrentes. A saída é comparada com a saída desejada e o algoritmo de retropropagação (seção 2.5.3.1) é usado para ajustar os pesos das conexões. Os pesos das conexões recorrentes (da camada oculta para a camada de contexto), que são fixos, valem *1* e não são ajustados pelo algoritmo. No próximo passo *t+1*, as unidades de contexto contêm os valores que são exatamente os valores das unidades ocultas no tempo *t*. Assim, estas unidades de contexto fornecem memória à rede [ELM90].

Pelo fato dos padrões nas unidades ocultas serem salvos como contexto, estas unidades possuem a tarefa de mapear a entrada e, ao mesmo tempo, desenvolver representações que são codificações úteis das propriedades temporais da entrada seqüencial [ELM90]. Elman realizou diversos experimentos em linguagens utilizando-se da rede recorrente simples [ELM90, ELM91a, ELM91b].

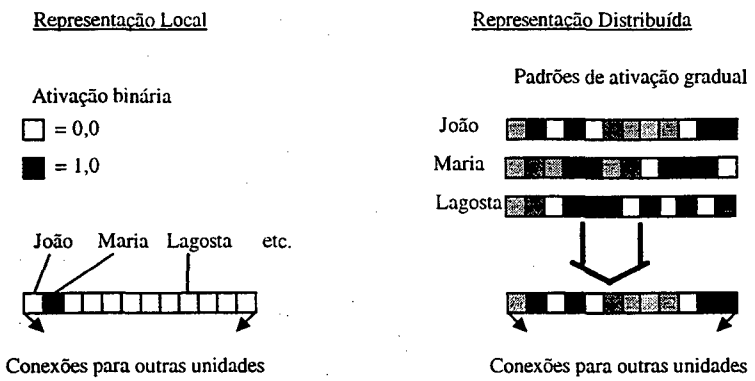

Figura 4-2 : Representação distribuída e local.

[15]Considerando que a saída da função de ativação esteja no intervalo [0,1].

4.3 Processamento Distribuído Paralelo

Processamento distribuído paralelo[16] (PDP) emergiu recentemente como uma alternativa à abordagem simbólica em ciências cognitivas. O objetivo é simular os princípios básicos de processamento de informações no cérebro nos modelos computacionais. As redes distribuídas paralelas operam em nível sub-simbólico, sendo assim elas possuem propriedades bem diferentes dos sistemas simbólicos.

4.3.1 Representações Distribuídas

As arquiteturas PDP são inspiradas e motivadas pela estratégia de representação distribuída das informações no cérebro. Sistemas PDP consistem de um grande número de unidades de processamento densamente interconectadas. Cada unidade faz a soma de suas entradas ponderadas pelos seus respectivos pesos, e produz um valor de saída que é geralmente uma função não linear da soma. O valor de saída é então enviado a outras unidades através das conexões ponderadas. Uma unidade sozinha não representa qualquer item em particular, nem um peso da conexão representa uma relação. Cada unidade e conexão estão envolvidas na representação de diferentes partes da informação. Os itens de dados são representados como padrões diferentes de ativação sobre o mesmo conjunto de unidades (Figura 4-2) e o conhecimento para diferentes situações é sobreposto no mesmo conjunto de pesos das conexões.

A Figura 4-2 mostra a diferença entre representações distribuídas e representações locais. Na representação local cada unidade representa um item em particular. O número de itens é diretamente limitado pelo número de unidades, mas diversos itens podem ser representados em paralelo. Na representação distribuída cada item é representado como um padrão de ativação sobre o mesmo conjunto de unidades e a informação sobre como processar os padrões é sobreposta no mesmo conjunto de pesos das conexões. O número de itens que podem ser representados depende da precisão dos valores de ativação das unidades. Somente um item pode ser representado distintamente de cada vez.

4.3.2 Propriedades de Modelos Distribuídos Paralelos

Modelos distribuídos paralelos têm diversas qualidades atrativas. Elas são baseadas em 4 propriedades básicas de representações distribuídas [PLA94]:

a) As representações são em valores contínuos;
b) Conceitos similares têm representações similares;
c) Várias partes diferentes do conhecimento são sobrepostas no mesmo conjunto de unidades;
d) As representações são holográficas, isto é, qualquer parte da representação pode ser usada para reconstruir o todo.

Pelas duas primeiras propriedades, observa-se que as representações refletem os significados dos conceitos que elas representam. Pelo fato delas serem contínuas, é possível representar pequenas diferenças de significados, e representações de membros de categorias possuem pequenas diferenças.

---

[16] Também chamado rede neural artificial distribuída ou conexionismo distribuído [MII93].

A terceira propriedade resulta em generalização espontânea. O conhecimento sobre um item é automaticamente generalizado para todos os outros itens que são similares a ele. Contanto que as situações de entrada consistam de elementos familiares, o sistema pode generalizar razoavelmente bem em novas situações.

A terceira propriedade também resulta em degradação progressiva quando o sistema é sobrecarregado com informações. Quando diversas representações são sobrepostas no conjunto de unidades, variações individuais tendem a anular-se, e tendências centrais são realçadas. Não há um limite fixo da quantidade que pode ser armazenada, mas as representações tornam-se menos e menos precisas.

A propriedade holográfica torna o sistema robusto contra ruídos, danos e informações incompletas. Devido ao fato de que a mesma informação é representada em vários lugares, o processamento é efetivamente baseado na média de diversas representações. Mesmo quando o padrão de entrada está incompleto, o sistema pode usar o resto do padrão para reconstruir as informações que faltam. Assim, os sistemas distribuídos são inerentemente endereçáveis ao conteúdo.

4.4 Recirculação de Símbolos

O que se busca é um método pelo qual os símbolos tenham um relacionamento recursivo e estruturado com outros símbolos, mas sem utilizar microcaracterísticas, e ao mesmo tempo formar padrões distribuídos de ativações. A técnica geral para realizar este objetivo é conhecida como recirculação de símbolos. Os símbolos são mantidos numa rede conexionista separada que atua como uma memória de símbolos global (léxico), onde cada símbolo é composto de um padrão de ativação. Com o tempo eles são "recirculados" e gradualmente vão se formando representações distribuídas destes símbolos enquanto a rede aprende a tarefa.

As representações para os símbolos surgem do resultado de ensinar o sistema a formar o mapeamento associativo entre os símbolos, onde esses mapeamentos capturam relacionamento estruturado.

A técnica básica de recirculação de símbolos envolve:

a) Começar com uma representação arbitrária para cada símbolo;
b) Colocar estes símbolos nas camadas de entrada e saída de uma rede que efetua a tarefa de mapeamento;
c) Formar uma representação distribuída de símbolos que auxilia na tarefa do mapeamento;
d) Armazenar as representações dos símbolos de volta na memória de símbolos global;
e) Interagir em todos os símbolos para toda a tarefa de mapeamento, até todas as representações convergirem.

Como os mesmos pares de entrada e saída são apresentados à rede que faz o mapeamento, as representações dos símbolos que compõem estes pares sofrem modificações. Assim, o sistema está "disparando em um alvo em movimento" (*shooting at a moving target*), uma vez que as representações estão sendo alteradas enquanto a rede está aprendendo o mapeamento. Não somente os pesos estão sendo modificados, mas também as codificações das representações, dessa maneira o ambiente de treinamento é reativo.

Padrão de entrada	Representação camada oculta	Padrão de saída
A+N	$R_{AN}(t)$	A´+N´
D+$R_{AN}(t)$	$R_{DAN}(t)$	D´+$R_{AN}(t)$´
D+N	$R_{DN}(t)$	D´+N´
P+$R_{DN}(t)$	$R_{PDN}(t)$	P´+$R_{DN}(t)$´
V+$R_{PDN}(t)$	$R_{VPDN}(t)$	V´+$R_{PDN}(t)$´
$R_{DAN(t)}$+$R_{VPDN}(t)$	$R_{DANVPDN}(t)$	$R_{DAN(t)}$´+$R_{VPDN}(t)$´

Tabela 4-1 : Representações para a árvore *((D(A N))(V(P(D N))))*. Observe que a rede primeiro desenvolve representações para todas as sub árvores até obter a representação da árvore.

4.4.1 Memória Autoassociativa Recursiva

Pollack desenvolveu uma técnica para representar estruturas recursivas, como seqüências e árvores, chamada RAAM (*Recursive Auto-Associative Memory* [POL90, POL89]). Uma RAAM, automaticamente, desenvolve representações distribuídas destas estruturas usando o algoritmo de retropropagação.

Uma RAAM é composta de dois mecanismos: Um compressor e um reconstrutor que são simultaneamente treinados. A função do compressor é codificar um pequeno conjunto de padrões de tamanho fixo em um padrão de mesmo tamanho. Esta compressão pode ser recursivamente aplicada, das folhas para a raiz, em uma árvore de valência fixa com terminais distintos (folhas), resultando em um padrão fixo representando a estrutura da árvore inteira. A função do reconstrutor é decodificar este padrão nas partes que o compõem, e então decodificar as partes até os padrões terminais serem encontrados, resultando na reconstrução da árvore original.

Figura 4-3 : Redes direta para a) Compressor; b) Reconstrutor; c) Rede composta do compressor e reconstrutor.

Para árvores com folhas de padrões binários com *K* bits, o compressor (Figura 4-3a) será uma rede direta de duas camadas (entrada e saída) com *2K* unidades de entrada e *K* unidades de saída, junto com um mecanismo de controle adicional. O reconstrutor (Figura 4-3b) também será uma rede direta com *K* unidades de entrada e *2K* unidades de saída com um mecanismo para testar se um padrão é terminal (folha).

As duas redes são treinadas simultaneamente (Figura 4-3c) em uma estrutura auto-associativa. Considere a árvore *((D(A N))(V(P(D N))))*, como um membro de um conjunto de árvores binárias que formam o conjunto de treinamento, em que os terminais são pré-codificados como vetores com *K* bits. Se uma rede *2K-K-2K*[17] é treinada com estes padrões, o compressor e o reconstrutor podem formar representações para estas árvores binárias. A Tabela 4-1 mostra as representações para a árvore acima.

Os valores das unidades ocultas (aleatórios inicialmente) $R_i(t)$, fazem parte do ambiente de treinamento, assim elas (e as representações das árvores) evoluem juntas com os pesos. Se a parte reconstruída, digamos R_{PDN}, estiver suficientemente próxima do padrão original, então suas partes podem ser reconstruídas também.

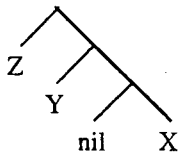

Figura 4-4 : Representação gráfica em árvore da seqüência *(X, Y, Z)*.

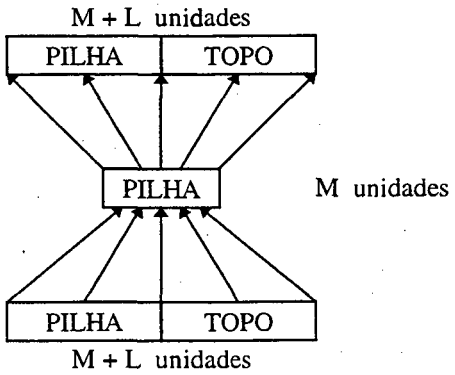

Figura 4-5 : O compressor combina uma representação M-dimensional para uma pilha (PILHA) com um novo elemento (TOPO), retornando um novo vetor M-dimensional. O reconstrutor decodifica-o em suas componentes.

4.4.1.1 Memória Autoassociativa Recursiva Seqüencial

Já que seqüências, como *(X, Y, Z)*, podem ser representadas como árvores binárias (Figura 4-4) com ramificações apenas à esquerda: *(((nil X)Y)Z)*, uma arquitetura RAAM pode desenvolver representações para uma seqüência que se comporta como uma pilha (Figura 4-5).

Esta arquitetura é mais simples do que o mecanismo para árvores. As representações compactadas são recirculadas apenas de um lado, assim elas não têm que ser armazenadas externamente. Há menos restrições no tamanho das representações também, apenas deve-se ter a

[17] Esta notação representa uma rede direta com 3 camadas: 2k unidades na camada de entrada, k unidades na camada oculta e 2k unidades na camada de saída.

dimensão *(M)* da representação dos padrões compactados maior que a dimensão *(L)* dos símbolos terminais.

4.4.1.2 Representações Distribuídas para Árvores Sintáticas

A árvore considerada anteriormente é um membro do 1° experimento feito com RAAMs. Foi usada uma gramática livre de contexto para gerar um conjunto de árvores binárias, utilizando-se a notação de parênteses para árvores:

(D (A (A (N A))))
((D N)(P (D N)))
(V (D N))
(P (D (A N)))
((D N) V)
((D N)(V (D (A N)))))
((D (A N))(V (P (D N))))

Cada padrão terminal (D A N V e P)[18] foi representado como um vetor binário de tamanho 10, com cada vetor possuindo um único componente com valor 1 e os demais 0 (observe que existem 5 bits desnecessários). Então uma rede RAAM 20-10-20 foi treinada para representar as árvores no conjunto de treinamento.

NP	(D N) (D (A (A (A N)))) (D (A N)) ((D N)(P (D N)))
VP	(V (P (D N))) (V (D (A~~ N))) (V (D N)
PP	(P (D N)) (P (D (A N)))
AP	(A N) (A (A N)) (A (A (A N)))
S	((D N) V) ((D N)(V (D (A N)))) ((D (A N))(V (P (D N))))

Tabela 4-2 : Árvores agrupadas segundo o tipo de sintagma.

As árvores e sub árvores foram classificadas manualmente pelo tipo de sintagma (NP, VP, PP, AP e S). A RAAM, sem ter qualquer conceito intrínseco do tipo de sintagma, claramente desenvolveu uma representação com similaridades entre membros do mesmo tipo (Tabela 4-2).

Estes vetores são um novo tipo de representações, uma representação distribuída recursiva como sugerido pela noção de descrições reduzidas por Hinton [HIN88].

[18] Os terminais D, A, N, V e P representam, respectivamente: Determinante, Adjetivo, Substantivo, Verbo e Preposição. Esta gramática é para análise de sentenças simples.

4.4.2 Método para Formação de Representações Distribuídas de Símbolos

O mecanismo FGREP [MID91a, MII93] (*Forming Global Representations with Extended BackPropagation*) é uma extensão do algoritmo de retropropagação em que as representações de I/O, e conseqüentemente o mapeamento entrada/saída, mudam como parte do processo de aprendizado.

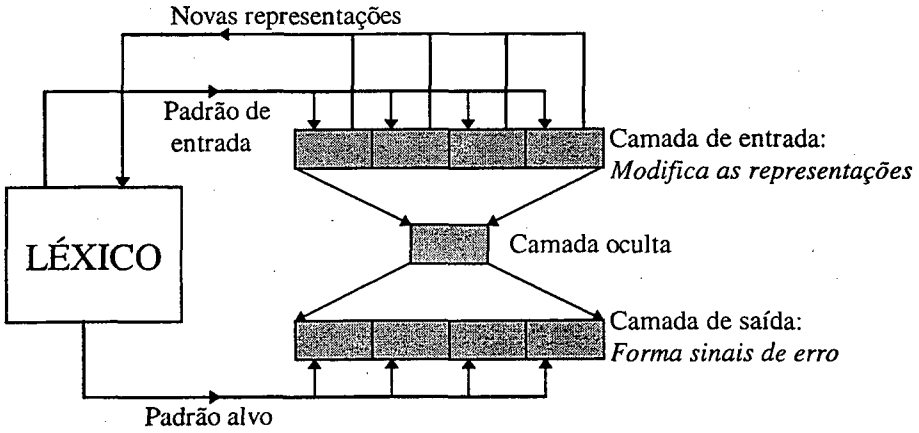

Figura 4-6 : A arquitetura básica de FGREP. O sistema consiste de uma rede direta de 3 camadas e um léxico externo que contém as representações I/O.

O método FGREP é baseado em uma rede direta[19] de 3 camadas treinada com o algoritmo de retropropagação (Figura 4-6). A rede aprende a tarefa adaptando os pesos de acordo com as equações 1.7 a 1.17 do algoritmo de retropropagação. Ao mesmo tempo, representações para os dados de entrada são desenvolvidas baseadas nos sinais de erros estendidos até à camada de entrada.

As representações são armazenadas em um léxico externo. Cada padrão é tirado do léxico e ele forma os padrões de entrada e alvo (saída desejada). Assim cada item é representado pelo mesmo padrão na entrada e na saída da rede.

O processo de formação das representações começa com um léxico aleatório não contendo informações pré-codificadas. Uma componente na representação resultante não tem uma interpretação clara. A representação não implementa uma classificação do item com características identificáveis (como em codificação semântica).

O método consiste em estender o algoritmo de retropropagação até a camada de entrada. Nas unidades de entrada, o valor de ativação é idêntico à componente correspondente na representação. Com esta analogia, a função de ativação é a função identidade e sua derivada é igual a 1. O sinal de erro pode ser calculado para cada unidade de entrada como no caso da equação 2-12:

[19] Também pode ser utilizada uma rede recorrente simples para lidar com entrada (ou saída) seqüencial, como foi feito em [MII93].

$$\delta_i^{\,1} = \sum_k \delta_k^{\,2} w_{ki}^{\,1} \tag{4-1}$$

onde $\delta_y^{\,x}$ representa o sinal de erro da a unidade y na camada x, e $w_{ki}^{\,l}$ é o peso entre a unidade i na camada de entrada e a unidade k na primeira camada oculta. As representações são alteradas de acordo com o sinal de erro:

$$r_{ci} = \eta \delta_i^{\,1} \tag{4-2}$$

onde r_{ci} é a componente i da representação do item c, $\delta_i^{\,l}$ é o sinal de erro da unidade da camada de entrada correspondente, e η é a taxa de aprendizado. O aprendizado da representação é implementado como uma extensão do algoritmo de retropropagação.

Os valores dos pesos são ilimitados, mas os valores das representações devem ser limitados entre os valores de ativação máximo e mínimo das unidades. O novo valor para a componente da representação i do item c é obtido como:

$$r_{ci}(t) = max\left(o_L, min\left(o_U, r_{ci}(t-1) + r_{ci}\right)\right) \tag{4-3}$$

onde o_L é o limite inferior e o_U é o limite superior para a ativação da unidade (0 e 1 são, respectivamente, os limites inferior e superior para a função sigmóide padrão). Note que o algoritmo de retropropagação "vê" as representações simplesmente como uma camada extra de pesos.

4.5 Um *Parser* Conexionista para Análise de Cláusulas Relativas

Nesta seção será descrito um modelo de rede neural distribuída chamado SPEC (*Subsymbolic Parser for Embedded Clauses*) [MII95, MII96] para processar sentenças com cláusulas relativas.

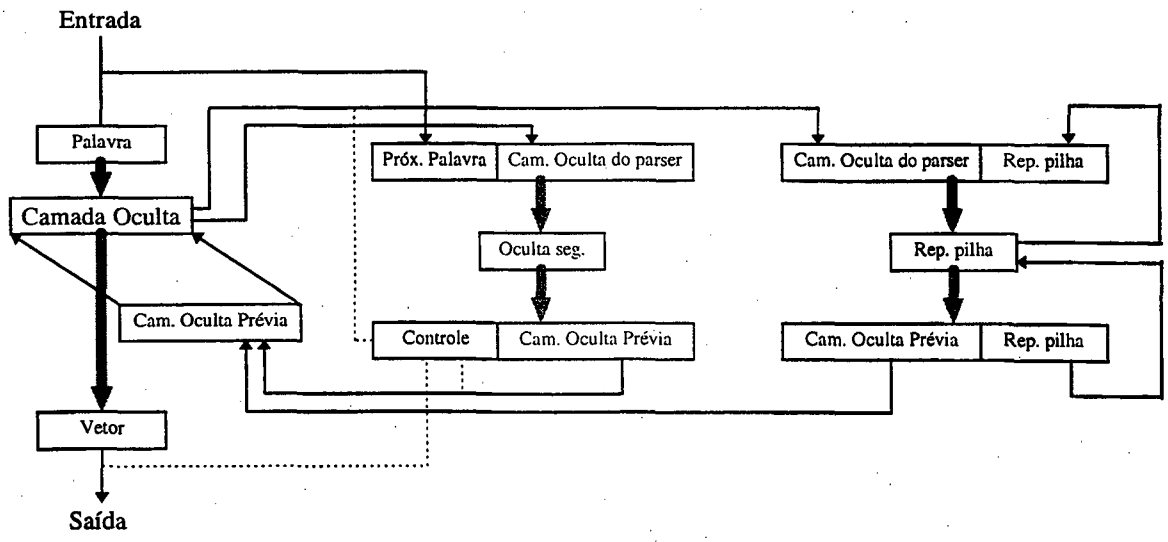

Figura 4-7 : Arquitetura do SPEC - As setas cinza (maiores) representam conexões entre camadas.

SPEC foi especialmente projetado para tratar do problema de generalização para novas estruturas de sentenças, isto é, estruturas que não se encontram nos dados de treinamento.

O componente central de SPEC é uma rede recorrente simples (SRN - *Simple Recurrent Network*) que tem como entrada representações distribuídas de palavras e representações das constituintes em seus papéis temáticos como saída. A capacidade de generalização do modelo é baseada em simplificar a tarefa da SRN através de três inovações na arquitetura do sistema:

a) Treinar a SRN para gerar uma seqüência de representações de constituintes em seus papéis como sua saída (como [STO90]) ao invés de uma só representação compreensiva da sentença como um todo (como [MII90]);
b) Introduzir um Segmentador que quebra a seqüência de entrada em pequenos pedaços;
c) Introduzir uma Pilha que memoriza as constituintes quando encontra uma cláusula aninhada.

4.5.1 Arquitetura do Modelo

A Figura 4-7 ilustra a arquitetura do modelo que consiste de 3 componentes principais:

a) *Parser* (Rede recorrente simples - SRN);
b) Segmentador (Rede Direta);
c) Pilha (Memória Autoassociativa Recursiva - RAAM).

A seguir cada componente será descrito em detalhes.

4.5.1.1 *Parser*

O *parser* é baseado numa rede recorrente simples (Figura 4-8) e transforma a seqüência de palavras em representações das constituintes em seus papéis temáticos. As palavras são representadas por vetores com valores entre 0 e 1. Os valores das componentes são atribuídos aleatoriamente no início e modificados pelo método FGREP como parte do processo de aprendizado, mas SPEC não é dependente de FGREP.

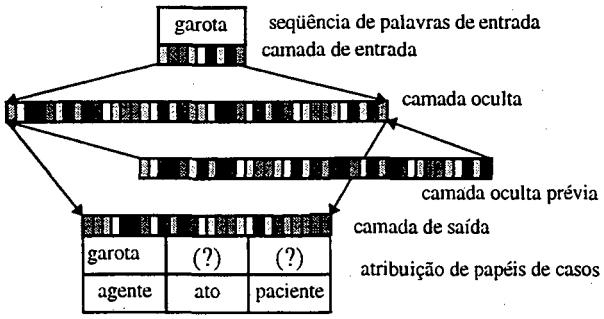

Figura 4-8 : A rede *Parser*.

A atribuição das constituintes em seus papéis temáticos é representada na saída do *Parser* como um vetor (CRV - *Case Role Vector*), isto é, uma concatenação dos três vetores (cada um representando uma palavra) que preencherão os papéis de agente, ação e paciente na sentença

(Figura 4-8). Por exemplo, a seqüência de palavras **A garota viu o garoto** recebe a atribuição de papéis de caso: agente=**garota**, ação=**viu** e paciente=**garoto**, que é representado pelo vetor |**garota viu garoto**| na saída da rede *Parser*.

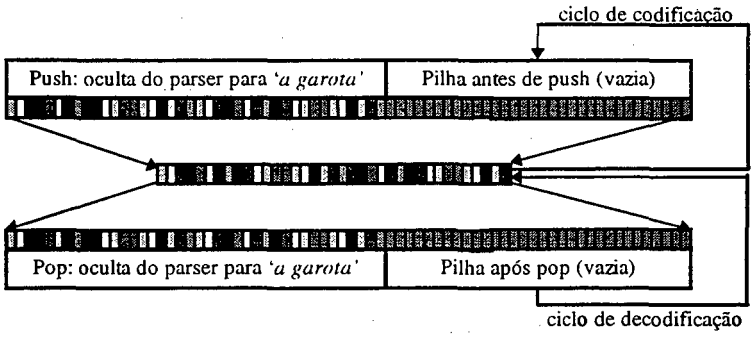

Figura 4-9 : A rede Pilha

4.5.1.2 A Pilha

A Pilha tem a tarefa de armazenar a descrição compactada da seqüência da camada oculta da SRN (*Parser*) em cada ponto onde se começa uma sentença aninhada, e restaurá-la após o *Parser* processar a sentença aninhada. Por exemplo, ao analisar **A garota, que adorou o cão, viu o garoto**, a representação da camada oculta é colocada na Pilha após processar **A garota** e recolocada na camada oculta prévia após processar **que adorou o cão**. Desta maneira a SRN pode analisar a cláusula principal como se a cláusula aninhada não estivesse lá.

A Pilha é implementada como uma rede RAAM treinada para codificar e decodificar listas lineares (Figura 4-9).

Quando o *Parser* retorna do centro de aninhamento, isto é, quando terminar de processar a cláusula aninhada, o padrão armazenado precisa ser recolocado no *Parser*. Para fazer isso a representação atual da Pilha é colocada na camada oculta da rede Pilha e a ativação é propagada para a camada de saída, dessa maneira obtém-se o padrão armazenado na Pilha. O elemento de saída do topo da Pilha é obtido na partição *Pop* (Figura 4-9), e é colocado na camada culta prévia do *Parser*.

4.5.1.3 O Segmentador

Quando o *Parser* + Pilha são treinados somente com exemplos como **A garota, que adorou o cão, viu o garoto** e sentenças com aninhamento de cauda como **A garota viu o garoto, que perseguia o gato**, a arquitetura generaliza bem para sentenças como **A garota, que adorou o cão, viu o garoto, que perseguia o gato**, mas falha para generalizar sentenças como **A garota viu o garoto, que o cão, que perseguia o gato, mordeu**. A solução é treinar uma rede adicional, o Segmentador (Figura 4-10), para dividir a seqüência de entrada em cláusulas. O Segmentador recebe o padrão atual da camada oculta do *Parser* mais a representação da próxima palavra como sua entrada. Se a próxima palavra está no meio de uma sentença, a saída é similar ao padrão atual da camada oculta do *Parser*. Se a próxima palavra é um pronome relativo, o Segmentador modifica o padrão de modo que só informações relevantes permaneçam.

4.5.1.4 Controle

Os padrões de ativação propagam entre as redes de um jeito muito específico e o processamento de cada rede precisa ser cuidadosamente sincronizado em relação ao que as outras redes estão fazendo. Assim o Segmentador é treinado para realizar o controle da execução entre as redes. Há cinco diferentes tarefas de controle no sistema SPEC:

a) Detectar transições de cláusulas e modificar a memória da seqüência para remover contextos prévios desnecessários, como descrito acima;
b) Reconhecer o fim da sentença, e subseqüentemente limpar a camada oculta prévia (que são todas 0 no início de cada seqüência);
c) Decidir quando colocar a representação da camada oculta do *Parser* na Pilha;
d) Decidir quando retirar a camada oculta prévia da Pilha. Esta tarefa envolve permitir propagação da camada oculta da Pilha para sua camada de saída;
e) Decidir quando a saída CRV do *Parser* está completa, e conseqüentemente, abrir o caminho de saída para o sistema de memória de curta duração que é externa ao modelo.

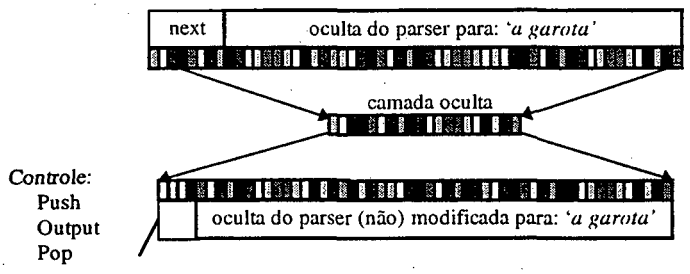

Figura 4-10 : A rede Segmentador

O controle é implementado através de 3 unidades adicionais à saída do Segmentador (Figura 4-10). Elas são chamadas *Push*, *Pop* e *Output*, correspondendo às tarefas (c), (d) e (e) acima.

S	→	NP VP
NP	→	ART N | N RC
VP	→	V NP
RC	→	que VP | que NP V
N	→	garoto | garota | cão | gato
V	→	perseguia | adorou | viu | mordeu
ART	→	o | a

Tabela 4-3 : Gramática que forma as sentenças para treinamento e teste de SPEC.

verbo	papel de caso	preenchedor possível
perseguiu	**Agente:**	garoto, garota, cão, gato
	Paciente:	gato
adorou	**Agente:**	garoto, garota
	Paciente:	garoto, garota, cão
viu	**Agente:**	garoto, garota, gato
	Paciente:	garoto, garota
mordeu	**Agente:**	cão
	Paciente:	garoto, garota, cão, gato

Tabela 4-4 : Restrições semânticas impostas às sentenças.

4.5.2 Experimentos, Treinamento e Resultados

SPEC foi testado com um *corpus* gerado artificialmente de sentenças de cláusulas relativas geradas pela gramática da Tabela 4-3. Esta gramática gera sentenças onde cada cláusula consiste de três constituintes: Agente, Verbo e Paciente. A cláusula relativa **que** poderá ser ligada ao agente ou ao paciente da cláusula pai ou preencher o papel de agente ou paciente na cláusula relativa. Além das palavras **que, o** e **a,** o vocabulário consiste dos verbos **perseguia**, **adorou**, **viu**, **mordeu** e os substantivos **garoto**, **garota**, **cão** e **gato**.

As sentenças contém ainda um número de restrições semânticas, um verbo só poderá ter certos substantivos como seu agente e paciente (veja Tabela 4-4). A gramática foi usada para gerar sentenças com até 4 cláusulas. Considerando as restrições semânticas, o corpus final consistiu de 49 diferentes estruturas de sentenças, com um total de 98100 sentenças diferentes.

1. A garota viu o garoto, que perseguia o gato, que o cão mordeu.
2. A garota , que o cão, que perseguia o gato, mordeu, viu o garoto.

Tabela 4-5 : Sentenças de treinamento de SPEC

Os componentes de SPEC podem ser treinados separadamente, com dados de treinamento compatíveis do mesmo conjunto de exemplos. Neste experimento todos o módulos foram treinados separadamente e simultaneamente numa única máquina. Com esta estratégia é suficiente treinar SPEC somente com as *templates* da Tabela 4-5, porque elas contém todas as construções básicas para o *Parser* e o Segmentador. Os dados de treinamento para a Pilha são obtidos na camada oculta do *Parser* durante o treinamento.

As representações das palavras consistem de 12 unidades. A camada oculta do *Parser* tem 75 unidades, a do Segmentador e da Pilha 50 unidades. Todas as redes foram treinadas com o

backpropagation on line [RHW86] com taxa de aprendizado 0,1 e sem *momentum*. O conjunto de treinamento consistiu de 100 sentenças geradas aleatoriamente das 2 sentenças da Tabela 4-5.

A convergência foi rápida, após 400 épocas, o erro médio por unidade de saída foi de 0,019 para o *Parser*, 0,008 para o Segmentador (0,002 para as saídas de controle), e 0,003 para a Pilha.

O desempenho de SPEC foi então testado no *corpus* inteiro das 98100 sentenças. As saídas nas unidades de controle eram consideradas 1 se fossem maior que 0,7 e 0 se fossem menor que 0,3. Medido dessa maneira, o desempenho foi excelente: SPEC não cometeu nenhum erro no *corpus* todo, nem nas palavras de saída, nem no controle. O erro médio das unidades foram 0,019 para o *Parser*, 0,009 para o Segmentador (0,002 para o controle) e 0,005 para a Pilha.

O resultado principal, portanto, é que a arquitetura de SPEC generalizou com sucesso não só para novas instâncias das estruturas de sentenças familiares, mas para novas estruturas também, que as arquiteturas de processamento de sentenças anteriores não fizeram, como CLAUSES [MII90].

5. Uma Rede Recorrente Simples para Resolução de Anáfora pronominal em Textos de Duas Sentenças

5.1 Introdução

Neste capítulo será descrita a implementação de uma rede recorrente simples (SRN) ou rede de Elman [ELM90] para resolver o problema de referência anafórica pronominal. A rede aprende a tarefa com segmentos de texto compostos de apenas duas sentenças; a primeira sentença fornece o contexto para que a rede possa "saber" a quem o pronome está se referindo (neste trabalho **Ele** ou **Ela**).

Este capítulo tem o objetivo de introduzir o problema a ser resolvido e alguns conceitos para uma melhor compreensão da arquitetura implementada (descrita no próximo capítulo), que é uma variação da arquitetura SPEC discutida na seção 4.5.

5.2 Referências

O termo referência é tradicionalmente usado para indicar como sintagmas em uma sentença se ligam a objetos do mundo real. Um exemplo poderia ser como o NP **a vaca branca** identifica uma **vaca** em particular. Em sistemas computacionais, onde existe uma representação de conhecimento, referência geralmente considera o relacionamento de sintagmas com termos na representação do conhecimento (que representa algum objeto do mundo real).

Existem duas formas de referência à NP: referência anafórica e referência não anafórica [ALL87]. Referência anafórica envolve um NP que refere a um objeto mencionado anteriormente; e em referência não anafórica, o NP identifica um objeto em particular que não foi previamente mencionado. O exemplo a seguir ilustra estas definições.

(1) **João foi ao restaurante. Ele pediu camarão ao garçom.**
(2) **Eu achei uma caneta azul e um lápis preto. A caneta não escrevia.**

O pronome **Ele** na primeira sentença refere-se ao NP **João**, mencionado anteriormente no texto. Este tipo de referência é conhecido como referência anafórica pronominal. O mesmo acontece com o NP **A caneta** na segunda sentença do segundo exemplo. Este tipo de referência é conhecido como referência de sintagma nominal. Os NPs **João** no primeiro exemplo e **A caneta azul** no segundo são exemplos de referência não anafórica.

Neste trabalho será tratada somente a referência anafórica pronominal, utilizando apenas os pronomes pessoais **Ele** e **Ela** com uma estrutura mostrada no exemplo abaixo:

(3) **O garoto viu o cão. Ele perseguiu o gato.**

Neste exemplo, o pronome **Ele** na segunda sentença poderia estar se referindo ao **cão** ou ao **garoto**. A resolução de anáforas consiste em determinar qual dos dois indivíduos corresponderia à melhor referência para este pronome.

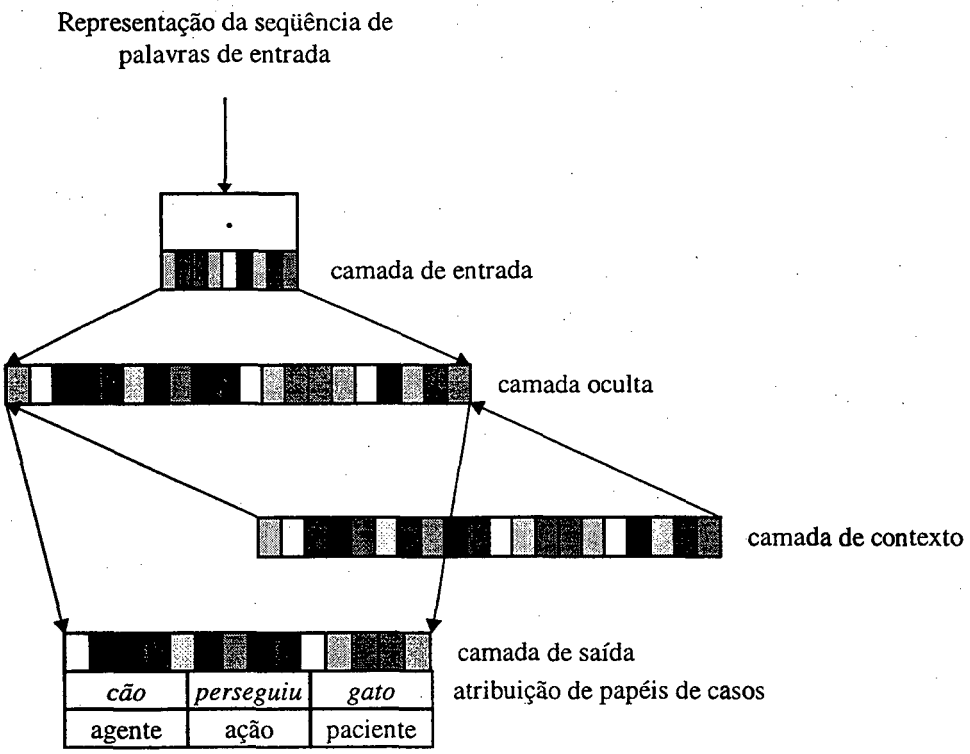

Figura 5-1 : A rede mostra a saída para a última sentença do exemplo **O garoto viu o cão. Ele perseguiu o gato.** Neste caso o pronome **Ele** foi instanciado pelo substantivo **cão.** O ponto marca o fim de cada segmento de texto.

5.3 A Rede

A rede implementada para resolver o problema proposto é mostrada na Figura 5-1, esta rede é conhecida como rede recorrente simples [ELM90]. A rede receberá, como entrada, uma seqüência de palavras como as do exemplo 3, uma palavra de cada vez. Cada palavra possui uma representação armazenada no léxico (veja próxima seção). A saída será uma representação de papéis de caso das constituintes da sentença de entrada. Sendo assim, a rede realizará um mapeamento de uma entrada seqüencial para uma saída estacionária.

As unidades de saída da rede são divididas em partições, com cada partição correspondendo aos casos: agente, ação e paciente.

Considerando o exemplo **O garoto viu o cão. Ele perseguiu o gato** como entrada para a rede, a saída desta será composta por duas cláusulas:

| garoto viu cão |
| cão perseguiu gato |

considerando que o pronome **Ele** está se referindo ao **cão** (devido a algumas restrições semânticas, veja seção 5.4). O papel de agente foi preenchido por **garoto** e **cão** na primeira e segunda sentença, o de ação por **viu** na primeira e **perseguiu** na segunda sentença, e finalmente o papel de paciente foi preenchido por **cão** na primeira sentença e **gato** na segunda. As palavras são

apresentadas à rede uma após a outra, porém a saída é determinada em duas vezes, ou seja, após terem entrado as três primeiras palavras (não contando os artigos o, **a**, **O** e **A**) tem-se uma saída, e após as três últimas palavras é determinada a última cláusula e com ela o substantivo a que o pronome faz referência.

5.4 Dados de Treinamento

As sentenças são constituídas com o mesmo padrão da sentença do exemplo 3, da seguinte forma:

Sujeito verbo objeto. Ele verbo objeto.
Sujeito verbo objeto. Ela verbo objeto.

O conjunto de treinamento será formado por sentenças geradas pela gramática de constituintes imediatos da Tabela 5-1 que gera sentenças compostas por três constituintes: agente, ação e paciente.

S → NP VP.
NP → Det N
VP → V NP CP
CP → **ele** VP | **ela** VP | NP VP | ε
Det → **o** | **a**
V → **viu** | **mordeu** | **adorou** | **perseguiu**
N → **garoto** | **garota** | **homem** | **mulher** | **cão** | **cadela** | **gata** | **gato**

Tabela 5-1 : Gramática de constituintes imediatos que gera o conjunto de treinamento para PCRAP.

A gramática da Tabela 5-1 utiliza um vocabulário semelhante ao utilizado no modelo SPEC. A diferença é que foi retirado o pronome relativo **que** e foram adicionados os pronomes pessoais **Ele** e **Ela** e os femininos dos substantivos já existentes. A Tabela 5-2 mostra o vocabulário que será utilizado para treinar a rede.

Determinantes	**Substantivos**	**Verbos**	**Pronomes**
o, O	garoto, garota	viu	Ele
a, A	homem, mulher	mordeu	Ela
	cão, cadela	adorou	
	gato, gata	perseguiu	

Tabela 5-2 : Vocabulário para o treinamento e teste da rede.

As restrições semânticas são as mesmas do modelo SPEC e são mostradas na Tabela 5-3.

Verbo	papel de caso	preenchedor possível
perseguiu	**Agente:**	garoto, garota, homem, mulher, cão, cadela, gato, gata.
	Paciente:	gato, gata.
adorou	**Agente:**	garoto, garota, homem, mulher.
	Paciente:	garoto, garota, cão, cadela.
viu	**Agente:**	garoto, garota, homem, mulher, gato, gata.
	Paciente:	garoto, garota, homem, mulher.
mordeu	**Agente:**	cão, cadela.
	Paciente:	garoto, garota, homem, mulher, cão, cadela, gata, gato.

Tabela 5-3 : Restrições semânticas impostas às sentenças.

Levando-se em conta a gramática da Tabela 5-1 e as restrições da Tabela 5-3 pode-se formar seis padrões de sentenças de tamanho um, como mostrado na Tabela 5-4. Na Tabela 5-4 as palavras entre chaves: TODOS, GATOS, HUMANOS, GAROTOS e CÃES são substituídas pelos seus respectivos preenchedores (veja Tabela 5-5) que são palavras retiradas do vocabulário da Tabela 5-2. Fazendo uma combinação de todas as possíveis sentenças obtidas pela a substituição das palavras chaves pelos seus preenchedores, obtém-se um total de 72 sentenças de comprimento um. Uma vez geradas estas 72 sentenças faz-se uma combinação destas com elas mesmas para gerar sentenças de tamanho 2. Como existe 72 sentenças de tamanho 1, o total de sentenças de tamanho 2 será de 5184. Estas sentenças são da forma:

Sujeito$_1$ verbo objeto. Sujeito$_2$ verbo objeto.

(1) {TODOS} perseguir {GATOS}.
(2) {HUMANOS} adorar {GAROTOS}.
(3) {HUMANOS} adorar {CÃES}.
(4) {HUMANOS} ver {HUMANOS}.
(5) {GATOS} ver {HUMANOS}.
(6) {CÃES} morder {TODOS}.

Tabela 5-4 : Padrões de sentenças de tamanho um, gerados pela gramática da Tabela 5-1 e pelas restrições da Tabela 5-3.

A partir de então, substitui-se o segundo sujeito (**sujeito$_2$**) pelos pronomes **Ele** ou **Ela**. Para fazer isso, todos os substantivos masculino foram substituídos pelo pronome **Ele** e os substantivos feminino pelo pronome **Ela** na segunda sentença, mas apenas os substantivos que eram o sujeito na segunda sentença. Depois, retirou-se as sentenças que possuiam pronomes fazendo referências erradas (veja exemplo abaixo) e as sentenças cujo preenchedor do pronome não satisfazia a

restrição semântica imposta às sentenças. Por exemplo, considere a sentença a seguir que foi gerada com o procedimento descrito acima.

O garoto viu o homem. Ela mordeu o homem.

Neste exemplo existem 2 erros. Primeiro na sentença antecedente não existe um substantivo feminino que serviria como preenchedor para o pronome **Ela** na segunda sentença. Segundo, se existisse tal preenchedor para o pronome **Ela**, este preenchedor deveria ser **a cadela** para que a restrição ao verbo **mordeu** seja satisfeita. Todas as sentenças que apresentavam um destes erros e as sentenças duplicadas foram retiradas do conjunto de treinamento. Após todos estes procedimentos restaram 1148 sentenças, destes 115 foram retiradas para formar o conjunto de teste e 1033 para formar o conjunto de treinamento.

TODOS	GATOS	HUMANOS	GAROTOS	CÃES
garoto	gato	garoto	garoto	cão
garota	gata	garota	garota	cadela
homem		homem		
mulher		mulher		
cão				
cadela				
gato				
gata				

Tabela 5-5 : Preenchedores das palavras-chave que compõem as sentenças.

5.5 O Léxico e a Representação dos Símbolos de Entrada

Também foi implementado o método FGREP [MII93] (*Forming Global Representation with Extended Backpropagation* - veja seção 4.4.2) para realizar a representação dos símbolos (palavras) de entrada.

Para a implementação de FGREP é necessário um léxico global para armazenar os símbolos (palavras) e suas representações que são determinadas enquanto a rede está aprendendo a realizar a tarefa (veja Figura 5-2). A determinação das representações dos símbolos ocorre durante o treinamento da rede através da extensão do algoritmo de retropropagação (seção 4.4.2).

Os símbolos são apresentados à rede um de cada vez. Eles são substituídos por sua representação armazenada no léxico, formando assim a entrada e a saída desejada para o treinamento da rede. Faz-se a propagação das ativações da entrada até atingir a saída da rede. Nas unidades de saída é calculado o erro que é estendido até à camada de entrada, com o sinal de erro calcula-se a nova representação para o item de entrada (veja Figura 5-2). A nova representação é armazenada no léxico. Repete-se este procedimento para todos os símbolos de entrada até a rede convergir.

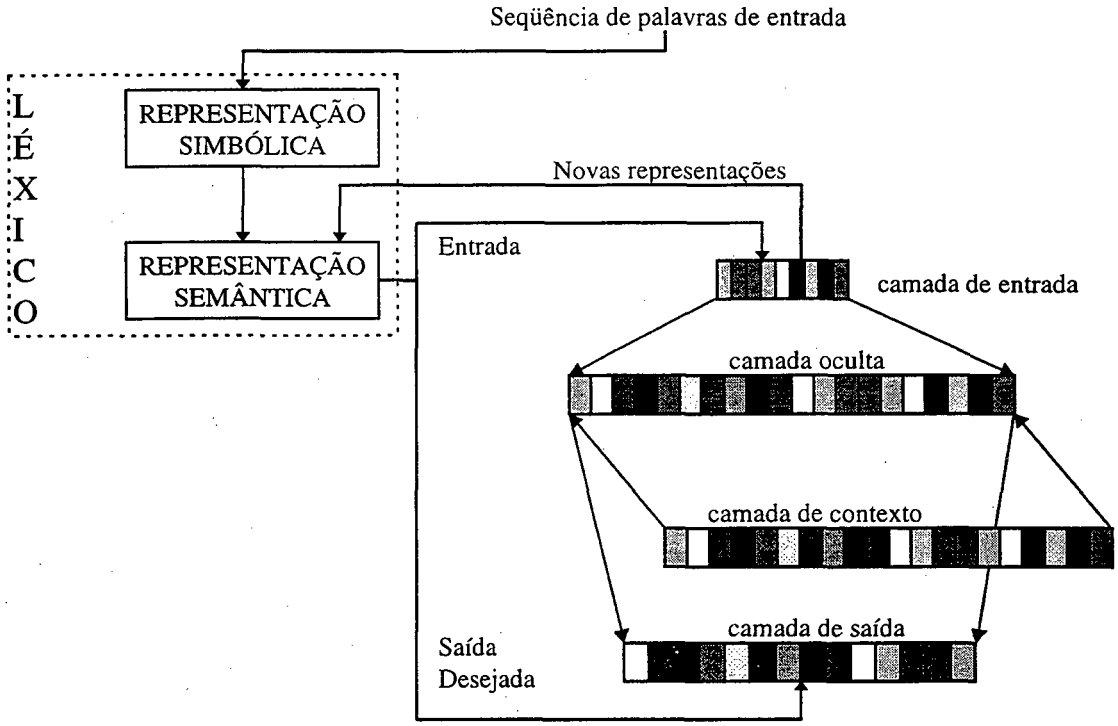

Figura 5-2 : Formação das representações dos símbolos pelo mecanismo FGREP. Os símbolos de entrada são substituídos pelas suas respectivas representações no léxico, formando o padrão de entrada e o padrão alvo. O erro na camada de entrada é utilizado para atualizar as representações, então estas são colocada de volta no léxico.

Os valores das componentes das representações dos símbolos de entrada pertencem ao intervalo [0,1] que são atribuídos aleatoriamente no início do treinamento e modificados pelo algoritmo de retropropagação como parte do aprendizado da tarefa. As representações finais refletem como as palavras são usadas nos exemplos de treinamento, refletindo assim o significado das palavras. Sistemas com representações FGREP geralmente possuem uma forte representação do contexto, resultando em uma boa generalização e robustez contra ruídos e danos [MII95, MII96].

5.6 Treinamento

Cada símbolo é representado por um padrão de tamanho 12, portanto a rede consiste de 12 unidades na camada de entrada, 70 na camada oculta e de contexto e 36 para a camada de saída. As 36 unidades na camada de saída representam os papéis de agente, ação e paciente, cada um com 12 unidades.

O padrão de entrada da rede é composto pela representação de um símbolo, ao passo que o padrão de saída é formado pela união da representação de 3 símbolos correspondendo à agente, ação e paciente. O padrão de saída – padrão alvo – será o mesmo durante a apresentação dos 3 primeiros símbolos de entrada, correspondendo à primeira sentença, depois será trocado pelo padrão que corresponde à segunda sentença, veja Figura 5-3. Por exemplo se a entrada para a rede é a seguinte:

O garoto viu o cão. Ele mordeu o gato.

a padrão alvo será | **garoto viu cão** | para a entrada **O garoto viu o cão** e mudará para | **cão mordeu gato** | para o restante: **Ele mordeu o gato** (Figura 5-3).

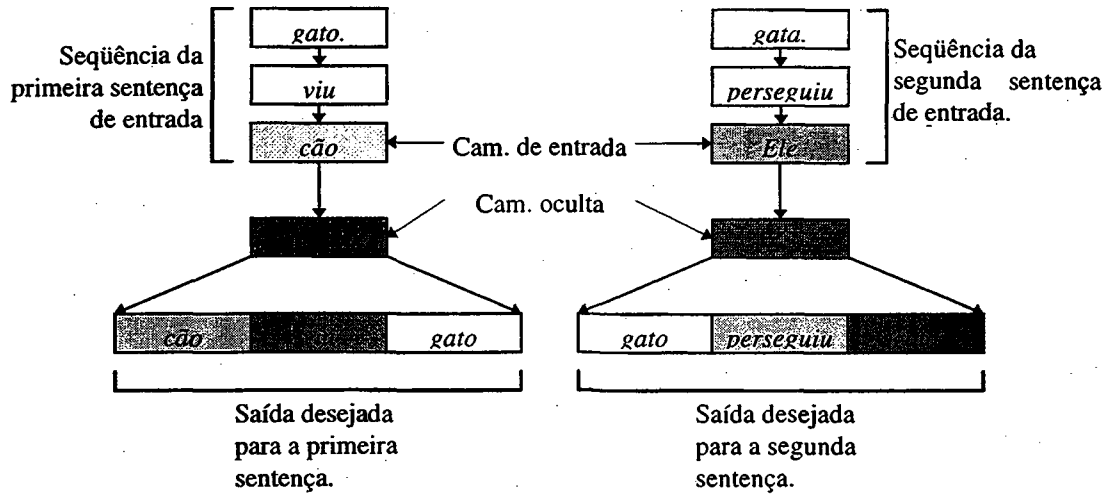

Figura 5-3 : Treinamento da rede recorrente simples. A entrada é seqüencial enquanto a saída é estacionária. Com isso tem-se um mapeamento de uma entrada seqüencial para uma saída estacionária.

O padrão alvo é composto pelo seguimento de texto com o preenchedor para o pronome **ele** satisfazendo as restrições semânticas. Às vezes, mesmo com as restrições da Tabela 5-3 surgem casos de ambigüidade em que existe dois preenchedores do pronome em uma sentença válida. Por exemplo, o texto:

O cão viu o gato. Ele perseguiu a gata.

possui os dois substantivos da sentença precedente satisfazendo as restrições semânticas, assim tanto **O cão** quanto **o gato** podem substituir o pronome na segunda sentença. Neste caso escolhe-se o substantivo mais próximo como sendo o elemento referenciado pelo pronome, que para este exemplo é o NP **o gato**. Portanto, a saída desejada para este exemplo será:

O cão viu o gato. O gato perseguiu a gata.

5.7 Resultados

A representação para cada palavra consistiu de 12 unidades. A rede foi treinada com o algoritmo de retropropagação com uma taxa de aprendizado igual a 0,1 sem utilizar o termo *momentum*. O conjunto de treinamento consistiu de 1033 segmentos de texto que eram apresentados aleatoriamente à rede. A rede atingiu um erro médio de $9{,}44 \times 10^{-4}$ por unidade de saída em 340 épocas. A taxa de aprendizado para a formação das representações foi de 0,001.

A rede foi testada com um conjunto de 115 segmentos de texto nunca apresentados a ela. Para saber a qual representação no léxico um padrão nas partições de saída da rede correspondia, foi escolhido a representação no léxico mais próxima do padrão de saída da partição através de distância euclidiana:

$$d_e = \sqrt{\sum_i \left(x_i - y_i\right)^2}$$

Medido desta maneira a rede obteve um resultado excelente, ela acertou todos os segmentos de texto, ou seja, ela determinou corretamente os preenchedores para os pronomes **ele** e **ela**.

5.7.1 Degradação Progressiva da Memória da Rede

A rede teve um ótimo desempenho para generalizar para segmentos de texto não apresentados a ela durante o treinamento; como vimos acima. Estes segmentos de texto possuíam o mesmo tamanho que os segmentos de treinamento.

À mesma rede foram apresentados segmentos de texto compostos de 3 sentenças. Estes textos foram obtidos através de combinações de 10 segmentos de textos obtidos aleatoriamente no conjunto de teste. Com este procedimento obteve-se um total de 50 segmentos de texto compostos de 3 sentenças. A rede conseguiu acertar 74% destes segmentos, isto é, ele errou a referência para 13 segmentos de texto.

Novamente, dos 50 segmentos obtidos anteriormente escolheu-se 10, combinando-os obteve-se 30 segmentos compostos de 4 sentenças, porém obteve-se apenas 30 segmentos de texto de tamanho 4. Com este conjunto de teste, a rede acertou apenas 70% do total, ou seja, errou 9 segmentos.

Este desempenho diminuindo progressivamente se deve à degradação da memória da camada de contexto para analisar segmentos de texto cada vez maiores – com 3 ou mais sentenças. Para que a rede tenha um melhor desempenho para segmentos de comprimento x , tem-se que treiná-la novamente para segmentos de texto de comprimento x, em suma: a rede não generaliza para uma nova estrutura de segmentos de texto. No próximo capítulo será discutido um modelo que poderá fazer tal generalização.

5.7.2 Capacidade de Previsão

Também foi testado a capacidade de previsão da rede. O teste foi feito da seguinte forma: apresenta-se à rede a primeira sentença e em seguida é apresentado partes da segunda sentença e verifica-se a saída para cada partição. É medida a distância euclidiana do padrão de ativação de cada partição com cada representação das palavras armazenadas no léxico, a menor distância é a saída mais provável de estar correta. (veja Figura 5-4, Figura 5-5 e Figura 5-6).

Pelo primeiro gráfico da Figura 5-4 observa-se que a rede já sabe a quem o pronome **Ele** está se referindo, neste caso o substantivo **cão**, pois a distância para a partição agente é menor para este substantivo. Não é difícil, para a rede, saber quem é o preenchedor correto para o pronome antes de conhecer o resto do contexto, uma vez que na sentença precedente existe apenas um substantivo masculino. O segundo gráfico mostra que a rede também já sabe qual verbo preencherá a partição que corresponde a ação será o verbo **mordeu**, pois pelos dados de treinamento apenas **cão** e **cadela** é que podem **morder** alguém. Pelo último gráfico vê-se que a

rede ainda não tem a menor previsão para o preenchedor da partição paciente, isto porque o verbo **morder** aceita todos os substantivos como objeto direto.

Estes gráficos mostram que a rede aprendeu a regra para a instanciação dos pronomes encontrados na segunda sentença. Também foi feita uma análise[20] para a sentença: **A mulher adorou a garota. Ela ....** Esta sentença tem a peculiaridade de que qualquer um dos substantivos precedentes podem ocorrer no lugar do pronome **Ela**. A distância para a partição agente foi de 0,047 para **garota** e as distâncias para as outros substantivos foram maiores que 1,23. Isto porque **garota** e **mulher** pertencem a mesma categoria (**humanos**) e a regra ensinada a rede é que neste caso de ambigüidade opta-se pelo segundo substantivo.

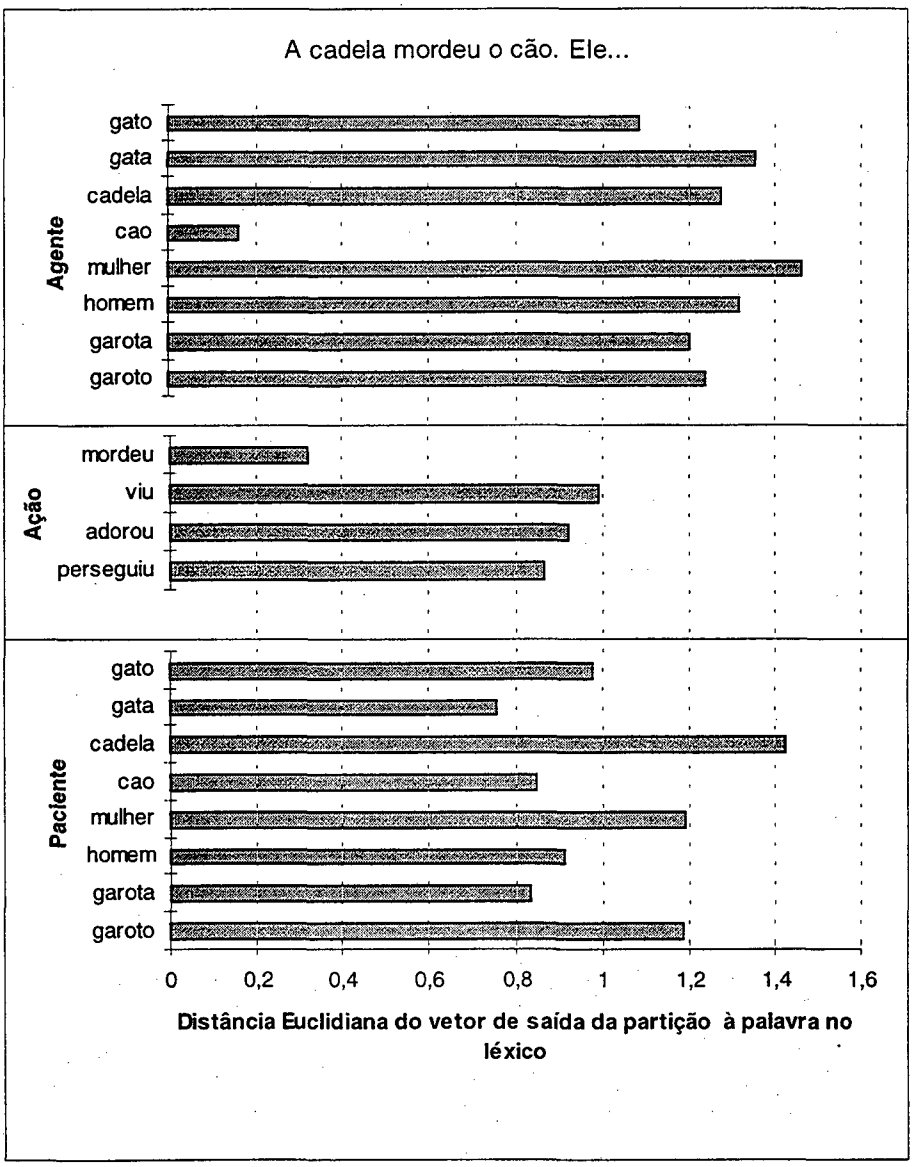

Figura 5-4 : Os gráficos mostram a distância euclidiana entre cada padrão de ativação nas partições de saída da rede e as representações das palavras no léxico quando a última entrada na rede é o pronome **Ele**.

A Figura 5-5 apresenta os gráficos após o verbo já ter sido apresentado à rede. Pelo primeiro gráfico fica mais nítido que o substantivo **cão** é o agente da segunda sentença e a confirmação que o verbo é **mordeu**. Mas ainda a rede não sabe quem é o paciente da segunda sentença.

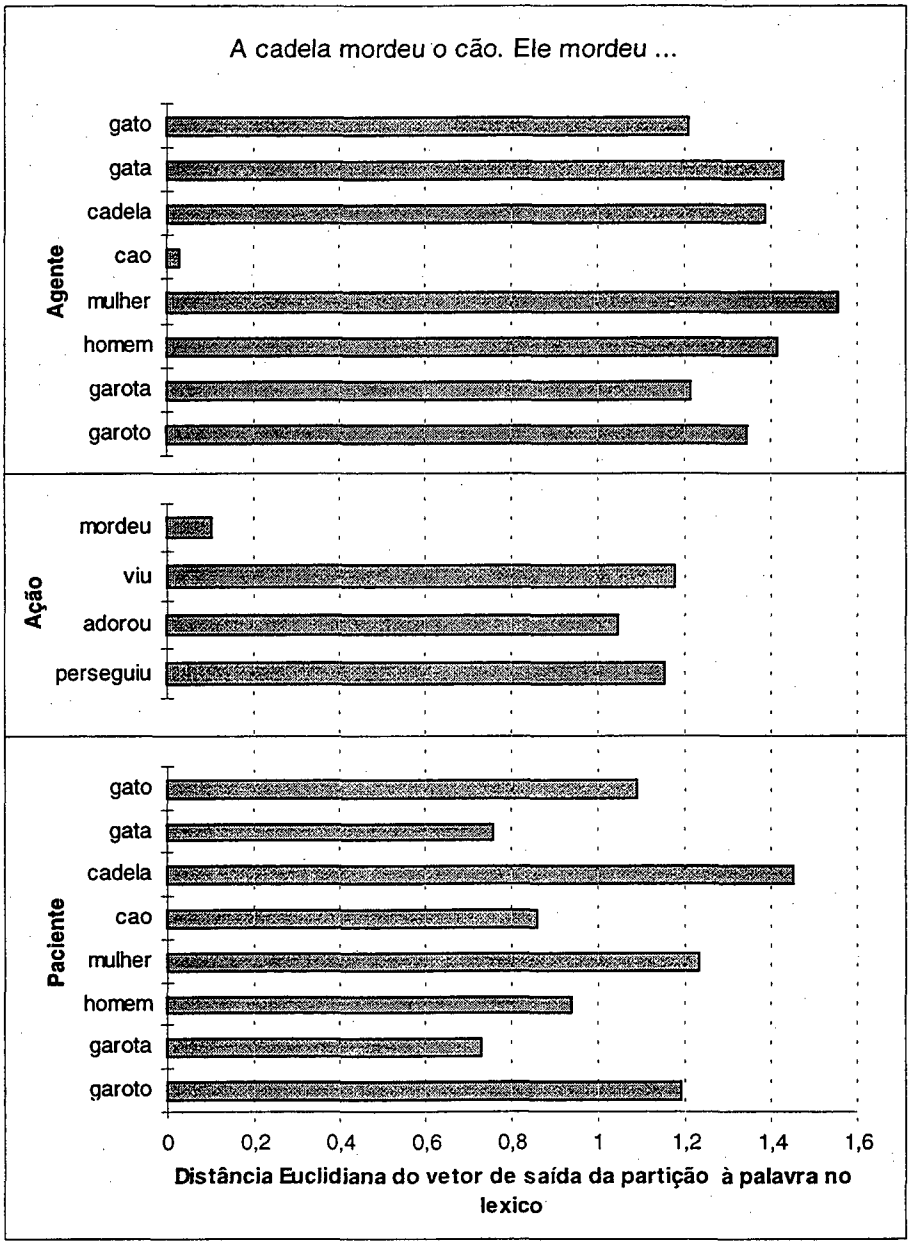

Figura 5-5 : Mesmo gráfico da figura anterior, porém com a última palavra apresentada à rede sendo o verbo **mordeu**.

Na Figura 5-6 todos os componentes da sentença já foram apresentados à rede. Esta figura ilustra as distâncias euclidianas para todas as partições no final da análise de um segmento de texto. Pode-se perceber claramente que a saída da rede é:

A cadela mordeu o cão. O cão mordeu a garota.

[20] Não foram feito gráficos, para não ficar repetitivo.

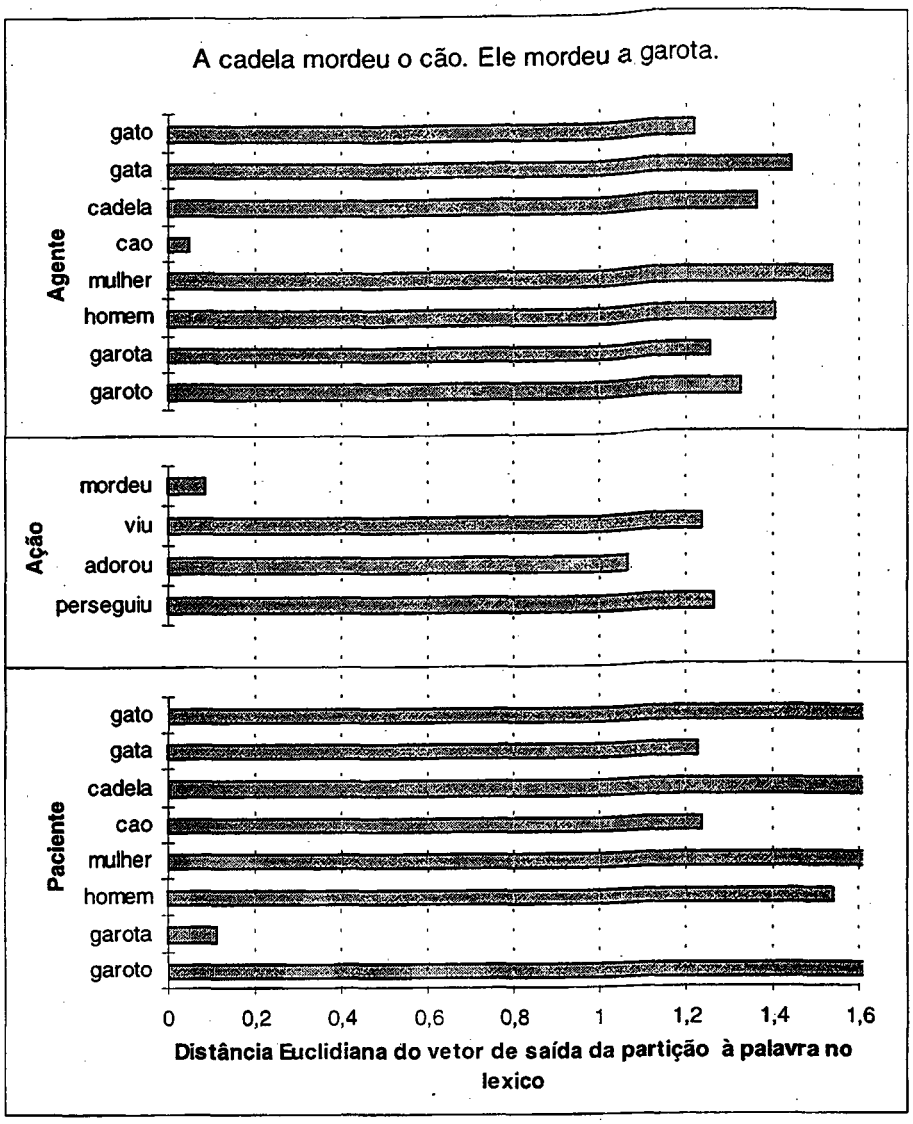

Figura 5-6 : Mesmo gráfico; agora com todo o texto já apresentado à rede.

Pelos resultados descritos acima observa-se que a SRN resolveu eficientemente a referência anafórica para segmentos de textos compostos de duas sentenças e teve um desempenho razoável na resolução do problema para segmentos de texto de 3 e 4 sentenças. No próximo capítulo será descrita uma abordagem que possibilitará a rede resolver todos os segmentos de textos de 3 e 4 sentenças corretamente.

6. Um *Parser* Conexionista Modular para Resolução de Anáforas

6.1 Introdução

Como vimos no capítulo anterior, uma rede recorrente simples resolve eficientemente a referência anafórica para as estruturas de segmentos de textos para os quais ela foi treinada. Mas, ao submeter tal rede a segmentos de textos de estrutura diferente (com mais sentenças) seu desempenho diminui com o aumento do número de sentenças que compõe o segmento de texto.

Neste capítulo será descrito um novo modelo para analisar segmentos de textos visando resolver o mesmo problema como discutido no capítulo anterior. O modelo é composto de dois módulos – duas redes neurais artificiais – e foi chamado de *Parser* Conexionista para Resolução de Anáforas Pronominais (PCRAP) . Tais redes são treinadas simultaneamente utilizando-se do mesmo conjunto de treinamento. A novidade nesta abordagem é que será introduzido uma segunda rede para auxiliar e controlar a tarefa da rede recorrente simples de modo que esta execute melhor sua tarefa. O objetivo deste modelo é fazer com que a rede possa generalizar eficientemente para segmentos de textos com um número arbitrário de sentenças.

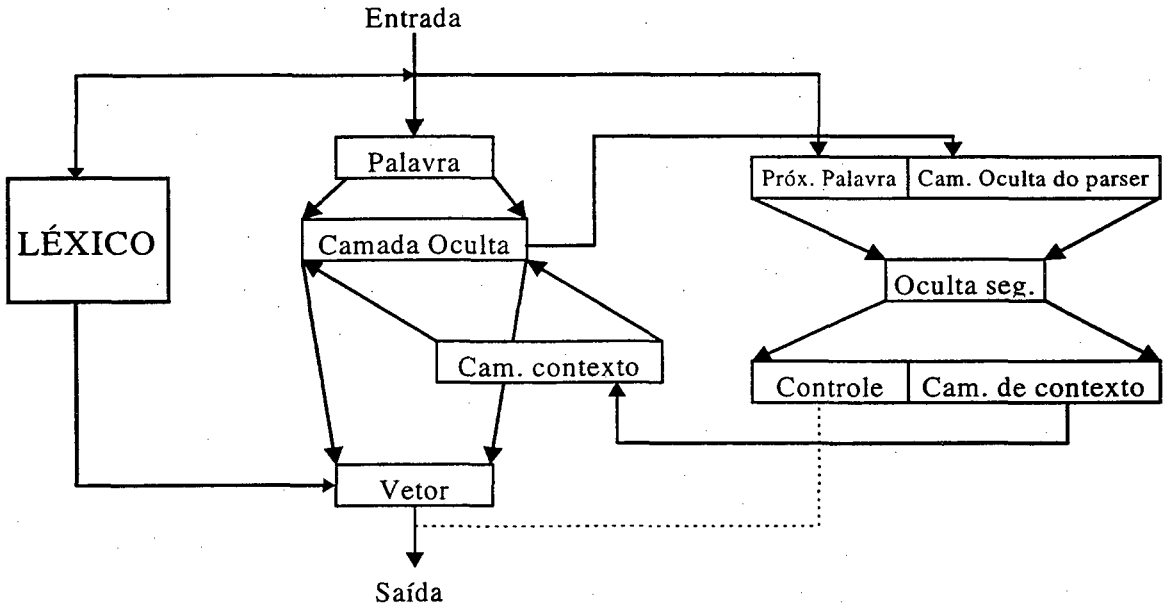

Figura 6-1 : Arquitetura de PCRAP. O *Parser* tem, como entrada, uma seqüência de palavras e, como saída, a representação em papéis de caso das constituintes da seqüência de entrada. O segmentador controla a execução do modelo e "particiona" a seqüência de entrada.

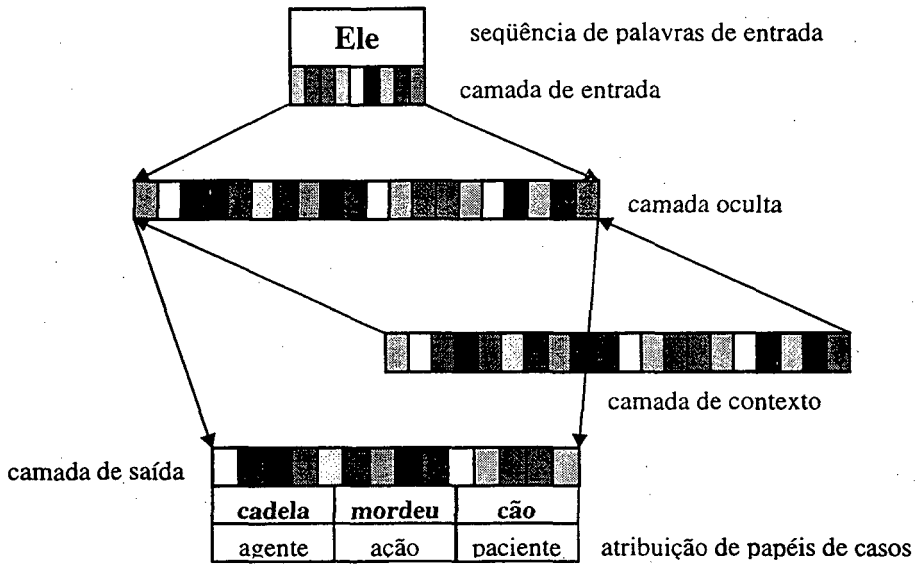

Figura 6-2 : A rede *Parser*. A rede mostra a saída para o exemplo: **A cadela mordeu o cão. Ele perseguiu o gato**. Neste momento os papéis de caso da primeira sentença já foram determinados.

6.2 Arquitetura do Modelo

Uma visão geral da arquitetura é mostrada na Figura 6-1. O modelo recebe uma seqüência de representações de palavras como sua entrada, e para cada sentença no segmento de texto é formado uma representação de saída indicando a atribuição das constituintes da sentenças em seus papéis de casos. A representações de papéis de casos constituem o resultado final da análise de uma sentença. O modelo é constituído por duas redes neurais artificiais: O *Parser* (rede recorrente simples) e o segmentador (rede direta multicamada), cada qual com uma função específica que serão discutidas separadamente nas próximas seções. Em PCRAP também existe um léxico nos mesmos moldes da rede descrita no capítulo anterior.

6.2.1 O *Parser*

O *Parser* (Figura 6-2) receberá uma seqüência de representações das palavras como sua entrada, estas palavras são apresentadas uma de cada vez à rede. A saída da rede será uma representação de papéis de caso das constituintes da sentença de entrada. Como no capítulo precedente as representações são determinadas pelo método FGREP.

Considerando o exemplo **A cadela mordeu o cão. Ele perseguiu o gato** como uma seqüência de entrada para o *Parser*, este produzirá a seguinte saída:

| cadela mordeu cão |
| cão perseguiu gato |

A Figura 6-2 exibe o *parser* quando este processou a primeira sentença de entrada, e está tendo como entrada o pronome **Ele**. O *parser* é idêntico à rede descrita no capítulo anterior.

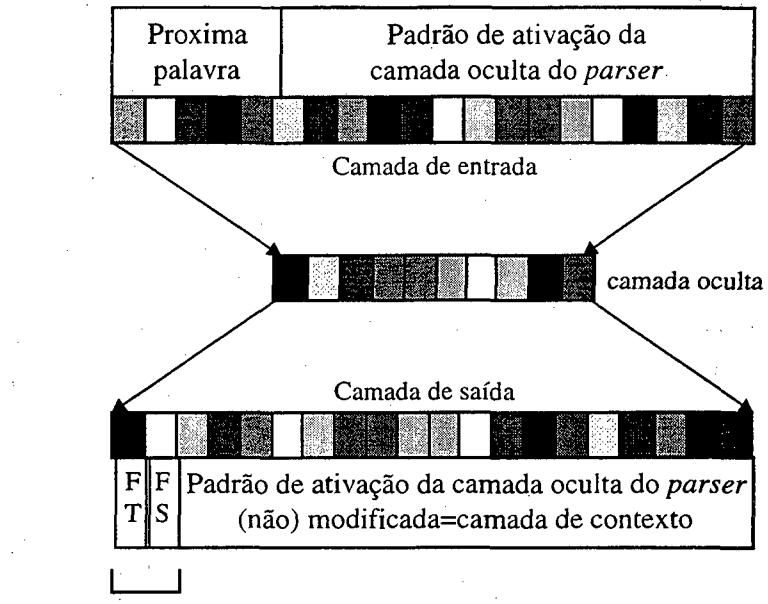

Figura 6-3 : A rede segmentador. A entrada é a próxima palavra na seqüência mais o padrão de ativação da camada oculta do *parser*. A saída será a camada oculta do *parser* modificada ou não, dependendo da próxima palavra. FS e FT significam, respectivamente, fim de texto e fim de sentença.

6.2.2 Segmentador

O segmentador tem a função de quebrar a memória da seqüência do texto na camada de contexto do *parser* de modo que este não armazene informações irrelevantes, somente as informações necessárias para resolver a referência na próxima sentença.

Com o segmentador, o próprio modelo detém o controle durante a fase de execução; e ainda, basta treiná-lo com apenas segmentos de textos compostos de apenas duas sentenças que o modelo generalizará para segmentos de textos de comprimento arbitrário, ou seja, com três ou mais sentenças.

A entrada para o segmentador será o padrão de ativação da camada oculta do *parser* mais o próximo elemento na seqüência de entrada. Quando a entrada para o *parser* for a palavra (representação desta) que corresponde ao paciente da sentença precedente, a entrada para o segmentador será a próxima palavra na seqüência de entrada (neste caso corresponderá a um dos pronomes: **Ele** ou **Ela**) mais o padrão de ativação da camada oculta do *parser*. Com esta entrada, a saída do segmentador será um padrão de ativação similar ao da camada oculta do *parser* quando este teve o preenchedor do pronome na sentença atual como a primeira palavra de entrada em algum segmento de texto precedente no conjunto de treinamento. Esta saída modificada será colocada na camada de contexto do *parser*, apagando assim as informações desnecessárias armazenada na camada de contexto. É como se o *parser* "enxergasse" apenas uma sentença de cada vez.

Quando o próximo elemento da seqüência (entrada atual para o segmentador) não for um pronome nem o "." (ponto final), a saída do segmentador corresponderá ao padrão atual da

camada oculta do *parser*, desse modo não haverá modificações na camada de contexto, ela será uma cópia da camada oculta do *parser*. Assim sendo, o *parser* funcionará como uma rede recorrente simples padrão, como descrito por Elman [ELM90].

O controle é realizado pelo segmentador através de duas unidades adicionais na saída, chamadas fim_sentença e fim_texto; ambas serão 0 ou 1. A unidade fim_sentença valerá 1 se o próximo elemento na seqüência for um pronome ou um "." (ponto), indicando que uma sentença foi analisada, isto é, já foi determinada a saída; fim_sentença será 0 caso contrário. A unidade fim_texto valerá 1 se o próximo elemento na entrada for um ponto indicando o fim do texto, neste caso a saída do segmentador valerá: 1 para as unidades de controle e 0 para as demais unidades ("limpa" a camada de contexto).

Resumindo, as tarefas do segmentador são:

a) Detectar o pronome quando este for a entrada para o segmentador e modificar o padrão de ativação da camada de contexto do *parser* (fim_sentença igual a 1 e fim_texto igual a 0);
b) Detectar o fim da análise de uma sentença (fim_sentença igual a 1);
c) Reconhecer o fim do texto, indicados por um "." (ponto) e limpar a camada de contexto que consiste em atribuir 0 (zero) para as unidades da camada oculta do *parser* preparando o para analisar o próximo segmento de texto (fim_texto e fim_sentença são iguais a 1).

6.3 Treinamento e Resultados

O treinamento de PCRAP foi realizado pelo mesmo conjunto de segmentos de texto utilizado para treinar a rede descrita no capítulo anterior. As duas redes foram treinadas simultaneamente, com o segmentador pegando os dados de treinamento na camada oculta do *parser*. Porém pode-se treiná-las separadamente.

As representações das palavras foram constituídas de 12 unidades. A camada oculta do *parser* possuía 75 unidades, a do segmentador 50 unidades. Todas as redes foram treinadas com o algoritmo de retropropagação com taxa de aprendizado igual a 0,1 e com termo *momentum* igual a 0 (zero). Ambas redes desenvolveram representações das palavras em suas respectivas camadas de entrada com uma taxa de aprendizado igual a 0,01.

O modelo foi treinado com 400 épocas. O erro médio por unidade de saída foi de $9{,}85 \times 10^{-4}$ para o *parser;* $1{,}93 \times 10^{-4}$ para o segmentador, segundo Miikkulainen [MII93], um nível de erro de 0,020 geralmente resulta em um desempenho aceitável em sistemas baseados em partições como este. O treinamento do modelo poderia ter sido interrompido antes de chegar nas 400 épocas.

Após o treinamento o modelo foi submetido aos mesmos conjuntos de teste de tamanho 1, 3 e 4 como descrito no capítulo anterior. O modelo acertou todos os segmentos de texto de todos os conjuntos de teste, diferentemente da rede do capítulo anterior que errou alguns textos de tamanho 3 e 4. Considerando que nestes testes o controle da execução era inteiramente realizado pelo segmentador.

Será comprovado, através de gráficos que realmente o segmentador aprendeu sua tarefa. Os gráficos a seguir, foram feitos utilizando-se do mesmo procedimento anterior, isto é, o

modelo terá como entrada partes do segmento de texto e será calculada a distância euclidiana entre a saída do segmentador e a saída desejada. O texto que aparece em cima nas figuras correspondem às entradas para o *parser* até aquele momento. Lembrando que a saída do segmentador é o padrão de ativação da unidade oculta do *parser* modificada ou não, e as duas saídas de controle.

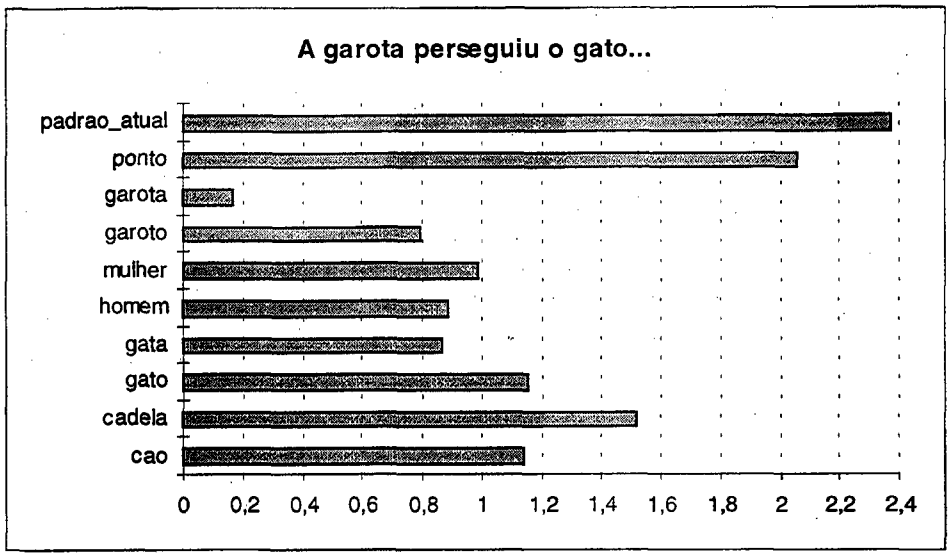

Figura 6-4 : A distância euclidiana mostra que a camada de contexto do *parser* será modificada para corresponder ao padrão de ativação de uma sentença que está começando com **A garota...** Neste momento a entrada para o segmentador é o pronome **Ela** mais o padrão ativação atual da camada oculta do *parser*.

Os 4 gráficos (Figura 6-4, Figura 6-5, Figura 6-6 e Figura 6-7) mostram as distâncias euclidianas entre a saída do segmentador e o padrão de ativação na camada oculta do *parser* quando uma das palavras do eixo *y* inicia um segmento de texto.

Pela Figura 6-4 pode-se perceber que o segmentador neste momento vai modificar o padrão de ativação da camada de contexto do *parser* de modo que este "pense" que está começando a analisar uma nova sentença. A entrada para o segmentador é o pronome **Ela** mais o padrão de ativação da camada oculta do *parser*. Como já foi dito, o segmentador faz com que o *parser* "enxergue" apenas uma transição de uma sentença inicial com uma outra a seguir que possui um pronome como sujeito.

Deste modo percebe-se que se houvesse uma outra sentença – uma terceira – a saída do segmentador corresponderia ao padrão de ativação na camada oculta do *parser* de quando o sujeito da segunda sentença foi a entrada inicial para o *parser*.

Na Figura 6-5 pode-se ver claramente que a camada de contexto do *parser* não será modificada, pois a saída do segmentador neste instante está muito próxima ao padrão de ativação de sua própria entrada, que por sua vez corresponde ao padrão de ativação da camada oculta do *parser* (padrão atual). A entrada para o segmentador é o verbo **adorou**.

A Figura 6-6 possui os padrões equivalentes aos da figura anterior. Nesta também o segmentador não modificará a memória da seqüência até o momento, pois não é final de sentença.

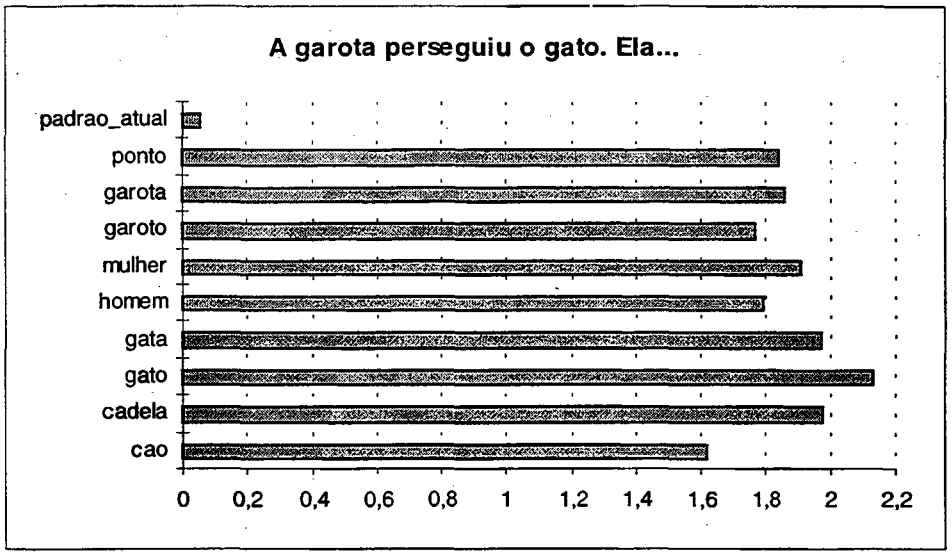

Figura 6-5 : Neste instante, a entrada para o segmentador é o verbo **adorou**. Observe que a rede segmentador tem como saída o padrão atual não modificado.

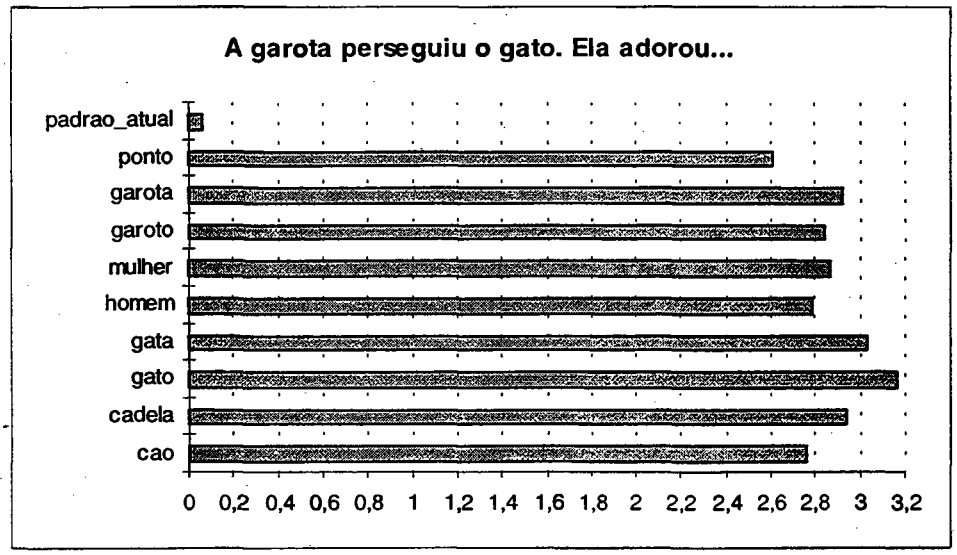

Figura 6-6 : Neste instante, a entrada para o segmentador é o substantivo **cão**, por isso o padrão de ativação na camada de contexto do *parser* não será modificado.

A sentença chegou ao final, e as distâncias exibidas na Figura 6-7 mostra que o segmentador detectou com muita eficiência o fim de um segmento de texto. A saída do segmentador será 0 (zero) que corresponde a "limpar" a memória na camada de contexto do *parser*, preparando-o para um novo segmento de texto.

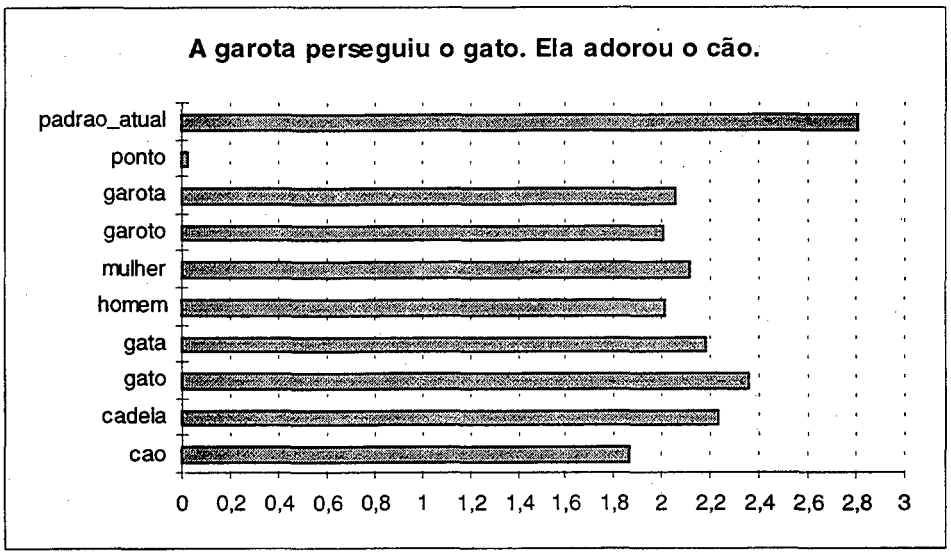

Figura 6-7 : Fim de sentença, neste momento a saída do segmentador corresponde ao ponto, portanto a camada de contexto será "zerada", preparando o *parser* para analisar mais uma sentença.

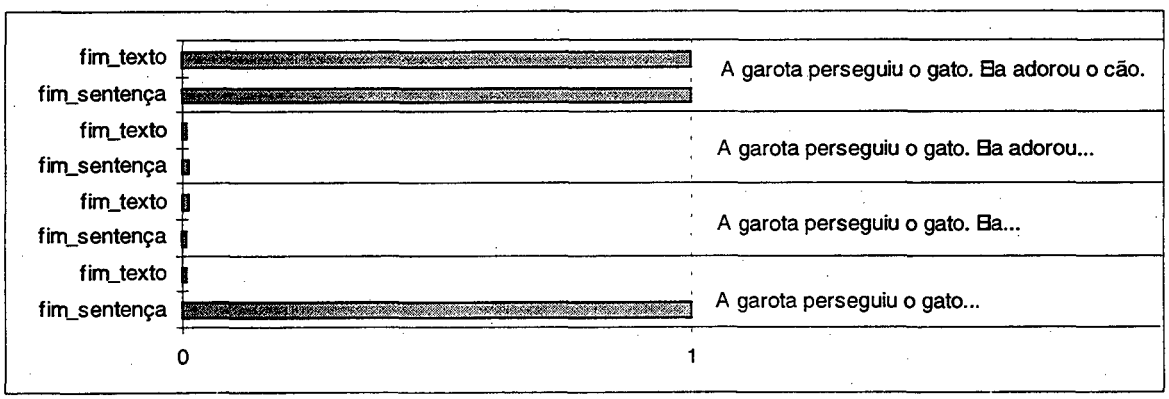

Figura 6-8 : Saídas de controle do segmentador. A figura mostra a saída de controle para cada palavra da segunda sentença.

A Figura 6-8 acima mostra a saída de controle. Observe que o segmentador consegue uma saída muito próxima de 0 (zero) ou 1 (um) e todas estão corretas. A unidade fim_texto é 1 quando a próxima palavra na seqüência de entrada é um ponto (final de segmento de texto) e zero nos demais casos. A unidade fim_sentença é 1 quando a próxima palavra é um pronome ou um ponto, e zero nos demais casos.

7. Considerações Finais

Neste trabalho foram apresentadas duas abordagens conexionistas para a resolução de anáforas pronominais com os pronomes **ele** e **ela** em segmentos de textos de duas sentenças.

Na primeira abordagem foi utilizada uma rede recorrente simples (SRN) treinada com um subconjunto de todos os possíveis segmentos de texto gerados artificialmente. A rede resolveu eficientemente todos os exemplos inéditos apresentados à ela com a mesma estrutura com a qual ela foi treinada.

Na segunda abordagem foi acrescentado à SRN uma rede direta multicamada que permitiu a SRN generalizar para sentenças de tamanho arbitrário.

Mesmo sendo implementado para analisar textos composto com mais de duas sentenças, o modelo foi testado com poucos segmentos de textos de tamanho maiores que 2 (50 e 30 para segmentos de textos compostos de 3 e 4 sentenças, respectivamente). Uma maneira mais eficiente para analisar o modelo, seria gerar todos os segmentos de textos de tamanho maior que dois e menor que um número qualquer, digamos 5, e destes escolher aleatoriamente um conjunto de segmentos de texto de cada para compor os conjuntos de teste. Deste modo faz-se uma comparação mais eficiente. Infelizmente , isto não foi possível, devido ao grau de dificuldade encontrado para gerar tais textos e será deixado como proposta para um trabalho futuro.

Foram feitos vários testes durante o treinamento do modelo PCRAP, como: variar taxa de aprendizado para as representações das palavras, número de neurônios nas camadas ocultas das redes, etc. Para taxa de aprendizado das representações das palavras percebeu-se que quanto maior, mais oscilações no erro ocorrem durante o treinamento e as representações das palavras atingem valores próximos ou iguais a 1 ou 0 (zero).

O modelo PCRAP também foi treinado com apenas 23 neurônios na camada oculta do *parser*, mas apesar deste ter conseguido um erro na ordem de 10^{-3} para o *parser* e o segmentador, ele não acertou nenhum exemplo no conjunto de teste. Isto aconteceu porque não haviam unidades de contexto suficiente para representar a memória da seqüência das palavras de entrada. Depois, aumentando-se para 75 foram obtidos os resultados descritos anteriormente.

As sentenças utilizadas são artificiais e pouco usuais. O modelo PCRAP poderá ser treinado com textos mais complexos. Por exemplo, as sentenças podem ter mais papéis de casos (recipiente, modificador, etc.), o vocabulário pode ser estendido e pode-se utilizar pronomes oblíquos, isto é, a ocorrência de um pronome não será apenas como sujeito na sentença, mas também como objeto. Levando estas sugestões em consideração, o texto para o treinamento do modelo pode ser composto por pequenos roteiros (veja seção 3.10) como o do exemplo na Tabela 7-1.

PESSOA foi ao restaurante. O garçom a/o acompanhou até a mesa. Ele/Ela pediu COMIDA.

Tabela 7-1 : Tabela com um roteiro simples que possui dois pronomes (um pronome pessoal e outro oblíquo).

Na tabela acima, as palavras em caixa alta são palavras chaves que terão seus preenchedores correspondentes, e assim formarão várias pequenas historias. Com este e outros roteiros obtém-se uma quantidade considerável para treinar o modelo. Enfim, o modelo implementado pode ser estendido para a resolução de problemas mais complexos relacionados à resolução anafórica.

Referências Bibliográficas

[ALL87] ALLEN, J. **Natural Language Understanding**. Melon Park: Benjamin/Cummings Publishing, 1987.

[ARB89] ARBIB, M. A. **The Metaphorical Brain 2**: Neural Network and Beyond. New York: Wesley, 1989.

[BAF81] BARR, A.; Feigenbaum, E. A. **The Handbook of Artificial Intelligence**. Massachusetts: Addison-Wesley Publishing Company, Inc., 1981. v.1.

[BAR96a] BARRETO, J. M. **Conexionismo e a Resolução de Problemas**. Florianópolis, 1996. Trabalho para concurso público para professor titular – Departamento de Informática e Estatística, Universidade Federal de Santa Catarina.

[BAR96b] BARROS, F. A.; Robin, J. **Processamento de Linguagem Natural**. XV JAI/SBC'96, Recife, PE, 1996.

[BIT96] BITTENCOURT, G. **Inteligência Artificial - Ferramentas e teorias**. 10ª Escola de Computação, Unicamp, Campinas, SP, 1996.

[CAG87] CARPENTER, G. A.; Grossberg, S. **A massively parallel architecture for a self-organizing neural pattern recognition machine**. *Computer Vision, Graphics, and Image Processing* 37, 54-115, 1987.

[DYE91] DYER, M.G. **Symbolic neuroengineering for natural language processing: a multilevel research approach**. In: Barnden, J.A.; Pollack, J.B. (Ed.) Advances in connectionist and neural computation theory, Norwood, NO: Ablex Publishing, p.32-86, 1991. v.I:High level connectionist Models.

[ELM90] ELMAN, J. L. **Finding structure in time**. *Cognitive Science*, 14:179-211, 1990.

[ELM91a] _____. **Distributed representations, simple recurrent networks, and grammatical structure**. Machine Learning, n.7 195-225. 1991.

[ELM91b] _____. **Incremental learning, or The importance of starting small**. In: Proceedings of the 13th Annual Conference of the Cognitive Science Society, 443-448. Hillsdale, NJ:Erlbaum. 1991.

[FEL93] FELDMAN, J. A. **Structured connectionst models and language-learning**. Artificial Intelligence Review, 7(5):301-312, 1993.

[FIL68] FILMORE, C. J. **The case for case**. In: Bach, E.; Harms, R. T., (Ed.), Universals in Linguistic theory, p.0-88. New York: holt, Rinehart and Winston, 1968.

[FOP88] FODOR, J. A.; Pylyshyn, Z. W. **Connectionism and cognitive architecture: A critical analysis**. Cognition, 28:3-71, 1988.

[FUK75] FUKUSHIMA K. **Neocognitron: A self-organizing multilayered neural network**. *Biological Cybernetics* 20, 121-136, 1975.

[FUK88] _____. **Neocognitron: A hierarchical neural network capable of visual pattern recognition**. *Neural Networks* 1, 119-130, 1988.

[GHL95] GILES, C. L.; Horne, B.G.; Lin, T. **Learning a class of large state machines with a recurrent neural networks**. Neural Network, v.8, n.9, p.1359. 1995.

[GMC92] GILES, C. L.; Miller, C.; Chen, D.; Chen, H.; Sun, G.; Lee Y. **Learning and extracting finite-state automata with second-order recurrent neural networks**. Neural Computation, v.4, n.3, p.393-405. 1992.

[GRO76] GROSSBERG, S. **Adaptive pattern classification and universal recording I e II**. *Biological Cybernetics* 23, 121-134, 187-202, 1976.

[HAY94] HAYKIN, S. **Neural Networks: A Comprehensive Fundation**. IEEE Press, 1994.

[HEB49] HEBB, D. O. **The Organization of Behavior: A Neuropsychological Theory**. New York: Wiley, 1949.

[HEN89] HENDLER, J. A. **Marker passing over microfeatures: towards a hybrid symbolic connectionist model**. Cognitive Science 13, p.79-106. 1989.

[HIN81] HINTON, G. E. **Implementing semantic networks in parallel hardware**. In: Hinton, G. E.; Anderson, J. A., (Ed.), Parallel models of Associative Memory. Lawrence Erbaum, 1981.

[HIN88] _____. **Representing Part-Whole hierarchies in connectionist networks**. In: Proceedings of the Tenth Anual Conference of the Cognitive Science Society. Montreal, 48-54, 1988.

[HIN91] _____ (Ed.). **Connectionist Symbol Processing**. MIT Press/Elsevier, 1991.

[HPK95] HONKELA, T.; Pulkki, V.; Kohonen, T. **Contextual Relations of words in Grimm tales, Analyzed by Self-Organizing Map**. Proceedings of International conference on Artificial Intelligence, ICANN-95, Paris, p.188-192. 1995.

[JAC92] JACQUEMIN C. **Activition Diffusion: A connectionist network for robust parsing**. In: Newmann, B., (Ed.), Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI92), 183-187, Chichester, UK, 1992.

[JAM95] JAMES, D. L.; Miikkulainen, R. **SARDNET: A Self-Organizing Feature Map for Sequence**. In: Tesauro, G.; Touretzky, D. S., Leen, T. K. (Ed.). Advances in Neural Processing Systems 7. 1995.

[JAW90] JAIN, A. N.; Waibel, A. H. **Incremental Parsing by Modular recurrent connectionist networks**. In: Touretzky, D. S., (Ed.), Advances in Neural Information Processes, volume 2, 364-371. Morgan Kaufman, 1990.

[JMM96] JAIN, A. K.; Mao, J.; Mobiuddin, K. M. **Artificial Neural Networks: A Tutorial.** Computer, Vol 29, n° 3, 31-44, March 1996.

[JOR86] JORDAN, M. I. **Attractor dynamics and parallelism in a connectionist sequential machine**. In: *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, 531-546. Hillsdale, NJ: Erlbaum, 1986.

[KAM90] KAMIMURA, R. **Aplication of temporal supervised learning algorithm to generation of natural language**. In: International Joint Conference on Neural Networks (IJCNN90), volume 1, 201-208, San Diego, IEEE, 1990.

[KOH88] KOHONEN, T. **Self-Organization and Associative Memory**, 3ª edição, New York, Springer-Verlag, 1989.

[KOH90] _____. **The Self–Organizing Map**. Proceedings of IEEE, Special Issue on Artificial Neural Network, v.78, n.9, p.1464-1480. 1990.

[KSV96] KARTALOPOKLOS, S. V. **Understandings Neural Networks and Fuzzy Logic**. IEEE Press, 1996.

[LEE91] LEE, G. **Distributed Semantic Representations for Goal/Plan Analysis of Narratives in a Connectionist Architecture**. Ph.D thesis, Computer Science Department, University of California, Los Angeles. Technical Report UCLA-AI-91-03. 1991.

[McK86] McLELLAND, J. L.; Kawamoto, A. H. **Mechanism of sentence processing: Assigning roles to constituents**. In: McLELLAND, J. L.; Rumelhart, D. E., (Ed.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 2: Psychological nd Biological Models, 272-325. Cambridge, MA:MIT Press, 1986.

[McP43] McCULLOCH, W. S.; Pitts, W. H. **A logical calculus of ideas immanent in nervous activity**. *Bulletin of Mathematical Biophysics*, 5:115-133, 1943.

[MID91a] MIIKKULAINEN, R.; Dyer, M. G. **Encoding input/output representations in connectionist cognitive systems**. In: Touretzky, D. S.; Hinton, G. E.; Sejnowski, T. J., (Ed.), *Proceedings of the 1988 connectionist Model Summer School*, 347-356. San Mateo, CA: Morgan Kaufmann, 1991.

[MID91b] _____. **Natural language porcessing with modular neural networks and distributed lexicon**. *Cognitive Science*, 15:343-399, 1991.

[MII90] MIIKKULAINEN, R. **A PDP architecture for processing sentences with relative clauses**. In: Karlgren, H., (Ed.), *Proceedings of the 13th International Conference on Computational Linguistics*, 201-206. Helsinki, Finland: Yliopostopaino, 1990.

[MII93] _____. **Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory**. Cambridge, MA: MIT Press, 1993.

[MII95] _____. **Subsymbolic Parsing of Embedded Structures**. In: Sun R.; Bookman L. A., (Ed.), *Computational architectures integrating neural and Symbolic Processes: A Perspective on the state of the art*, 153-186. Kluwer Academic Publisher, 1995.

[MII96] _____. **Subsymbolic Case-Role Analyses of Sentences with Embedded Clauses**. *Cognitive Science*, 20:47-73, 1996.

[MIP69] MINSKY, M. L.; Papert, S. A. **Perceptrons: An Introduction to Computational Geometry**. MIT Press, 1969.

[OMG96] OMLIN, C. W.; Giles, C. L. **Constructing deterministic finite-state automata in recurrent neural networks**. Technical Report CS-TR-3460. University of Maryland. 1996

[PAR85] PARKER, D. B. **Learning-logic: Casting the cortex of the humans brain in silicon**. Technical Report TR-47. Center of Computational Research in Economics and Management Science, MIT, Cambridge, MA, 1985.

[PLA94] PLATE, T. A. **Distributed Representation and nested compositional structure**. tese de PhD. Cumputer Science Department, University of Toronto, 1994.

[POL89] POLLACK, J.B. **Implications of Recursive Distributed Representations**. In: dvances in Neural Information Processing System I, San Mateo: Morgan Kaufmann, 527-536. 1989.

[POL90] POLLACK, J. B. **Recursive distributed representations**. *Artificial Intelligence*, 46:77-105, 1990.

[RAB95] RABUSKE, R. A. **Inteligência Artificial**. Florianópolis: Editora da UFSC, 1995.

[RHW86] RUMELHART, D. E.; Hinton, G. E.; Williams, R. J. **Learning internal representations by error propagation**. In: Rumelhart, D. E.; McClelland, J. L., (Ed.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 1: Foundations, 318-362. Cambridge, MA:MIT Press, 1986.

[RIK89] RITTER, H.; Kohonen, T. **Self-Organizing Semantic Maps**. Biological Cybernetics, n.61, p.241-254, 1989.

[ROI96] ROISEMBERG, M. **Emergência da Inteligência em agentes autônomos através de modelos inspirados na natureza**. Florianópolis, 1996. Tese de qualificação, Departamento de engenharia elétrica - GPEB. Universidade Federal de Santa Catarina.

[ROS58] ROSENBLATT, F. **The Perceptron: A probabilistic model for information storage and organization in the Brain**. Psychological rewiew 65, 386-408, 1958.

[ROS62] _____. **Principles of Neurodynamics**. Washington, DC: Spartan Books.

[RUM86] RUMELHART, D. E.; McLelland, J. L. **On Learning of the past tenses of English verbs**. In: Mclelland, J. L.; Rumelhart, D. E., (Ed.), Parallel Distributed Processing, vol 2, 216-271. MIT Press, Cambridge, MA, 1986.

[RUZ85] RUMELHART, D. E.; Zipser, D. **Feature discovery by competitive learning**. *Cognitive Science* 9, 75-112, 1985.

[SAV88] SAVADOVSKY, P. A. **Construção de interpretadores para linguagem natural**. Ed. EBAI-88. São Paulo: EBAI, 1988.

[SCA77] SCHANK, R.C.; Abelson, R. P. **Scripts, plans, goals, and Understanding: An Inquiry into Human Knowledge Structures**. Hillsdale, NJ: Erlbaum.

[SCH91] SCHOLTES, J. C. **Kohonen features maps in Natural Language Processing**. Technical Report CL-91-01, Institute for Language, Logic and Information, University of Amsterdam. 1991.

[SCS82] SMALL, S. L.; Cotrell, G.; Shastri, L. **Towards connectionist parsing**. In: Proceedings of the National Conference on Artificial Intelligence: AAAI, 1982.

[SEH85] SELMAN, B.; Hirst, G. **A rule-based connectionist parsing system**. In: Proceedings of the Seventh Annual Conference of the Cognitive Science Society, 212-221, Erlbaum, 1985.

[SER87] SEJNOWSKI, T. J.; Rosenberg, C. R. **Parallel networks that learn to pronounce English text**. *Complex Systems*, 1:145-168, 1987.

[SHR92] SHARKEY, N. E.; Reily, R. G. **Connectionist Natural Language Processing**. In: _____ (Ed.). Connectionist approaches to Natural Language Processing. P.1-12. Hove:Lawrence Erlbaum associates. 1992

[SJT88] SCHWARTZ, J. T. **The new connectionism - Developing relationships between neuroscience and artificial intelligence**. In: S. R. Graubard, (Ed.), The Artificial Inteligence Debate, 123-141. MT Press, 1988.

[SMO88] SMOLENSKY, P. **On the proper treatment of connectionism behavioral and brain sciences**, v.11, p.1-74, 1988.

[STO90] STOLKE, A. **Learning feature-based semantics with simple recurrent networks**. *In Proceedings of the 13th Annual Conference of the Cognitive Science Society*, 913-917. Hillsdale, NJ: Erlbaum, 1990.

[SUB95] SUN R.; Bookman L. A., (Ed.), **Computational architectures integrating neural and Symbolic Processes: A Perspective on the state of the art.** Massachusetts:Kluwer Academic Publisher, 1995.

[TIN96] TINO, P. at al. **Finite state machine and recurrent neural networks - Automata and dynamical systems approaches**. Technical Reports CS-TR-3396. Institute for Advanced Computer Studies, University of Maryland. 1996.

[TOU91] TOURETZKY, D. S. **Special issue on connectionist aproaches to language-learning - introduction**. Machine Learning, 7(3):105-107, 1991.

[WAK92] WATRAUS, R.; Kulus, G. **Induction of finite state languages using second-order recurrent networks**. Neural Computation, n.4, n.3, p.406-414, 1992.

[WAP85] WALTZ, D. L.; Pollack, J. B. **Massively Parallel parsing: A strongly interative model of natural language interpretation**. Cognitive Science, 9:51-74, 1985.

[WAS89] WASSERMAN, P. D. **Neural Computing: Theory and Practice**. Van Nostrand Reinhold, New York, 1989.

[WEI86] WEISCHEDEL, R. M. **Knowledge Representation and Natural Language Processing**. Proceedings of IEEE, Special Issue on Natural Language Processing, v.74, n.7, p.905-920. July 1986.

[WER74] WERBOS, P. J. **Beyond regression: New tools for prediction and analysis in the behavioral sciences**. Ph.D thesis, Haward University, Cambridge, MA, 1974.

[WER95] WERMTER, S. **Hibrid connectionist natural language processing**. London: Chapman & Hall, 1995.

[WHH89] WAIBEL, A.; Hanazawa, T.; Hinton, G. E.; Shikano, K. et al. **Phoneme recognition using time-delay networks**. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37:328-339, 1989.

[WIH60] WIDROW, B.; Hoff, M. E. **Adaptive switching circuits**. In: 1960 IRE WESCON *Convention Record*, Part 4, p.96-104. New York: IRE.

[WIN83] WINOGRAD, T. **Language as a Cognitive Process**. Massachusetts: Addison-Wesley Publishing Company, Inc., 1983. v.I: Sintax.