Anais da 2° Jornada USP - SUCESU-SP de Informática e Telecomunicações - 30 de maio a 01 de junho de 1994 - Escola Politécnica - Depto. de Engenharia de Eletricidade - Universidade de São Paulo - São Paulo - SP - Brasil, 139-148.

LÓGICA E CONEXIONISMO EM PROCESSAMENTO DE LINGUAGEM NATURAL

João Luís Garcia Rosa
Departamento de Eletrônica e Computação - Instituto de Informática – PUC-Campinas
Rod. D. Pedro I, km. 136 - Caixa Postal 317
13.020-904 - Campinas - SP - Fone: (0-XX-19) 756-7195 Fax: (0-XX-19) 756-7162
e-mail: joaol@ii.puc-campinas.br

Márcio Luiz de Andrade Netto
Departamento de Eng. de Computação e Automação Industrial - Faculdade de Engenharia Elétrica – Unicamp
Caixa Postal 6101
13081-000 - Campinas - SP - Fone: (0-XX-19) 788-3784 Fax: (0-XX-19) 788-3706
e-mail: marcio@dca.fee.unicamp.br

RESUMO: As várias abordagens existentes para o Processamento de Linguagem Natural (PLN) se traduzem em aplicação de técnicas de Inteligência Artificial, ou baseadas em lógica ou em sistemas conexionistas. Este trabalho faz um "mix" destas duas abordagens, acrescentando a extensão temporal da análise da sentença. O resultado é um sistema que faz a análise sintática, baseada na lógica de predicados do Prolog, as análises semântica e recorrente, baseadas numa abordagem de redes neurais, de frases da língua portuguesa. Este sistema possui léxico e regras de sintaxe limitados. Mas, para este universo, os resultados obtidos foram bastante satisfatórios.

1. INTRODUÇÃO

Existem várias abordagens para o processamento de linguagem natural (PLN). E existem muitos trabalhos publicados nas várias abordagens. Entretanto, combinações das diversas abordagens existentes são raras. Este trabalho ousa em misturar uma abordagem sintática baseada na lógica de predicados, uma abordagem conexionista, baseada em atribuições de papéis de caso às palavras, referente à análise semântica, e uma abordagem recorrente, que leva em consideração características temporais da análise da sentença.

A grande maioria dos trabalhos publicados tratam de processamento da língua inglesa. Houve necessidade da transposição de procedimentos e idéias para a língua portuguesa. A língua portuguesa traz a grande dificuldade no tratamento de tempos verbais, mas em compensação, tem menos palavras ambíguas no seu léxico.

E por falar em ambigüidade, este é o maior desafio enfrentado pelos sistemas que tratam da linguagem natural. Identificar o verdadeiro significado de uma determinada palavra pode ser tão complicado, que às vezes só é possível com uma consulta ao usuário. Neste sistema, não se tem a intenção de resolver o problema da ambigüidade, mas apenas contribuir com idéias e apontar direções, quando talvez se possa transpor, ao menos parcialmente, este obstáculo.

1.1. A análise da forma

Este trabalho sobre PLN se divide, basicamente, em três etapas. A primeira trata da **análise sintática** de frases da língua portuguesa. Esta implementação foi baseada na Gramática de Cláusulas Definidas de Pereira e Warren (1980). São frases afirmativas, compostas por até quatro elementos-chave, que são o sujeito, o verbo, o objeto e o complemento, que pode ser o instrumento ou o modificador. Estes elementos-chave podem vir acompanhados de outras palavras como artigos, adjetivos, partículas reflexivas, etc. A análise sintática faz verificação de concordância de gênero e número, além de montar uma estrutura, chamada de estrutura-chave, constando apenas dos elementos-chave, que alimentará o analisador semântico (segunda etapa). Por exemplo, a frase

A menina bonita quebrou a frágil vidraça com um martelo. (1)

gerará a estrutura-chave

menina-quebrar-vidraça-martelo

onde *menina* é o sujeito, *quebrar* é o verbo, *vidraça* é o objeto e *martelo* é o instrumento. Note que uma frase nunca tem, ao mesmo tempo, um instrumento e um modificador. Uma frase pode ser

Todos os homens comem macarrão com cenouras. (2)

onde *cenouras* é o modificador de *macarrão*. A estrutura-chave da frase anterior será

homem-comer-macarrão-cenoura

É claro que uma frase pode não estar completa. Por exemplo, na frase

O homem se moveu. (3)

não há objeto e nem complementos (o verbo *mover* aqui é reflexivo).

O analisador sintático é baseado na lógica de predicados e foi implementado em Prolog.

1.2. A análise do significado

A segunda etapa trata da **análise semântica**. Nesta etapa a frase, já analisada sintaticamente pelo analisador sintático e já no formato de estrutura-chave, vai alimentar a entrada de uma rede de elementos com três camadas (chamada de abordagem **conexionista**), que dirá se a frase tem significado.

Esta etapa foi baseada nos trabalhos de McClelland e Kawamoto (1986) e Waltz e Pollack (1985), que tratam a palavra como um conjunto de microcaracterísticas semânticas. Ou seja, toda palavra é descrita como um vetor de bits, onde cada subconjunto de bits tem um significado associado, como *humano-não humano*, *frágil-duro*, *masculino-feminino*, etc.

A rede é alimentada com a representação canônica da palavra, ou seja, com o seu conjunto de microcaracterísticas semânticas. Na verdade para um determinado verbo, existem quatro redes: uma para o agente, uma para o paciente, uma para o instrumento e uma para o modificador. Por exemplo, para a frase (1), a rede do agente é ativada para uma matriz triangular, que é o confronto das microcaracterísticas de *menina* com elas mesmas. Esta estrutura é chamada de **estrutura de sentença**. O formato da saída da rede é chamada de **estrutura de caso**, que é o confronto das microcaracterísticas de *menina* com as do verbo *quebrar*. O processo se repete para as outras redes.

Como as redes foram treinadas para várias frases, elas têm condição de verificar se uma frase nova, ou seja, uma frase não conhecida, está ou não semanticamente correta. O algoritmo usado para implementar estas redes foi o "**backpropagation**". Este algoritmo consiste basicamente no seguinte. Primeiro, atribui-se pesos aleatórios às ligações entre os elementos das redes. Ao se entrar com uma estrutura de sentença, a saída da rede é comparada com a saída desejada, ou seja, a saída que deveria ocorrer caso a rede tivesse aprendido aquela estrutura. As ligações são enfraquecidas ou fortalecidas, para "corrigir" a saída real. Este processo é repetido dezenas de vezes, até que a rede atinja uma situação de convergência, ou seja, até que a rede "aprenda" aquela estrutura.

O sistema conexionista consiste de duas partes. A primeira, a fase do **aprendizado**, consiste na apresentação seqüencial de frases diferentes, mas corretas semanticamente, e na correção de pesos descrita acima, tudo isto dezenas de vezes (épocas). A segunda parte, a fase do **reconhecimento**, onde é apresentada uma frase, nova ou não, à rede, e ela deve ser capaz de, numa única época, dizer se esta frase está correta semanticamente.

Esta etapa foi implementada na linguagem de programação Pascal.

1.3. A análise temporal

A terceira etapa, a **análise recorrente**, trata da verificação das seqüências de palavras previstas. Pode-se, com esta etapa, verificar se as palavras estão numa seqüência apropriada. Esta abordagem foi baseada no trabalho de Elman (1990).

A análise semântica não faz verificação, em uma frase, de elemento para elemento. Só do elemento para o verbo e do paciente para o modificador, no caso deste existir. Ou seja, na frase (2), apenas existe verificação de relação entre *homem* e *comer*, entre *macarrão* e *comer* e entre *macarrão* e *cenoura*. Não há verificação entre *homem* e *macarrão* e nem entre *homem* e *cenoura*. Caso se deseje fazer esta verificação, utiliza-se uma rede recorrente, que é uma rede provida de memória, onde pode-se conhecer uma determinada seqüência de palavras que se ensinou.

O algoritmo empregado nesta rede também foi o "backpropagation". A diferença é que esta rede tem uma camada a mais, onde se armazena o estado anterior (memória). Na fase do aprendizado, ensina-se à rede todas as seqüências de palavras possíveis para todas as frases possíveis. Na fase de reconhecimento, entra-se com uma palavra e a rede fornece a próxima na seqüência. Esta etapa foi implementada conjuntamente com o analisador semântico, portanto também em Pascal.

2. A ARQUITETURA DO MODELO

A meta primária deste modelo é prover um mecanismo que possa começar a considerar conjuntamente o papel da ordem da palavra e restrições semânticas na atribuição do papel. Deseja-se que o modelo seja capaz de **aprender** a fazer isto baseado em experiência com sentenças e suas representações de caso. Deseja-se que este modelo seja capaz de **generalizar** o que aprendeu para novas sentenças formadas de combinações de palavras.

O modelo consiste de dois conjuntos de unidades: um para representar a estrutura superficial da sentença e um para representar sua estrutura de caso. O modelo aprende através de apresentações de pares corretos de estrutura superficial e estrutura de caso; durante o treinamento, é apresentada à entrada a estrutura superficial e examina-se a saída que o modelo gera no nível de estrutura de caso.

2.1. Sentenças.
As sentenças processadas pelo modelo consistem de um verbo e de um a três sintagmas nominais (SNs). Existe sempre um SN Sujeito e opcionalmente pode existir um SN Objeto. Se este estiver presente, pode também existir um com-SN; isto é, um SN em um sintagma preposicional de final de sentença começando com a palavra *com.*

2.2. Formato de entrada das sentenças.
O que o modelo vê como entrada não é a sentença propriamente dita, mas uma representação canônica da estrutura constituinte da sentença, na forma que poderia ser produzida por um simples analisador de superfície e um simples léxico.

2.3. Microcaracterísticas semânticas.
Nos formatos de entrada canônicos, as palavras são representadas como listas de microcaracterísticas semânticas (Waltz e Pollack, 1985; McClelland e Kawamoto, 1986). Para substantivos e verbos, as características são agrupadas em muitas dimensões. Cada dimensão consiste de um conjunto de valores mutuamente exclusivos e, em geral, cada palavra é representada por um vetor no qual um, e apenas um, valor em cada dimensão está "ON" (ligado) para a palavra e todos os outros valores estão "OFF" (desligados). Valores que estão "ON" são representados nos vetores de características como "1"s. Valores que estão "OFF" são representados por pontos (".").
Foram escolhidas as dimensões e os valores em cada dimensão para capturar o que se considerou dimensões importantes de variações semânticas nos significados de palavras que tinham implicações para a atribuição de papéis das palavras (McClelland e Kawamoto, 1986).

2.4. Unidades de estrutura de sentença.
A representação do nível de estrutura de sentença de uma sentença de entrada não é o conjunto de vetores de características constituintes; ela é o padrão de ativação que esses vetores produzem sobre as unidades que correspondem a *pares* de características. Estas unidades são chamadas unidades de estrutura de sentença (ES).
Cada unidade ES representa a conjunção de duas microcaracterísticas do preenchedor de um papel de superfície particular. Como há quatro papéis de estrutura de sentença, existem quatro conjuntos de unidades ES. Dentro de cada conjunto existe uma unidade que representa a conjunção de todo valor de microcaracterística em cada dimensão com todo valor de microcaracterística em qualquer outra dimensão.
Enquanto o modelo funciona bem com esta simulação, presume-se que simulações que usem um léxico maior requereria maior diferenciação de algumas representações de substantivos e verbos.

2.5. Representação de papel de caso.
A representação de papel de caso tem uma forma levemente diferente da representação de estrutura de sentença. Para entender esta representação, é útil voltar a um ponto de vista mais abstrato e considerar mais genericamente como se deve representar uma descrição estrutural numa representação distribuída. Em geral uma descrição estrutural pode ser representada por um conjunto de triplas da forma (A R B) onde A e B correspondem aos nós na descrição estrutural e R representa a relação entre os nós. Por exemplo, uma hierarquia de inclusão de classes pode ser representada por triplas da forma (X É-UM Y), onde X e Y são nomes de categorias. Qualquer outra descrição estrutural, seja uma estrutura de constituinte sintático, uma estrutura de constituinte semântico ou qualquer outra coisa, pode ser representada desta forma. Especificamente, a atribuição de papel de caso dos constituintes da sentença *O garoto quebrou a vidraça com o martelo* pode ser representada como (McClelland e Kawamoto, 1986):

Quebrou Agente Garoto
Quebrou Paciente Vidraça
Quebrou Instrumento Martelo

A estrutura constituinte de uma sentença tal como *O garoto comeu o macarrão com molho* poderia ser representada por:

Comeu Agente Garoto
Comeu Paciente Macarrão
Macarrão Modificador Molho

Uma das metas para o modelo é mostrar como ele pode selecionar o significado apropriado no contexto para uma palavra ambígua. Para palavras ambíguas (*galinha*, viva ou cozida) o padrão de entrada é a média dos padrões de características de cada uma das duas leituras da palavra. Isto significa que nos casos onde as duas concordam com o valor de uma dimensão de entrada particular, esta dimensão tem o valor acordado na representação de entrada. Nos

casos onde os dois discordam, a característica tem o valor de .5 na representação de entrada. Uma meta é ver se o modelo pode corretamente preencher estes valores não especificados, efetivamente recuperando os valores perdidos do contexto no processo da atribuição da palavra ao caso apropriado.

3. DETALHES DO PROCESSAMENTO DE SENTENÇA E APRENDIZADO

Na apresentação de uma sentença, a entrada da rede para cada uma das unidades de estrutura da sentença é determinada, baseada nos vetores de características das palavras. Cada uma dessas unidades é então ligada probabilisticamente. Cada unidade de estrutura de superfície tem uma conexão modificável para cada uma das unidades de estrutura de caso. Em adição, cada unidade de estrutura de caso tem uma polarização modificável (equivalente a uma conexão a partir de uma unidade especial que está sempre ligada). Baseado no padrão de estrutura de sentença e nos valores correntes dos pesos, uma entrada da rede para cada unidade de estrutura de caso é computada. Unidades de estrutura de caso têm valores de ativação 0 e 1 e a ativação é uma função probabilística da entrada da rede, que é implementada com o algoritmo "backpropagation".

Durante o aprendizado, a ativação resultante de cada unidade de estrutura de caso é comparada ao valor que ela deveria ter na leitura correta da sentença. A leitura correta é fomecida como uma "entrada mestre" especificando quais unidades de papéis de caso devem estar ligadas. A idéia é que esta entrada mestre seja análoga a representação que um aprendiz de linguagem real deveria construir da situação na qual a sentença ocorreria. O aprendizado simplesmente corresponde ao ajuste de pesos de conexão para fazer a saída gerada pelo modelo corresponder o mais próximo possível a entrada mestre.

4. EXPERIMENTOS DE SIMULAÇÃO

O mais importante sobre o modelo é o fato de que sua resposta a novos impulsos é estritamente dependente de sua experiência. Na avaliação de seu comportamento então, é importante ter conhecimento ao que ele foi exposto durante o aprendizado.

A experiência principal consiste na geração de várias sentenças derivadas dos "frames" de sentenças previstos. Deve ser enfatizado que estes "frames" de sentenças são simplesmente usados para gerar um conjunto de sentenças válidas. Cada "frame" especifica um verbo, um conjunto de papéis e uma lista de possíveis preenchedores de cada papel. Portanto, o "frame" de sentença *O humano quebrou o objeto_frágil com o quebrador* é simplesmente um gerador para todas as sentenças na qual *humano* é substituído por uma das palavras na lista de humanos, *objeto_frágil* é substituído por uma das palavras na lista de objetos frágeis e *quebrador* é substituído por uma das palavras da lista de quebradores. É claro que estes geradores não capturam todas as propriedades dos elementos envolvidos em cenários reais e então não se pode esperar que o modelo represente fielmente todas as sutilezas.

Ao modelo foi dado 20 ciclos de treinamento, com o conjunto de sentenças de treino. Em cada ciclo, cada sentença era apresentada, a resposta do modelo era gerada e os pesos de conexão era ajustados de acordo com o procedimento "backpropagation".

4.1. Redes "backpropagation"

A habilidade para treinar redes com várias camadas é um passo importante na direção da construção de máquinas inteligentes a partir de componentes parecidos com os neurônios. A meta é pegar uma massa de elementos processadores, que simulam a célula nervosa, e ensiná-la a realizar tarefas úteis. É desejável que ela seja rápida e resistente a danos. É desejável que generalize a partir das entradas que vê.

A existência da camada escondida permite que a rede desenvolva representações internas. O comportamento destas unidades escondidas é automaticamente aprendido, não é pré-programado.

A maior propriedade dos sistemas conexionistas é que a rede neural não aprende apenas a classificar as entradas nas quais ela é treinada, mas também a **generalizar** e ser capaz de classificar entradas nunca vistas.

Uma limitação das redes atuais é como elas tratam com fenômenos que envolvem o tempo. Esta limitação é resolvida, de certa forma, pelas redes recorrentes, mas os problemas ainda são muitos.

Uma rede "backpropagation" tipicamente inicia com um conjunto de pesos aleatórios. A rede ajusta seus pesos cada vez que ela vê um par entrada-saída. Cada par requer dois estágios: um passo para frente e um passo para trás. O passo para frente envolve a apresentação de uma amostra de entrada à rede e as ativações se propagam até alcançarem a camada de saída. Durante o passo para trás, a saída real da rede (do passo para frente) é comparada com a saída desejada e as estimativas de erro são calculadas para as unidades de saída. Os pesos conectados às unidades de saída podem ser ajustados a fim de reduzir estes erros. Pode-se usar as estimativas de erro das unidades de saída para

derivar as estimativas de erro para as unidades das camadas escondidas. Finalmente, os erros são propagados de volta às conexões que tiveram origem nas unidades de entrada.
O algoritmo "backpropagation" geralmente atualiza seus pesos, incrementando-os, depois de ver cada par entrada-saída. Depois de visto todos os pares entrada-saída (e ajustados seus pesos muitas vezes), diz-se que uma **época** completou-se. O treinamento de rede "backpropagation" usualmente requer muitas épocas.
Generalização. Se todas as entradas e saídas possíveis são mostradas à uma rede "backpropagation", a rede achará (provavelmente, eventualmente) um conjunto de pesos que mapeia as entradas nas saídas. Para muitos problemas de Inteligência Artificial, entretanto, é impossível fornecer todas as entradas possíveis. Para resolver este problema, a rede "backpropagation" é boa no mecanismo de generalização. Se se trabalha num domínio onde entradas similares são mapeadas em saídas similares[1], a rede "backpropagation" irá interpolar quando forem fornecidas entradas que a rede nunca viu antes.

5. RESULTADOS

Vai-se descrever as saídas do modelo, durante as simulações. A execução foi feita num microcomputador PC AT 286, de 20 MHz, com 1 MByte de memória principal.

5.1. O analisador sintático

As saídas do analisador sintático. Quando o analisador sintático é executado, a seguinte mensagem aparece no vídeo:

```
Entre com a frase:
```

Ao se entrar com uma frase entre aspas, o analisador sintático faz a verificação da sua estrutura, baseado nas regras sintáticas que possui e no seu vocabulário e mostra a sua estrutura-chave, ainda no formato inicial, com colchetes e vírgulas. Ao se entrar com a frase "*um homem bateu em uma mulher com uma pedra*", o resultado será (entradas sublinhadas):

```
Entre com a frase: "Um homem bateu em uma mulher com uma pedra".
[[[homem],x],[bater,[[[mulher],[[[pedra],x]]]]]]
Frase CORRETA sintaticamente

Entre com a frase:
```

5.2. O analisador semântico

Foi implementado um programa em Pascal, que "gera" uma matriz de 20 x 20 baseado no confronto de microcaracterísticas de verbo e substantivos, com pesos que são incrementados quando se apresenta uma nova sentença à "rede". É lógico que há a necessidade de se implementar um algoritmo de redes neurais para mexer nas conexões. Também há a necessidade de verificar como proceder na fase de **reconhecimento**, que é quando se introduz uma frase diferente da ensinada para ver como o modelo se comporta. Deve-se implementar também, o gerador de frases para treinamento da rede.
O número de ciclos necessários. Segundo McClelland e Kawamoto (1986), conclui-se que (para vetor de 25 microcaracterísticas) para sentenças familiares, após o 20º ciclo de aprendizado, a alteração em relação ao aprendizado muda pouco (no 20º, a média de bits incorretos é 48, o que significa um erro de 1,92%, pois foram treinadas 2500 unidades de papel de caso, e no 50º, a média é 33, o que significa um erro de 1,32%. Logo, a diferença é muito pequena (0,6%) e, portanto, perfeitamente admissível.) Observe-se que este sistema trata de sentenças da língua inglesa, porém foi observado experimentalmente que isto também é verdade para o sistema implementado.
Baseado nisto, vai-se assumir 20 ciclos para a fase de aprendizado da rede.

---

[1] No caso deste trabalho, como as palavras são descritas por microcaracterísticas semânticas, existem palavras próximas no significado (como, por exemplo, *homem* e *menino*), que têm muitas microcaracterísticas em comum, ou seja, seus vetores de microcaracterísticas são próximos.

O número de microcaracterísticas semânticas necessárias. McClelland e Kawamoto (1986) trabalham com vetores de 25 microcaracterísticas semânticas. Por questões de tempo e por limitação de memória, reduziu-se o vetor para 20 bits.
O modelo de rede adotado tem três camadas: a de entrada, a escondida e a de saída. Como a entrada recebe a estrutura da sentença, para 20 microcaracterísticas tem-se 190 unidades de entrada. Como a saída corresponde à estrutura de caso, tem-se 400 unidades de saída. O número de unidades da camada escondida é escolhida arbitrariamente, considerando as limitações do ambiente Turbo Pascal 6.0, onde este software foi desenvolvido, de permitir estruturas de no máximo 64 Kbytes (cada matriz de pesos terá o número de unidades da camada anterior multiplicado pelo número de unidades da camada posterior, multiplicado por 4, que é o tamanho em bytes do menor número real do Pascal). Assumiu-se, inicialmente, que a camada escondida teria 25 unidades.
A fase de reconhecimento dar-se-á quando se desejar descobrir a correção semântica de determinada frase. Entra-se com a frase, busca-se os arquivos correspondentes àquele verbo e a propagação através da rede se dá apenas uma vez, logo, o processo será extremamente rápido.
Saídas do analisador semântico. O analisador semântico traz como saída inicial, um "menu" de opções, onde o usuário entra com a opção de querer executar o analisador semântico (comandos 1 e 2) ou recorrente (comandos 3 e 4). Para o analisador semântico, o comando 1 consiste na fase do aprendizado. Vai-se "ensinar" à rede, todas as palavras que deseja-se que a rede conheça. Isto é feito automaticamente pelo gerador de frases.

```
MENU:
Deseja: 1- aprender palavra
        2- reconhecer palavra
        3- aprender seqüência
        4- reconhecer seqüência
1
Memória antes = 403536
Memória depois = 220928
 *** FASE DE TREINAMENTO ***
 Primeira vez? (S/N):
```

Quando se digita "N" (ou seja, já se "treinou" a rede com alguma estrutura), o analisador semântico pede a próxima frase a ser entrada no sistema (este comando permite que se "treine" a rede aos poucos, pois o treinamento completo pode ser um pouco demorado), e fornece a seguinte tela:

```
 Primeira vez? (S/N): n
 Qual verbo? (bater/comer/mover/quebrar): bater
 Qual classe? (agente/paciente/instrumento/modificador): agente
Aguarde! São 20 ativações:
```

Já quando se entra com a opção "S" (correspondente a primeira vez que se treina a rede), o analisador semântico começa a treinar a rede para muitas frases geradas pelo gerador. O primeiro verbo a ser treinado é o verbo *bater*. Depois, vem os outros. Para o verbo *bater*, o primeiro sujeito gerado é *homem.* Depois, *menina*, e assim por diante, até que se tenha esgotadas todas as frases previstas pelo gerador. Então, completa-se uma ativação (uma época). Todas as ativações acontecem para todos os substantivos previstos para o verbo em questão. Note que no início da primeira ativação, todos os valores correspondentes aos pesos de conexões estão próximos de 0.5, pois eles são gerados aleatoriamente entre 0.0 e 1.0. Depois de algumas vezes, o programa gera uma saída mais próxima dos valores reais, pois a rede vai corrigindo os pesos das conexões para ficar parecido com a saída desejada (procedimento *backpropagation*):

```
Ativação 1 - Verbo: Bater - Sujeito: menina
Vetor de saída:
0.8 0.2 0.8 0.2 0.7 0.3 0.2 0.3 0.7 0.2 0.2 0.8 0.2 0.8 0.2 0.8 0.2 0.2 0.2 0.8
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.1 0.2 0.2 0.2 0.2 0.2 0.2
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.8 0.2 0.8 0.2 0.7 0.3 0.2 0.3 0.7 0.2 0.2 0.8 0.2 0.8 0.2 0.8 0.2 0.2 0.2 0.8
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.1
0.8 0.2 0.8 0.2 0.7 0.3 0.2 0.3 0.8 0.2 0.2 0.8 0.2 0.8 0.2 0.8 0.2 0.2 0.2 0.8
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.1 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.8 0.2 0.8 0.2 0.7 0.3 0.2 0.3 0.7 0.2 0.2 0.8 0.2 0.8 0.2 0.8 0.2 0.2 0.2 0.8
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.8 0.2 0.8 0.1 0.7 0.3 0.2 0.3 0.7 0.2 0.2 0.8 0.2 0.8 0.2 0.8 0.2 0.2 0.2 0.8
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.8 0.2 0.8 0.2 0.7 0.3 0.2 0.3 0.7 0.2 0.2 0.8 0.2 0.8 0.2 0.8 0.2 0.2 0.2 0.8
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.8 0.2 0.8 0.2 0.7 0.3 0.2 0.3 0.7 0.2 0.2 0.8 0.2 0.8 0.2 0.8 0.2 0.2 0.2 0.8
```

Para a opção 2 do "menu" principal (fase do reconhecimento), a rede já "aprendeu" todas as frases previstas para cada verbo e está pronta para reconhecer uma sentença. O sistema pergunta ao usuário se deseja entrar com a frase ou se deseja que o mesmo leia do disco a estrutura-chave da última sentença analisada sintaticamente.

```
MENU:
Deseja: 1- aprender palavra
        2- reconhecer palavra
        3- aprender seqüência
        4- reconhecer seqüência
2
Memória antes = 384416
Memória depois = 201808
 Gostaria de entrar com a frase? (S/N):
```

Caso se digite "N", o sistema vai buscar em disco a estrutura-chave gerada pelo analisador sintático (neste caso, *um homem bateu em uma mulher com uma pedra*). Observe os vetores média resultante e desejado. Quando a diferença entre os valores das dimensões entre os dois vetores for inferior a 0.5, considera-se que estes valores são equivalentes. Até cinco valores diferentes são permitidos no vetor de 20 microcaracterísticas:

```
*** FASE DE RECONHECIMENTO ***
O sujeito é homem
O verbo é bater
O objeto é mulher
O complemento é pedra
O modificador é pedra
*** LEITURA DO DISCO ***
Verbo: bater - Sujeito: homem
Vetor Média Resultante:
0.9 0.1 0.9 0.1 0.9 0.1 0.1 0.1 0.9 0.1 0.1 0.9 0.1 0.9 0.1 0.9 0.1 0.0 0.1 0.9
Vetor Média Desejado:
0.9 0.1 0.9 0.1 0.9 0.1 0.1 0.1 0.9 0.1 0.1 0.9 0.1 0.9 0.1 0.9 0.1 0.1 0.1 0.9
Tecle ENTER para continuar...
*** LEITURA DO DISCO ***
Verbo: bater - Objeto: mulher
Vetor Média Resultante:
0.9 0.1 0.9 0.1 0.3 0.7 0.1 0.7 0.3 0.1 0.1 0.9 0.1 0.9 0.1 0.9 0.1 0.1 0.1 0.9
Vetor Média Desejado:
0.9 0.1 0.9 0.1 0.1 0.9 0.1 0.1 0.9 0.1 0.1 0.9 0.1 0.9 0.1 0.9 0.1 0.1 0.1 0.9
Tecle ENTER para continuar...
*** LEITURA DO DISCO ***
Verbo: bater - Complemento: pedra
Vetor Média Resultante:
0.2 0.9 0.4 0.7 0.6 0.4 0.9 0.2 0.2 0.7 0.2 0.4 0.2 0.9 0.2 0.9 0.2 0.4 0.7 0.2
Vetor Média Desejado:
0.1 0.9 0.1 0.9 0.1 0.9 0.9 0.1 0.1 0.1 0.1 0.9 0.9 0.1 0.1 0.9 0.1 0.1 0.9 0.1
Tecle ENTER para continuar...
*** LEITURA DO DISCO ***
Verbo: bater - Modificador: pedra
Vetor Média Resultante:
0.1 1.0 1.0 0.1 0.0 1.1 1.1 0.0 0.1 0.1 0.1 1.0 0.1 1.0 0.1 1.0 0.1 1.1 0.1 0.0
Vetor Média Desejado:
0.1 0.9 0.1 0.9 0.1 0.9 0.9 0.1 0.1 0.1 0.1 0.9 0.9 0.1 0.1 0.9 0.1 0.1 0.9 0.1
Tecle ENTER para continuar...
```

Como o sistema não tem condições de distinguir o Instrumento e o Modificador, então ele trata a palavra como se fosse as duas coisas (veja no exemplo acima a palavra *pedra* é tratada como Complemento (Instrumento) e como Modificador.
A saída do reconhecimento semântico, caso se digite a opção "S" é a seguinte, com a inclusão da estrutura *homem-quebrar-vidraça-martelo*:

```
 Gostaria de entrar com a frase? (S/N): s
 *** FASE DE RECONHECIMENTO ***
Entre com a frase: Sujeito: homem
Verbo: quebrou
Objeto: vidraça
Complemento: martelo
Modificador:
*** LEITURA DO DISCO ***
```

Neste ponto, o analisador avisa que vai consultar arquivos do disco, onde estão armazenados os pesos das ligações sinápticas da rede para o verbo *quebrar*. Depois de efetuada esta consulta, o analisador fornece a seguinte saída, para o sujeito *homem*:

```
Verbo: quebrou - Sujeito: homem
Vetor Média Resultante:
0.9 0.1 1.0 0.0 0.9 0.1 0.0 0.1 0.9 0.1 0.0 1.0 -0.0 1.0 -0.0 1.0 0.1 0.1 0.0 0.9
Vetor Média Desejado:
0.9 0.1 0.9 0.1 0.9 0.1 0.1 0.1 0.9 0.1 0.1 0.9 0.1 0.9 0.1 0.9 0.1 0.1 0.1 0.9
Tecle ENTER para continuar...
```

E também fornece os resultados para os demais substantivos da frase.

5.3. Rede recorrente

Baseado no artigo do Elman sobre Redes Recorrentes (1990), implementou-se uma rede recorrente que faz a "previsão" de ordem de seqüência de palavras e que tem como entrada a seqüência de palavras para cada verbo e a rede então deve ser capaz de "prever" a próxima palavra na seqüência, e com isso, fazer uma interligação entre os elementos da sentença.

Foram feitas modificações no programa de rede recorrente de Elman (1990) para que funcione com entradas de 20 elementos (as 20 microcaracterísticas dos substantivos). A rede copia a camada de saída na camada de contexto (memória), ao invés de copiar a camada escondida, como faz Elman. Modificou-se também, o gerador de palavras para que gere as combinações possíveis de seqüências previstas. Modificou-se a entrada, para que se entre com a palavra, ao invés de seu vetor de microcaracterísticas. A saída fornece também a palavra seguinte na seqüência treinada.

Saídas do analisador recorrente. O treinamento do analisador recorrente, referente ao comando 3, gera toda a seqüência de palavras previstas e treina o modelo durante 30 ativações, com a palavra de entrada, a saída esperada e o vetor de saída real. Como saída inicial, o analisador fornece a seguinte tela:

```
Aguarde! São 30 ativações:
Verbo - Bater; Entrada: homem
1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 1.0
Saída: homem
1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 1.0
Ativação 1
Vetor de saída:
0.5 0.5 0.5 0.6 0.5 0.4 0.6 0.5 0.5 0.4 0.5 0.6 0.6 0.6 0.5 0.5 0.7 0.6 0.5 0.5
```

No início da segunda ativação, o analisador começa a repetir toda a seqüência de palavras da primeira ativação. E assim por diante, para as trinta ativações.

Para a fase de reconhecimento (comando 4), o analisador fornece a próxima palavra na seqüência ensinada à rede. Para o caso de *homem*, para o verbo *bater*, o analisador não acha a próxima palavra na seqüência, provavelmente por causa das muitas palavras próximas de *homem* ensinadas ao modelo (tais como *menina*, *garoto*, etc.).

6. CONCLUSÃO

Este trabalho trouxe como contribuição para o processamento de linguagem natural, uma abordagem mista de lógica e conexionismo. Alcançou-se, com o mesmo, resultados bastante satisfatórios dentro do plano proposto. Espera-se poder continuar o presente trabalho, propondo uma versão com vocabulário e estruturas mais ricos, onde se pretende que formações mais complexas da língua portuguesa possam ser implementadas. Para isto é necessário aumentar as regras do analisador sintático, juntamente com sua base de dados (fatos). É necessário também, alterar o analisador semântico, construindo redes maiores, onde se permita adequar o tamanho dos vetores de microcaracterísticas semânticas às novas dimensões dadas as palavras, absolutamente necessárias para dar uma diferenciação entre as mesmas. Pode-se, utilizando máquinas mais rápidas, aumentar o número de épocas para treinamento das redes, contribuindo com isso para uma maior eficiência.

7. BIBLIOGRAFIA

1. Chomsky, N. (1972). "Syntactic Structures". Mouton, The Hague - Paris.
2. Elman, J. L. (1990). "Finding Structure in Time". *Cognitive Science* 14, 179-211.

3. Fanty, M. A. (1986). "Context-Free Parsing with Connectionist Networks". AIP Conference Proceedings 151, Neural Networks for Computing - Snowbird, UT, Editor: John S. Denker. pp. 140-145.
4. Feldman, J. A. (1982). "Dynamic Connections in Neural Models" *Biological Cybernetics*, 46, 27-39.
5. Feldman, J. A. & Ballard, D. H. (1982). "Connectionist Models and Their Properties". *Cognitive Science* 6, 205-254.
6. Gazdar, G. & Mellish, C. (1989). "Natural Language Processing in Prolog - An Introduction to Computational Linguistics". Addison-Wesley Publishing Company.
7. Geetha, T. V. & Subramanian, R. K. (1990). "Representing Natural Language with Prolog". *IEEE Software*. pages 85-92, vol. 7, no. 2.
8. Hillis, W. D. (1985). "The Connection Machine". The MIT Press.
9. Hinton, G. E. (1992). "How Neural Networks Learn From Experience". *Scientific American*, September, Vol. 267, No. 3, pp. 105-109.
10. Israel, D. J. (1983). "The role of logic in knowledge". *IEEE Computer*, pages 37-41, October.
11. Kohonen, T. (1984). "Self-Organization and Associative Memory". Springer-Verlag.
12. Lange, T. E. (1992). "Lexical and Pragmatic Disambiguation and Reinterpretation in Connectionist Networks". *International Journal of Man-Machine Studies*, 36, 191-220.
13. Lippmann, R. P. (1987). "An Introduction to Computing with Neural Nets". *IEEE ASSP Magazine*, April, pp. 4-22.
14. Marcus, C. (1986). "Prolog Programming - Applications for Database Systems, Expert Systems, and Natural Language Systems". Addison-Wesley Publishing Co.
15. Mates, B. (1972). "Elementary Logic". Second edition. Oxford University Press.
16. McClelland, J. L. & Kawamoto, A. H. (1986). "Mechanisms of Sentence Processing: Assigning Roles to Constituents of Sentences". In Parallel Distributed Processing - Explorations in the Microstructure of Cognition, Volume 2. The MIT Press.
17. McClelland, J. L. & Rumelhart, D. E. (1986). "Parallel Distributed Processing - Explorations in the Microstructure of Cognition". Volume 2 - Psychological and Biological Models. The MIT Press.
18. McClelland, J. L. & Rumelhart, D. E. (1988). "Explorations in Parallel Distributed Processing - A Handbook of Models, Programs, and Exercises". A Bradford Book, The MIT Press.
19. Nilsson, N. J. (1982). "Principles of Artificial Intelligence". Springer-Verlag.
20. Pereira, F. C. N. & Warren, D. H. D. (1980). "Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks". *Artificial Intelligence* 13, 231-278.
21. Rich, E. & Knight, K. (1991). "Artificial Intelligence", Second Edition - International Edition. McGraw-Hill, Inc.
22. Rocha, A. F. (1990). "Simbolic Reasoning: A Natural Affair for K-Neural Nets". IFLSI Conference, Tizuka, Japan, July.
23. Rosa, J. L. G. & Andrade Netto, M. L. (1993), "Processamento de Linguagem Natural – Uma Abordagem Conexionista". Revista do Instituto de Informática no. 1, pp. 9-15. Instituto de Informática, Pontifícia Universidade Católica de Campinas. Junho.
24. Rosa, J. L. G. (1993). "Redes Neurais e Lógica Formal em Processamento de Linguagem Natural". Dissertação de Mestrado. Faculdade de Engenharia Elétrica e de Computação. Universidade Estadual de Campinas. Setembro.
25. Rumelhart, D. E. & McClelland, J. L. (1986). "Parallel Distributed Processing - Explorations in the Microstructure of Cognition". Volume I - Foundations. A Bradford Book. The MIT Press.
26. Sowa, J. F. (1984). "Conceptual Structures: Information Processing in Mind and Machine". Addison-Wesley Publishing Company.
27. Swinney, D. A. (1979). "Lexical Access During Sentence Comprehension: (Re)Consideration of Context Effects". *Journal of Verbal Learning and Verbal Behavior* 18, 645-659.
28. Waltz, D. L. & Pollack, J. B. (1985) "Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretations". *Cognitive Science* 9, 51-74.
29. Warner, A. J. (1988). "Natural Language Processing in Information Retrieval". *Bulletin of the American Society for Information Science*, pages 18-19, August/September.
30. Woods, W. A. (1970). "Transition Network Grammars for Natural Language Analysis". *Communications of the ACM*, volume 13, no. 10, October, pp. 591-606.