

**Métodos Empíricos para Correção
Gramatical de Línguas Naturais**

**Andréia Gentil Bonfante
Maria das Graças Volpe Nunes**

Nº 108

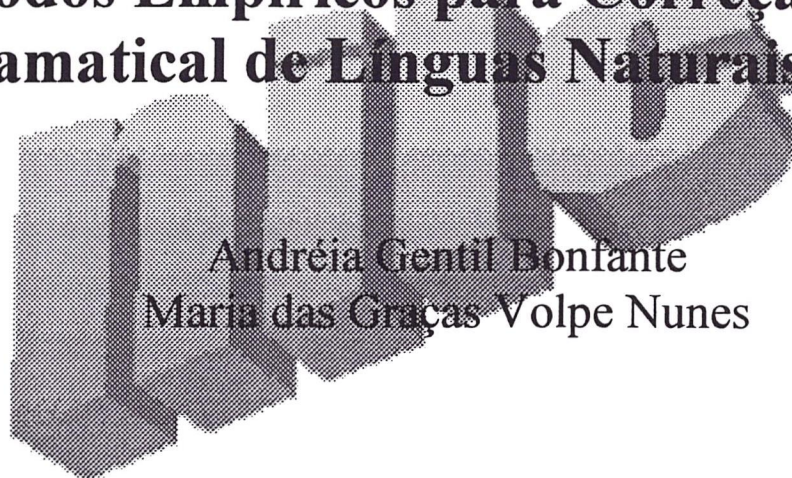
RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
Mar./2000

SYSNO	<u>1096924</u>
DATA	<u> / / </u>
ICMC - SBAB	

Universidade de São Paulo - USP
Universidade Federal de São Carlos - UFSCar
Universidade Estadual Paulista - UNESP

Métodos Empíricos para Correção Gramatical de Línguas Naturais



Andréia Gentil Bonfante
Maria das Graças Volpe Nunes

NILC-TR-00-4

Março, 2000

Série de Relatórios do Núcleo Interinstitucional de Lingüística Computacional
NILC - ICMC-USP, Caixa Postal 668, 13560-970 São Carlos, SP, Brasil

Métodos Empíricos para Correção Gramatical de Línguas Naturais*

Resumo

Este trabalho discute as limitações de métodos baseados em regras, ou métodos racionalistas, para tarefas como parsing e correção gramatical de línguas naturais, e apresenta os métodos baseados em corpus, ou empíricos, como possível alternativa para um bom desempenho nessas tarefas. Apresenta métodos conhecidos que foram aplicados à língua inglesa e aponta para a necessidade de futuros trabalhos aplicados à língua portuguesa.

1. Introdução

Muito esforço tem sido empregado na construção de sistemas de processamento de língua natural (PLN). Durante os anos 70, os sistemas construídos demonstravam aspectos interessantes de interpretação de língua em domínios restritos, como o Mundo dos Blocos (Winograd, 1970) ou sistemas de pergunta e resposta a bases de dados (Waltz, 1978; Woods, 1977).

Os anos 80 foram marcados pela construção de sistemas de PLN robustos (Allen, 1995), com gramáticas mais elaboradas, ainda na forma de regras compiladas manualmente (hand-coded). No entanto, os sistemas continuaram de domínio restrito e, portanto, bastante limitados. A codificação do conhecimento necessário chegou a grandes níveis de complexidade, e o aumento no volume das regras, juntamente com a conseqüente dificuldade de gerenciamento, fizeram com que as pesquisas não obtivessem avanços consideráveis durante esses anos.

A década de 90 trouxe muito interesse e pesquisa por novos paradigmas para guiar os caminhos do processamento de língua natural. O objetivo é tentar superar as dificuldades dos métodos usados anteriormente, os chamados métodos simbolistas.

Enquanto esses métodos **racionalistas**, baseados em regras compiladas, necessitam de um projetista, os novos métodos **empíricos**¹, ou baseados em corpus, têm por característica a

* Este trabalho tem o apoio do CNPq, da Finep e da Itaútec-Philco S.A.

¹ Ver em (Manning & Schütze, 1999), capítulo 1, uma discussão sobre métodos racionalistas X métodos empíricos.

eliminação do processo de representação manual. O conhecimento passa a ser adquirido automaticamente, através de grandes bases de exemplos (Brill & Mooney, 1997; Church & Mercer, 1993; Charniak, 1993).

As pesquisas com os métodos empíricos se estendem englobando três abordagens principais das áreas de aprendizado: o aprendizado de máquina simbólico, o aprendizado neural conexionista e o aprendizado estatístico.

As abordagens simbólicas para o aprendizado incorporam uma grande variedade de técnicas de aprendizado de máquina. Esses sistemas têm por característica a aquisição da informação de forma automática, processo chamado de indução, a partir de uma base de exemplos, e a representação do conhecimento inferido em formas que podem ser variadas. Para aplicações de linguagem, existem sistemas empregando várias dessas formas de representação, como árvores de decisão (Hermjakob & Mooney, 1997), regras de transformação (Brill, 1993, 1995) e programação lógica indutiva (ILP) (Zelle & Mooney, 1996; Mooney, 1996; Wermter, Riloff & Scheler, 1996).

A aplicação dos modelos estatísticos para problemas de língua natural, por outro lado, envolve o emprego de técnicas como n-gram, Hidden Markov Models (HMMs), gramáticas livres de contexto probabilísticas (Charniak, 1993, 1997) ou árvores de decisão probabilísticas (Magerman, 1995). A principal meta desses modelos é aprender através de observações dos exemplos e fazer previsões sobre novos exemplos. Os modelos estatísticos já estão bastante consolidados quando aplicados para tal propósito.

Já os modelos conexionistas, até então com resultados não satisfatórios quando aplicados a problemas de língua, contra-atacam na década de 90 com novas propostas, fortemente representadas pelas redes parcialmente recorrentes de Elman (Elman, 1990; Reilly & Sharkey, 1992; Miikkulainen, 1993; Lawrence et al., 1996; St. John & McClelland, 1990).

As redes recorrentes resolvem o problema do tamanho variado das sentenças, uma vez que possuem entrada incremental. Conseguem também generalizar sobre as posições estruturais das palavras na sentença.

Há ainda outras arquiteturas neurais, extensões naturais das recorrentes, como a proposta em Shastri & Ajjanagadde (1993) e em Lane & Henderson (1998). Os autores propõem uma arquitetura que suporta computação simbólica dentro de uma arquitetura neural. Tal arquitetura é capaz de juntar habilidades necessárias para a recuperação das características da língua, como a

combinação de várias fontes de restrição e a habilidade de manipular dinamicamente as representações complexas. Apesar de algumas limitações, o sistema proposto tem se mostrado de grande eficiência.

Em uma das experiências realizadas em (Bonfante, 1997) pudemos comprovar a eficiência da arquitetura das redes recorrentes para problemas da língua portuguesa. Nesses experimentos, as redes neurais foram usadas para detectar problemas quanto ao uso incorreto ou ausência da crase, apresentando resultados satisfatórios, considerando o caráter da não restrição do domínio tratado (em média 70%).

No processamento das línguas naturais, temos que um dos requisitos fundamentais para o bom funcionamento dos sistemas é a eficiência na recuperação da estrutura sintática das sentenças. As pesquisas atuais têm se focado muito na construção de analisadores sintáticos ou *parsers*² eficientes, cuja função principal é o reconhecimento dessas estruturas sintáticas³.

Para a construção de sistemas que utilizam métodos empíricos na construção dos *parsers*, são necessários grandes corpora de sentenças etiquetadas, que são usadas para o treinamento e garantem a precisão. Exemplos desses corpora são o Penn Treebank (Marcus et al., 1993), para o inglês, com palavras marcadas com rótulos sintáticos, árvores de *parsing* e alguns traços semânticos, e um corpus de sentenças em português, hoje com 100.000 palavras, que foi etiquetado com o NILC Tagset (Aluísio, 2000).

Um passo além do processo de *parsing* é a revisão gramatical automática (*checking*). É desejável que as ferramentas de revisão sejam mais do que reconhecedores sintáticos retornando “sim” ou “não” quando solicitadas para verificar a gramaticalidade das sentenças numa determinada língua. Tais ferramentas têm de apontar os erros cometidos e também sugerir possíveis correções para esses erros.

A construção de ferramentas de revisão de textos consiste ainda num grande desafio para os desenvolvedores de sistemas de processamento de língua natural. Apesar de estarem bem estabelecidos os passos para a construção de tais sistemas, a dificuldade surge ao se tentar

² Nesse trabalho, análise sintática e *parser* são tidos como sinônimos. Passamos, daqui para frente, a usar o termo *parser* para analisador sintático e *parsing* para o processo de análise.

³ O *parser* identifica os constituintes maiores formadores da sentença (S), como por exemplo, em ‘O gato sumiu’ temos uma das formas mínimas de formação, que é um sintagma nominal (SN), representado por ‘O gato’ atualizado por constituintes imediatos det=“o” + nome=“gato” e um sintagma verbal (SV), atualizado pelo constituinte imediato v= ‘sumiu’.

construir ferramentas que tratam qualquer tipo de texto, ou seja, que não restringem domínio, estilo ou abrangência sintática.

O processamento de línguas requer uma linguagem bastante rica para a representação do conhecimento, que inclui relações, recursões, e representações estruturais não limitadas.

Assim, a revisão gramatical automática necessita de um *parser* bastante eficiente, pois a recuperação da estrutura sintática das sentenças é requisito fundamental, uma vez que só assim se saberá se existe alguma irregularidade nessa formação. No entanto, também é necessário um sistema de diagnóstico, responsável por detectar o local exato onde o erro ocorreu na sentença, bem como sugerir possíveis correções para esse erro.

Com a grande disseminação dos computadores e dos processadores de textos surgiram, ainda na década de 90, grandes ferramentas de revisão. Dos revisores comerciais atuais encontrados para o tratamento da língua inglesa, destacamos o Grammatik (Reference Software International, 1992) e o Correct Grammar (Writing Tools Group, Inc. 1991). São ferramentas de grande eficiência e bastante consolidadas para a revisão de textos em inglês.

Para o tratamento da língua portuguesa europeia, encontramos os revisores comerciais Flip (FLiP 97). No entanto, não é divulgada a metodologia usada na construção. Encontramos também um revisor proposto em (Baião, 1994).

Existem também no mercado as ferramentas específicas para o tratamento da língua portuguesa do Brasil, como o DTS (DTS Software Ltda, 1995), a Gramática Eletrônica (Lexicon Informática, 1999) e o ReGra (Martins et al., 1998), comercializado nos produtos RLP Itaotec (Itaotec softwares, 1999) e Microsoft Office 2000. Encontramos também, o revisor SOS (Moura, 1996; Lins et al., 1999).

O revisor ReGra, construído nos moldes racionalistas pela equipe do NILC (Núcleo Interinstitucional de Lingüística Computacional)⁴, apresenta um padrão de eficiência e robustez destacável para o tratamento da língua portuguesa falada no Brasil. Construído na forma de regras compiladas, o revisor conta hoje com mais de 1000 regras sintáticas (*parsing + checking*).

O desempenho alcançado deve-se em grande parte ao fato de que, para a construção e validação das regras, são utilizados os padrões encontrados em um corpus compilado pela própria equipe do projeto. Tal corpus contém mais de 39 milhões de palavras não etiquetadas,

⁴ Num convênio com a empresa Itaotec/Philco, desde 1993.

construído de textos reais, retirados de jornais, revistas, teses e redações de vestibular, entre outros.

ReGra conta também com a existência de um léxico com mais de 1.5 milhão de entradas, incluindo nomes próprios, acrônimos e abreviações (Nunes et al., 1996), com informações morfosintáticas sobre essas entradas.

Contudo, existem ainda limitações no revisor. Um dos problemas é a grande quantidade de regras do qual é composto, número esse que aumenta a cada dia. Aumentando o número de regras, aumenta também a dificuldade de gerenciamento, uma vez que a inserção de novas regras pode gerar conflitos com as já existentes. Isso torna também o trabalho de gerenciamento e manutenção bastante exaustivo.

Outro problema que surge é proveniente da própria estrutura da língua. A rica variedade de construções sintáticas e a grande quantidade de conhecimento léxico, sintático, semântico e pragmático a ser incorporado no revisor, formam uma barreira de difícil transposição para seus projetistas. O revisor ainda tem limitações devido a essa dificuldade na representação do conhecimento.

2. Revisão Gramatical Automática de Português

A construção de revisores gramaticais ainda consiste num grande desafio para os seus projetistas. A grande variedade de construções lingüísticas e a enorme quantidade de conhecimento a ser incorporado nesse tipo de sistema impõem uma barreira de difícil transposição para seu desenvolvimento.

Soma-se a isso o fato de que, para se ter um revisor eficiente, alguns recursos caros de se desenvolver devem estar presentes, como um *parser* robusto e um léxico com informações sobre as classes gramaticais das palavras e outras de natureza morfosintática.

A pluricategorização gramatical das palavras (por exemplo, a palavra “casa” pode ser tanto um substantivo feminino quanto uma forma do verbo transitivo direto “casar”) é uma das principais fontes de problema para um *parser* e, logo, para um revisor. Um tratamento de desambigüização de palavras é imprescindível, caso contrário, o número de sentenças para as quais o revisor se omitirá torna-se muito elevado.

Levando-se em conta somente a revisão gramatical, existem duas abordagens principais utilizadas para a construção de sistemas de detecção automática de erros: a abordagem localista ou *error-driven* e a abordagem genérica, que requer análise sintática [Martins et al., 1998]. Os revisores gramaticais localistas são direcionados à localização e correção imediata dos problemas mais frequentes no uso da língua. Tais sistemas não fazem um pré-processamento das sentenças e operam sobre seqüências de regras especialmente categorizadas e ordenadas. Eles são rápidos e eficientes, uma vez que sua função é procurar por problemas isolados em combinações curtas de palavras. No entanto, só conseguem tratar itens adjacentes e os erros detectados limitam-se a estruturas contínuas da língua, como sintagmas nominais na sua mais simples configuração.

Já a abordagem geral conta com meios para o tratamento de estruturas mais complexas, uma vez que faz uma análise sintática da sentença. O processo de *parsing* se inicia com a atribuição de valores sintáticos aos itens lexicais. Isso possibilita a identificação de relações de dependência entre as palavras nas sentenças e conseqüentemente, a checagem das inconsistências gramaticais.

Essa identificação das dependências entre as palavras da sentença é feita pelo *parser*, com a ajuda do léxico. Com a estrutura montada, por exemplo, na forma de uma árvore sintática, o revisor analisa sua validade de acordo com as normas de escrita da língua, a gramática, que pode estar representada, por exemplo, na forma de regras.

Essa representação, quando feita de forma manual, acaba por gerar muitos problemas oriundos da grande interdependência existente entre as regras. Isso torna bastante difícil o processo de gerenciamento dessas regras e também o de ajuste na inserção de novas regras.

A seguir, apresentamos em linhas gerais as características das ferramentas de revisão gramatical encontradas no mercado, aqui representado pelo revisor ReGra. Descrevemos também o que é encontrado em pesquisa acadêmica, representado por uma proposta de uma ferramenta de revisão gramatical para o português europeu, e por uma proposta de revisão para o português brasileiro. Delineamos ainda, as características de um protótipo empírico de revisão, baseado em redes neurais, que iniciamos em (Bonfante, 1997).

2.1. Modelo de Baião (1994) – Português europeu

Baião propõe um modelo de revisão gramatical composto por três módulos: um *parser*, um formulador de hipóteses, e um corretor.

O revisor proposto detecta e corrige vários tipos de erros morfossintáticos. Dada uma cadeia de caracteres, o sistema tenta classificá-la como uma frase gramaticalmente correta ou sugere correções baseadas na análise sintática que realizou.

O sistema, construído em Prolog, utiliza um *chart-parser* (Gazdar & Mellish, 1989) para a pesquisa dos constituintes e em caso de falha, interage com um formulador de hipóteses, que permite a correção da mesma. A gramática utilizada, de movimentação e ancoragem (Lopes, 1992), permite a identificação dos constituintes fora da ordem canônica.

Os *chart-parsers* são concebidos para gramáticas livres de contexto e são caracterizados pela existência de um grafo para armazenamento das sub-análises já realizadas. Isso evita a repetição da análise, e permite que, em caso de falha, possa ser detectada com maior facilidade sua causa, auxiliando, assim, o processo de reparação.

No sistema proposto, o *chart-parser* sofreu uma adaptação (Paulo, 1993) para trabalhar com as gramáticas de movimentação e ancoragem. Essa adaptação foi necessária para impedir a falha do analisador face à existência de erros de concordância.

Exemplo de uma regra: $sn(N, G, sn(X, Y)) \rightarrow$

[**ilha** det(N1, G1, X), **ilha** nomep(N2, G2, Y),
{concorda(N1, N2, N), concorda(G1, G2, G)}],

que determina que um sintagma nominal (sn) poderá se constituído por um determinante X e um nome próprio Y, devendo existir concordância em gênero e número entre seus constituintes. O operador **ilha** é um operador tipo barreira, que garante que o material lingüístico descrito pelos constituintes não terminais não se movimente (dado que numa gramática de movimentação e ancoragem há a possibilidade da movimentação de material lingüístico a distâncias não pré-fixadas).

Para uma palavra desconhecida, o módulo invocará um corretor ortográfico bastante simples que tentará propor correções às palavras ou então a sua admissão como palavra desconhecida.

O sistema tem como referencial uma frase, representada através de uma lista de átomos Prolog, e usa como base um conjunto de regras sintáticas e um léxico.

Usa duas abordagens diferentes variando conforme existência ou não das palavras no léxico.

A detecção e correção dos erros ocorrem da seguinte forma:

- Inicialmente cada palavra é categorizada de acordo com um léxico ou considerada desconhecida;
- O *chart-parser* é solicitado para analisar a frase;
- Se o analisador falhar, desencadeia-se um formulador de hipóteses que, utilizando técnicas de relaxamento, tentará formular hipóteses que permitam superar os erros tratados, ou então as palavras que desconhece;
- Finalmente, desencadeia-se um detector e corretor de erros que, face às análises obtidas para o símbolo inicial da gramática, informará o usuário sobre as incorreções encontradas e as alternativas possíveis de correção;

Quanto ao tipo de erro, foram considerados casos em que o sistema encontra a palavra no léxico, bem como casos em que elas não existem:

- Palavras desconhecidas
 - Palavras mal escritas: são considerados 4 tipos de erros:
 - Substituição de uma letra por outra;
 - Omissão de uma letra;
 - Existência de uma letra a mais;
 - Troca de letras consecutivas.
 - Inexistência da palavra no léxico (considerado como neologismo)
- Palavras existentes: são tratados os seguintes erros:
 - Palavra a mais, seja por duplicação ou não : **João João foi à praia*, **João foi ia passear*
 - Questões de concordância em pessoa, gênero e número: **Eu comeu uma maçã* / **A menino comprou uma bola*. Com exceção das questões de concordância relativas a construções predicativas no que diz respeito ao predicativo do sujeito: **os livros são linda*;

- Palavras com categoria ou forma diferente da requerida: * *João deu o livro há Maria*, **O livro foi dar pelo João à Maria*;
- Alguns casos de palavras omissas, onde o formulador de hipóteses propõe a categoria da palavra faltante.

O formulador de hipóteses gera todas as possíveis correções para o erro encontrado. Isso ocorre devido à dificuldade de escolha por uma dada hipótese de correção.

Ex: em 'o prgrama terminou':

O *parser* atribui arcos para as palavras e tentará reuni-los de acordo com a gramática descrita. Quando necessário, usa técnicas de relaxamento para que a análise prossiga. Assim a sentença fica categorizada com vértices na forma: '1 o 2 prgrama 3 terminou 4', onde um arco do vértice 1 para o 2 representa uma categoria (no caso o determinante 'o'); o arco de 2 para 3, outra categoria, e assim por diante. Um arco se torna completo quando existe a análise completa dos constituintes. Ao analisar temos:

- A análise falha devido à inexistência da palavra prgrama
- A formulação de hipóteses repara o erro considerando as hipóteses de prgrama poder ser categorizado como **nome próprio** ou **nome comum**
- O desencadeamento posterior da análise só deve terminar após ter conseguido obter os dois arcos completos referentes ao símbolo inicial que corresponderão, respectivamente, às hipóteses de prgrama poder ser considerado um **nome próprio** ou um **nome comum**.
- Após a detecção e correção de erros, o usuário é informado que, entre outras possibilidades de correção, prgrama poderá ser corrigido por **programa**.

Quando a pesquisa termina, é retornado o vértice correspondente à última análise efetuada. Isso permite que, em caso de falha na análise dos constituintes se possa detectar quais os arcos ativos causadores da falha que serão objeto de reparação posterior.

No caso do exemplo, o analisador falhará na análise da palavra desconhecida prgrama e retornará o vértice 2 que permitirá a formulação de hipóteses sabendo que a falha foi detectada no vértice 2. Os arcos ativos necessários para o prosseguimento da análise têm como vértice final 2 e como primeiras categorias em resto para continuar a análise **nome comum** e **nome próprio**.

O corretor de erros recorre ao grafo obtido, ao vértice onde terminou a pesquisa de constituintes e à lista dos arcos referentes a palavras desconhecidas para construir uma lista de erros.

Por exemplo: em ‘O João’, como não existe arco completo, o formulador informará que existe a necessidade de pelo menos um verbo, e o corretor de erros sugerirá tal correção.

No caso de sentenças que estão fora da ordem como ‘queimou o livro’, o revisor se comporta da seguinte forma: quando se inicia a análise, o fato de o verbo aparecer antes quebra a sequência inicial. O verbo, então, é guardado em dispositivos chamados de canais, permitidos pela gramática. A análise prossegue e identifica o sintagma nominal ‘o livro’. De acordo com a gramática, há a necessidade de pelo menos um verbo depois dessa sequência. Assim, o *parser* procura nos canais se existe algo armazenado. Achando o verbo, este é então ancorado na sua posição canônica.

Trata-se, assim, de um sistema que trabalha num domínio bastante restrito, uma vez que conta com uma gramática de cerca de 45 regras.

O léxico também é bastante reduzido e incompleto. O problema é amenizado com o tratamento de palavras bem escritas, mas que o léxico desconhece.

Uma desvantagem é o tempo ocupado em pesquisas com bases de hipóteses formuladas que poderão redundar em fracasso. Assim, limitou-se o número de hipóteses formuladas usando estratégias como a preferência pelo erro devido a palavras a mais ao invés de omissões.

2.2. O revisor SOS (Lins et al., 1999) – Português brasileiro

O sistema SOS, proposto pelos autores, integra um projeto mais amplo, que engloba desde o reconhecimento de documentos históricos com o uso de OCR (Optical Character Recognizers), até sua correção, processamento e armazenamento.

SOS trata-se de um revisor ortossintático para a língua portuguesa falada no Brasil. O processo de revisão ortográfica foi desenvolvido baseado em um autômato finito determinístico, com o auxílio da ferramenta LEX [Sun-Microsystems Manual-Lex]. As regras ortográficas reconhecem palavras de acordo com a acentuação padrão do Latex, e também revisam textos escritos em HTML e mensagens eletrônicas (e-mail).

O corretor ortográfico trabalha principalmente sobre os erros mais frequentes cometidos durante a digitação, como: falta de acentuação ou uso incorreto de acentos nas palavras; letras dobradas nas palavras e erros mais comuns do Português. Ainda corrige outros erros mecânicos como verificação do uso de delimitadores não balanceados (aspas, chaves, colchetes e parênteses).

O processo de revisão sintática consiste em, primeiramente, transformar seqüências lineares de palavras em estruturas hierárquicas de relacionamentos, utilizando uma gramática livre de contexto estendida (Allen, 1995). Uma vez identificadas essas estruturas, tornam-se automáticas as regras de verificação da concordância nominal e verbal da frase. O analisador sintático foi construído com o auxílio da ferramenta YACC (Sun-Microsystems Manual-Yacc).

Dentre os erros corrigidos estão: alguns casos de uso de vírgula, alguns casos de emprego de crase e alguns casos de colocação pronominal (próclise, ênclise e mesóclise).

Com relação a problemas de concordância, tanto verbal quanto nominal, SOS limita-se a analisar somente frases escritas na ordem direta, isto é, frases constituídas de sujeito seguido de predicado, ou apenas um deles.

A seguir, apresentamos alguns exemplos:

Exemplo1:

- Frase original: *A leão do garotos morreram.
- Frase corrigida: O leão dos garotos morreu.

Nesse caso “A leão do garoto” representa o primeiro sintagma nominal do sujeito e o substantivo “leão” representa seu núcleo. Daí, a sugestão de do artigo “A” para “O” e do verbo “morreram” para “morreu”, uma vez que o núcleo do sujeito (palavra principal) e o verbo devem concordar entre si. O mesmo ocorre no segundo sintagma nominal “o garotos” onde ocorre a sugestão de troca de “o” para “os”.

Exemplo2: com orações subordinadas.

- Frase original: * O livros que o aluno que conhece nossa irma compramos era de matematica.
- Frase corrigida: Os livros que o aluno que conhece nossa irmã comprou eram de matemática.

Nesse tipo de frase, é utilizado um *buffer* para armazenar os diversos atributos (gênero, número e pessoa) do núcleo do sujeito das diversas frases. Daí, “comprou” concorda com

“aluno” e “eram” concorda com “livros”. As demais concordâncias dão-se de forma semelhante ao exemplo anterior.

Há também o tratamento de algumas frases negativas e interrogativas.

O problema da ambigüidade léxica, como a homonímia⁵ e a pluricategorização de uma palavra⁶, é tratado da seguinte forma: todas as classes gramaticais de uma palavra são armazenadas no dicionário. O *parser* recebe todas as classes e ativa uma função para selecioná-las. Essa função verifica quais estruturas sintáticas já foram reconhecidas pelo *parser* e, então, seleciona a mais adequada.

Alguns exemplos de conflitos são, por exemplo: o problema da ambigüidade da palavra “a”, que pode ser artigo (“A biblioteca está fechada”), preposição (“Ele cheira a vinho”) ou pronome (“Eu a convidei para ir ao cinema”); a substantivação de palavras através da inserção de um artigo (“A menina tem um olhar triste”); ou a transitividade de alguns verbos, quando esses possuem mais de uma regência, como por exemplo, o verbo “casar” pode ser intransitivo (“Ontem ele se casou”) ou transitivo indireto (“Ele se casou com uma bela garota”).

Segundo os autores, por estar em sua primeira versão, SOS apresenta resultados satisfatórios, considerando-se a complexidade da língua. Um tratamento de ambigüidade relacionada à classe gramatical de algumas palavras, como: “um” (numeral/artigo), “são” (verbo/adjetivo), “casa” (substantivo/verbo), etc., ainda deve ser feito. Devem ser também estendidas regras para o tratamento de outros tipos de frases, além de outros pontos a serem melhorados.

2.3. O revisor gramatical ReGra

O revisor gramatical ReGra nasceu de uma parceria universidade/empresa⁷, num projeto que teve como produto uma primeira versão comercializada em 1995. Conta, desde então, com uma

⁵ Segundo o Aurélio, um dos sentidos de uma palavra homônima é quando se pronuncia e se escreve da mesma forma que outra, mas cujo significado é diverso (ex: falácia = qualidade de falaz, e falácia = falatório).

⁶ Por exemplo, o verbete “trabalho” pode ser tanto um substantivo masculino singular: “O trabalho enobrece o homem”, quanto como verbo intransitivo: “Eu trabalho muito” e transitivo indireto: “Eu trabalho para o desenvolvimento da nação”.

⁷ Núcleo Interinstitucional de Linguística Computacional, Departamento de Computação, USP-São Carlos e a empresa Itaotec/Philco.

equipe de desenvolvimento composta por lingüistas e informatas, e por coordenadores de três universidades.

Devido ao fato de ser uma ferramenta bastante complexa, descreveremos aqui algumas características que consideramos relevantes para um entendimento básico.

2.3.1. Arquitetura do Revisor

Em sua versão atual, o ReGra é composto por quatro módulos, que são responsáveis pelas funções léxica, mecânica, estatística e lingüística por ele desempenhadas. O módulo mecânico é responsável pela identificação de problemas de digitação como parênteses desbalanceados, falta de pontuação no final do período, etc. O módulo léxico trata os erros ortográficos e é auxiliado por um léxico de mais de 1.5 milhão de palavras (Nunes et al., 1996). O módulo lingüístico trata das imperfeições gramaticais.

Dessa forma, o revisor trata erros que vão desde a escrita incorreta das palavras e erros cometidos durante a digitação até imperfeições quanto à estruturação das sentenças de acordo com as normas de escrita da língua portuguesa. Produz ainda medidas estatísticas do texto que está tratando.

A seguir, descrevemos brevemente cada um desses módulos, enfocando o módulo lingüístico, com o qual teremos mais contato. A Figura 2.1 ilustra como está a organização do revisor em sua versão 2000, onde as figuras retangulares representam os módulos atuantes (com exceção dos módulos de análise semântica e de análise estatística, que ainda estão em projeto de implantação), e as demais, as bases de conhecimento disponíveis. O projeto do módulo semântico está em andamento, ancorado pelo projeto TraSem (Martins et al., 1999). O projeto do módulo estatístico⁸ consiste no uso de técnicas estatísticas, tal como descritas nesse trabalho, por exemplo, em (Charniak, 1997), para ajudar no processo de desambigüização categorial das palavras.

⁸ Tal módulo difere do módulo estatístico, que calcula o Índice de Legibilidade dos textos.

2.3.1.1. Módulo Léxico

O analisador léxico é responsável pela atribuição categorial das palavras formadoras do parágrafo, que é a estrutura básica de análise da ferramenta. A partir de uma busca à base lexical, recupera as informações, que são utilizadas durante o processo de revisão gramatical.

2.3.1.2. Módulo Estatístico

O módulo estatístico produz uma análise do texto sendo analisado, de uma forma que o autor tenha em mãos medidas estatísticas do mesmo, tal como número total de parágrafos, sentenças, palavras, caracteres, letras e sílabas, o número médio de sentenças por parágrafo, de palavras por sentença e de letras e sílabas por palavra.

A partir da coleta dos dados, o módulo produz índice de legibilidade para o texto, que indica o nível de dificuldade deste, segundo alguns parâmetros, como descritos na Tabela 2.1.

Tabela 2.1. Tabela de Legibilidade obtida dos textos.

Pontuação do Texto	Nível de Dificuldade
75 a 100	Muito Fácil
50 a 75	Fácil
25 a 50	Difícil
0 a 25	Muito Difícil

E, assim, de acordo com a classificação proposta, textos muito fáceis são indicados para leitores com nível de escolaridade até quarta série primária; textos fáceis são indicados para leitores com escolaridade até a oitava série; textos difíceis são indicados para leitores de nível colegial e/ou universitário e textos muito difíceis são textos acadêmicos de áreas específicas.

2.3.1.3. Módulo Mecânico

O módulo mecânico é responsável pela análise do parágrafo procurando por erros cometidos durante a digitação, como o aparecimento de delimitadores desbalanceados, a escrita correta dos

números, palavras ou símbolos que aparecem repetidos em seqüências dentro de um parágrafo, a capitalização correta, o uso correto da pontuação e uso das aspas, entre outros.

2.3.1.4. Módulo Lingüístico

O módulo lingüístico (Figura 2.1) é composto basicamente por três sub-módulos: o de revisão pontual, o de análise sintática (*parser*) e o de revisão gramatical.

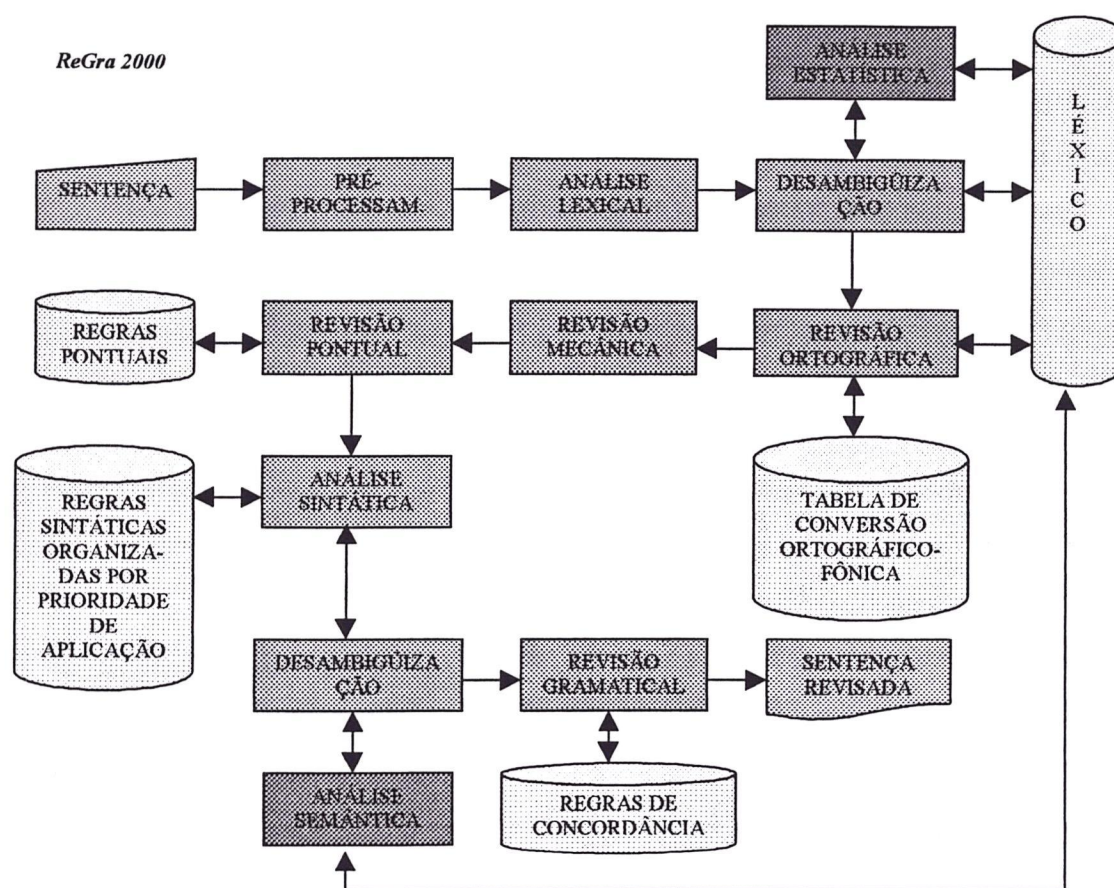


Figura 2.1. Arquitetura básica do ReGra-2000

O revisor ReGra adota uma abordagem híbrida que integra regras heurísticas pontuais para a detecção de erros freqüentes, que são sensíveis ao contexto, mas também facilmente identificados através de padrões pré-definidos, e também um *parser* robusto, que trata principalmente problemas de concordância nominal e verbal. O *parser* utiliza uma gramática Livre de Contexto com técnicas de restrição baseadas em relaxamento (*relaxation constraints-based techniques*) para permitir o casamento de sentenças agramaticais. O conjunto de regras é

ordenado de acordo com a frequência de ocorrência dos tipos de erros. Erros mais frequentes são priorizados na verificação. As regras mais frequentes correspondem àquelas construções sintáticas mais usadas por usuários de textos. São cerca de 1000 regras sintáticas atualmente contempladas pela ferramenta.

O módulo lingüístico é o responsável por todo o tratamento gramatical feito pela ferramenta. Este recebe como entrada um parágrafo livre de erros mecânicos e ortográficos, dividido em unidades menores cuja base de formação é através de pontuações e conjunções. A partir daí as regras de correção, construídas na forma de Redes de Transição Aumentadas (do inglês *Augmented Transition Networks*) (Woods, 1970), procuram por erros locais, num processo de checagem pontual.

As regras pontuais tratam problemas de uso da crase, posicionamento de pronomes, uso de prefixos, gêneros das palavras, uso de preposições em expressões, uso de *onde* e *aonde*, uso da forma particípio de alguns verbos, uso de *mau* e *mal*, uso de *há* e *a*, uso do verbo *haver* e suas flexões, inadequação léxica de algumas palavras em expressões (ex: o uso de expressões como “mal e parcamente”, que muitas vezes é usada como “mal e porcamente”), pleonasmos (“subir para cima”), utilização das contrações (como o uso da forma correta: “O fato de o sistema ser adequado...” ao invés de “O fato do sistema ser adequado”), concordâncias de *que* e *quem*, verificação da formação *verbo + se* e verificação da pontuação em expressões.

Após a categorização de todas as palavras, o *parser* parte para o processo de identificação da estrutura sintática da sentença. Uma vez que os desvios mais comuns da norma são corrigidos pelas regras pontuais, os mais complexos são checados, como relações de regência e concordância, que dependem de funções sintáticas.

Assim, o processamento do ReGra, que tem como característica ser top-down e left-to-right, segue 4 passos principais:

1. Atribui as classes gramaticais de cada palavra, sendo que as palavras que possuem mais de uma classe gramatical têm essas classes ordenadas pela ordem que aparecem no léxico. A seguir, é acionado o processo de desambigüização que determina as classes gramaticais das palavras.
2. Trata os erros mecânicos de acordo com os padrões encontrados na sentença;
3. Aplica as regras pontuais descritas anteriormente;

4. Realiza a revisão gramatical, que consiste do *parsing* e do *checking*, de forma intercalada, isto é, o *checking* é disparado imediatamente após o reconhecimento de um delimitador de sentença (algumas conjunções, algumas locuções adverbiais, marcadores de discurso e alguns casos de pontuação) durante o processo de *parsing* (*Interleaved Parsing and Checking*).

O *parsing* é feito utilizando uma tabela de predições de possibilidades sintáticas (na forma de ATNs). Se as regras corretas não casarem, o *parser* procura um casamento com estruturas supostamente incorretas (usando técnicas de relaxamento). Por exemplo, casando com um falso objeto direto, que ocorre quando o verbo é transitivo direto e o usuário coloca um objeto indireto depois dele. Já o *checking* consiste na aplicação de regras de concordância nominal e verbal entre os termos da sentença.

2.3.2. Recursos disponíveis para a ferramenta

É interessante destacarmos os recursos do ReGra, pois eles também deverão ser utilizados no trabalho aqui proposto.

2.3.2.1. O léxico

O léxico usado pelo ReGra contém, atualmente, mais de 1.5 milhão de formas não analisadas⁹, incluindo nomes, acrônimos e abreviaturas (Nunes et al., 1996).

Devido ao fato de o revisor tratar qualquer tipo de texto produzido por nativos da língua portuguesa do Brasil, o léxico também não pode ser de domínio restrito. A versão atual classifica as palavras de acordo com dez classes gramaticais, a saber: substantivo, adjetivo, advérbio, preposição, interjeição, conjunção, numeral, pronome, verbo, artigo e algumas subclassificações como (nome próprio, nome comum, etc.).

Todos os lexemas que fazem parte do léxico são providos com informações gramaticais de forma a atender as exigências morfológicas e sintáticas de cada palavra quando requisitadas pela regras gramaticais.

⁹ Por exemplo, são colocadas todas as formas dos verbos.

Apresentamos abaixo, alguns exemplos de entradas lexicais, onde por exemplo, um substantivo, tem informação de gênero: masculino (SM) ou feminino (SF), de número: singular (SI), e a canônica. Já um verbo (V), por exemplo tem informação de transitividade: transitivo direto (TD), intransitivo (INT), pronominal (PRONOM), etc; pessoa e tempo: presente indicativo (PRES.IND), terceira pessoa (ELE), etc.; forma: não infinitiva (N); regência (a, com), e a canônica.

menino = <SM. SI. ?. []. [menino]>

casa = <SF. SI. ?. []. [casa]

casa = V.[TD.TI.INT.PRONOM.][PRES.IND.ELE.IMP.AFIR.TU.]N.[a,com][casar]

As informações de palavras que possuem mais de uma classe gramatical foram preenchidas de acordo com uma indicação hierárquica quanto à maior frequência de uso, obtida da análise no corpus.

2.3.2.2. O corpus

Para atender às necessidades da construção do revisor gramatical, foi feita uma compilação de textos, hoje com mais de 39 milhões de palavras. Esse corpus é composto de textos completos extraídos de livros, jornais, revistas e outras publicações.

Esse banco de textos foi criado com três objetivos principais: a) de ser fonte de levantamento dos erros mais frequentes dos usuários da língua portuguesa de nível médio, com escolaridade de segundo grau completo; b) de ser fonte de levantamento dos padrões sintáticos mais frequentes na língua escrita; e c) de servir aos testes da ferramenta.

Para atender a esses objetivos, o corpus foi dividido em três classes: textos corrigidos, textos semi-corrigidos e textos não corrigidos. A classe de textos corrigidos, obtidos de jornais, revistas e livros, e a classe de textos semi-corrigidos, obtidos de teses, dissertações e projetos científicos, são usadas principalmente com o propósito de determinar falsos erros do revisor, bem como de levantar padrões sintáticos possíveis. A terceira classe, composta de textos produzidos por alunos de graduação, secretárias e vestibulandos, é usada para testar o desempenho da ferramenta, bem como para obtenção dos erros mais comuns cometidos por usuários da língua.

Para o léxico, o corpus tem a função de licenciar a inserção das palavras que não estão dicionarizadas, provendo assim, um enriquecimento da base lexical.

2.3.3. Limitações

O aprimoramento do ReGra tem como objetivos principais: a) a redução do número de omissões da ferramenta quando a correção é necessária – falsos acertos; b) a redução do número de intervenções desnecessárias – falsos erros; e c) atualização progressiva das possíveis regras de revisão gramatical para admitir a incorporação crescente das construções gramaticais da língua portuguesa (Pardo & Rino, 2000).

Uma das grandes limitações (de cunho lingüístico) do revisor consiste na dificuldade de tratar a ambigüidade categorial, uma vez que a determinação correta das classes gramaticais é fundamental no processo de revisão. Entra aí o problema do tratamento semântico, que tem sido pouco contemplado pela ferramenta. Somente com o auxílio de informações semânticas é possível decidir criteriosamente sobre a classe gramatical das palavras. Em (Martins et al., 1999), os autores levantam esse problema, exemplificado pela resolução da ambigüidade categorial da forma “se”. Encontra-se ainda em andamento, o projeto de um *tagger* (etiquetador) morfossintático (Aluísio, 2000), cujo intuito é exatamente o de minimizar essas ambigüidades, uma vez que leva em conta o contexto ao qual se encontram inseridas as palavras, bem como medidas estatísticas das freqüências nas quais ocorrem, para atribuir a classe gramatical das palavras.

O problema da pluricategorização das palavras é um dos fatores que mais contribuem para o insucesso do *parsing*. Os métodos empíricos para a construção de *parsing*, tal como veremos na próxima seção, possuem mecanismos que minimizam tal problema, uma vez que seu princípio de funcionamento consiste na aprendizagem e identificação de regularidades, bem como freqüências, dos padrões que lhes são apresentados através de *corpus* de treinamento.

Uma outra limitação, de cunho implementacional, que torna o funcionamento do ReGra bastante complexo, é o fato de o módulo lingüístico (o núcleo do sistema) gerenciar o *parsing* e o *checking* utilizando a técnica de intercalação, cuja principal característica se manifesta da seguinte forma: se durante a análise sintática de uma sentença aparecer algum disparador de verificação gramatical, o controle do sistema é desviado para essa verificação antes mesmo que a análise sintática da sentença esteja completa. Esta característica traz como principal desvantagem

a impossibilidade de separação entre *parsing* e *checking*, tornando sua manutenção e documentação tarefas árduas que implicam dificuldades maiores para a atualização do sistema. Essa característica proveniente, em parte, da estruturação na forma de regras, deve ser minimizada com o uso de técnicas alternativas como as que investigamos aqui.

Somam-se a isso problemas ainda a serem resolvidos com relação à concordância nominal e verbal, ao uso de crase e à pontuação, entre outros. Nas Tabelas (2.2. a 2.7.) abaixo, apresentamos algumas frases, exemplificando esses problemas, em casos onde o revisor cometeu falsos erros, omissões (falsos acertos), erros de pontuação, erros quanto ao emprego da crase, etc.

Tabela 2.2. Falsos Erros: Concordância Nominal.

Daí o maior número de mulheres eliminadas do sistema protetor da grande família, obrigadas a procurar **sozinhas** seu sustento, pela dificuldade em se conseguir o almejado casamento, criando a nova categoria social da "solteirona".

Erro: pede a concordância entre 'sozinhas' e 'seu'

Sugestão: 'sozinhas suas'

M.I.: concordância nominal (os apostos devem concordar com as palavras a que se apoem).

Com as descobertas de Pasteur, foi criada a **Bacteriologia**, e o meio científico da época vislumbrou esperançoso um futuro próximo livre de germes e doenças.

Erro: pede a concordância entre 'criada', 'Bacteriologia' e 'científico'

Sugestão: 'criados a Bacteriologia, e o meio científico da época'

M.I.: concordância nominal

As aspas, itálico e negrito presentes refletem sinais de pontuação ou destaque contidos no original em inglês.

Erro: pede a concordância entre 'itálico' e 'negrito' e 'aspas'

Sugestão: 'As aspas, itálicas e negritas presentes'

M.I.: concordância nominal (os adjuntos adnominais concordam com o nome)

É erro supor que a **ausência de reação seja sinônimo** de que a vacina "não pegou".

Erro: pede a concordância entre 'sinônimo' e 'ausência'

Sugestão: 'a ausência... sinônima' / 'as ausências... sinônimas'

M.I.: concordância nominal

Tabela 2.3. Falsos Erros: Concordância Verbal.

Trata-se de um aglomerado de **proteínas que envolve** um tipo de ácido nucléico, DNA ou RNA.

Erro: pede a concordância entre 'proteínas' e 'envolve', devido à presença do 'que' como elemento posterior a 'proteínas'.

Sugestão: 'proteínas que envolvem'

M.I.: concordância verbal

A meia légua dali, alguns paisanos passavam bois ou ovelhas para o Uruguai ou no sentido inverso,

conforme o preço da lã ou da carne.

Erro: pede a concordância entre 'passavam' e 'A meia légua'

Sugestão: 'A meia légua dali... passava' / 'As meias léguas dali... passavam'

M.I.: concordância verbal

Contrabandista desde o berço, muito cedo me desinteressei por gado, mas nem por isso abandonei este vício de fronteirista.

Erro: pede a concordância entre 'desinteressei' e 'contrabandista'

Sugestão: 'Contrabandista... me desinteressou' / 'Contrabandistas... me desinteressaram'

M.I.: concordância verbal

Algumas coisas eu perdi e outras apaguei sem querer.

Erro: pede a concordância entre 'apaguei' e 'outras'

Sugestão: 'outra apagou' / 'outras apagaram'

M.I.: concordância verbal

Ambas são o resultado da associação dos elementos que compõem o quadro físico, já mencionados.

Erro: pede a concordância entre 'são', 'resultado'

Sugestão: 'é o resultado'

M.I.: concordância verbal (o verbo 'ser' prefere conc. c/ o termo mais determinado)

Tabela 2.4. Emprego da Crase.

Aninhado sobre um cobertor do Exército que, dobrado sobre o chão, fazia as vezes de cama, Caetano avistou o inseto passeando pela parede iluminada apenas pela luz que entrava do corredor.

Erro: pede a crase

Sugestão: 'às'

M.I.: emprego da crase (locuções conjuncionais)

Habitualmente intenso nas suas descrições, acredita que deva ter o equivalente a cerca de 500 páginas.

Erro: pede a crase (entende 'cerca' como objeto indireto de 'ter')

Sugestão: 'à'

M.I.: emprego da crase (os objetos indiretos são craseados)

Deixa-se de lado a importância de se ensinar a língua através de textos, entendendo-se TEXTO como uma unidade que se estrutura internamente por processos de coesão e coerência.

Erro: pede a crase (os verbos pronominais devem vir preposicionados)

Sugestão: 'à'

A base desse estado de coisas foi a burguesia, cujas condições de bem-estar econômico e formação cultural foram favorecidas pelo regime do presidente Battle y Ordonez, sobretudo depois de 1911.

Erro: pede a crase

Sugestão: 'à'

M.I.: emprego da crase (adj. de lugar junto ao verbo "ir")

A exposição vai apresentar um lado inédito de Brecheret, conta a diretora do IEB, professora e crítica de arte Marta Rossetti Batista.

Erro: pede a crase

Sugestão: 'à'

M.I.: emprego da crase

Tabela 2.5. Falsos Erros: Pontuação.

Filho do campo, me criei entre contrabandistas e muitas vezes cevei um mate para um guarda aduaneiro vindo da cidade. <i>Erro:</i> pede a ausência da vírgula <i>Sugestão:</i> ‘me criei’ <i>M.I.:</i> pontuação (não se separa sujeito de verbo)
Seu irmão de quatro anos, Tales, permanece internado em estado grave no hospital Montreal, em Osasco. <i>Erro:</i> pede a ausência da vírgula <i>Sugestão:</i> Tales <i>M.I.:</i> pontuação (não se separa com vírgula sujeito e verbo)

Tabela 2.6. Confusão entre “há” e “a”.

Desde a invenção do telescópio por Galileu, há 400 anos, nada de parecido foi observado nas vizinhanças da Terra. <i>Erro:</i> pede a palavra 'a' em vez de 'há' <i>Sugestão:</i> ‘a’ <i>M.I.:</i> confusão entre 'há' e 'a'

Tabela 2.7. Omissões

Nos encontramos em face de uma rebelião. Se a rebelião mata homens, a revolução mata homens e princípios. *Colocação pronominal
A importância paradigmática das imagens dos "Desastres" pode ser comprovada através das rimas entre essas gravuras e uma gravuras de Pablo Picasso, entre outras. *Concordância nominal
Qualquer estudo dos processos atuantes na crosta inferior e seu desenvolvimento através do tempo, devem iniciar-se na fonte de dados mais fidedigna e acessível - afloramentos de terrenos metamórficos de alto grau. *Pontuação
E finalmente, os poros da madeira retém pintura, laca, verniz e muitos outros tipos de acabamento. *Concordância verbal
Para onde iremos? *As palavras 'onde' e 'aonde'

2.4. Um protótipo connexionista de revisão gramatical do português brasileiro (Bonfante, 1997).

Nesse trabalho, bem como em [Bonfante & Nunes, 1999], tivemos nosso primeiro contato com os métodos empíricos e seu comportamento, não apenas como analisadores sintáticos, mas como reconhecedores de padrões em segmentos de sentenças.

Investigamos o comportamento de dois modelos connexionistas (uma arquitetura backpropagation e uma arquitetura recorrente de Elman) quando aplicados na identificação de problemas na boa-formação de sentenças em língua portuguesa. Como estudo de caso, escolhemos o tratamento do fenômeno da crase, especificamente, com relação à contração que existe entre o artigo “a” e a preposição “a”.

Para o treinamento das redes neurais, foram compilados cerca de 1500 segmentos de sentenças contendo a ocorrência da crase. Esses exemplos foram retirados de forma aleatória do corpus do NILC, inclusive de textos não corrigidos. A partir daí, foi montado o conjunto de treinamento, separando-se os exemplos de uso correto dos exemplos de uso incorreto. Foram gerados, ainda, exemplos de ausência de crase, uma vez que tínhamos como propósito além da detecção de uso incorreto, a detecção de ausência.

Os exemplos receberam as suas respectivas classes gramaticais (primeira ocorrência no léxico), e a partir daí as redes foram treinadas, semelhante ao realizado em (Lawrence et al., 1996).

Por exemplo, dado o segmento: “.... violência causa náuseas à sociedade”, este recebeu os seguintes traços:

violência = substantivo, feminino, singular

causa = verbo transitivo direto e indireto, rege preposição “a”

náuseas = substantivo, feminino, plural

à = crase

sociedade = substantivo, feminino, singular

E ainda, o status gramatical “1”, como representação de que está gramaticalmente correta.

Os testes realizados apresentaram resultados em geral satisfatórios, levando-se em conta a grande quantidade e variedade de dados, e o caráter totalmente randômico com o qual foram extraídos. Assim, não era nossa pretensão cobrir todos os possíveis casos.

As Tabelas 2.8 e 2.9 a seguir sumarizam alguns resultados obtidos. BP representa a rede backpropagation e Elman, o modelo de rede recorrente de Elman. A segunda coluna corresponde à porcentagem de classificações corretas (na identificação tanto de exemplos corretos, quanto de incorretos); a terceira coluna representa a porcentagem de classificações incorretas (aqui também entram todos os casos de exemplos corretos quanto incorretos). Ela representa classificações incorretas ou problemas quanto à generalização, que ocorre, por exemplo, quando a rede recebe um exemplo não antes visto e não consegue classificá-lo de forma correta. A coluna “desc” representa a porcentagem de incerteza da rede sobre a classificação correta.

Tabela 2.8. Uso incorreto

modelo	Correto	Incorreto	“desc”
BP	62.8%	32.1%	5.1 %
Elman	41.2%	53%	5.8%
ReGra	100%	—	—

Tabela 2.9. Ausência de crase

Modelo	Correto	Incorreto	“desc”
BP	64%	26%	10%
Elman	68%	26.5%	5.5%
ReGra	28%	72%	—

A Tabela 2.8. sumariza testes com segmentos cuja característica é a presença de exemplos incorretos. A tarefa das redes foi detectar o uso incorreto de crase. Testes comparativos com o revisor ReGra mostraram que este tem uma excelente performance devido às suas regras pontuais, descritas na seção anterior. A baixa performance do ReGra no teste de ausência (Tabela 2.9) ocorre devido ao fato do revisor optar, na maioria das vezes, por se omitir em muitos casos onde não tem certeza sobre os resultados corretos. Já isso não ocorre com as redes neurais, uma vez que têm que produzir, necessariamente, uma saída. Além disso, pudemos observar que o comportamento das redes não foi de todo ruim, uma vez que não houve qualquer intervenção humana no processo de aprendizado.

Alguns pontos contribuíram para os resultados não tão bons:

- O não tratamento de exceções: o processo de codificação manual não pode lidar com exceções;
- O fato de não haver qualquer tipo de processo de desambigüização. Por exemplo, um problema semântico que ocorre com a palavra “uma”, que pode ter duas classes gramaticais: artigo (“O menino ganhou uma bicicleta”) e numeral (“Ele chegou há uma hora.”).

O trabalho com os modelos connexionistas representou nossa primeira tentativa no sentido de construção de sistemas empíricos para revisão gramatical do português, envolvendo textos reais em ambientes abertos, sem manipulação prévia.

Resultados comparativos com o revisor ReGra mostraram que ainda há muito para ser feito, no sentido de produção de modelos baseados em corpus para a revisão gramatical automática. Reformulações ou adaptações de tais modelos são extremamente necessárias para cobrir todo o processo de revisão (incluindo o *parser* e o sistema de diagnóstico (*checking*)). Também é imprescindível um processo de desambigüização (casos que possam ser computacionalmente tratados) para que haja uma categorização correta das palavras. Enfim, muito trabalho ainda tem que ser feito para a solidificação de tais modelos aplicados para esse propósito.

Como pudemos perceber, muitas são as dificuldades na construção de ferramentas para a revisão automática de textos. Grandes problemas surgem com os modelos construídos nos moldes racionalistas, com relação ao gerenciamento e à manutenção de toda a estrutura do sistema, no intuito de contemplar o maior número possível de construções lingüísticas.

O *parsing* também se constitui num gargalo para a revisão gramatical. A atribuição da função sintática de todos os elementos da sentença é fundamental para a fase de diagnóstico. O problema da pluricategorização das palavras é um dos fatores que mais contribuem para o insucesso do processo de parsing. O que ocorre nas línguas naturais é que muitas palavras têm mais de uma classe gramatical. Sendo assim, para que determinada palavra receba a categorização correta, algum tratamento de desambigüização deve ser disparado. O desambigüizador pode lançar mão de informações semânticas e de contexto no qual estão inseridas as palavras. Evidências mostram que os métodos empíricos têm vantagens nesse sentido sobre os métodos baseados em regras codificadas manualmente.

Outro problema é quanto à variedade de construções lingüísticas das línguas naturais, principalmente em línguas como o português, onde a inversão dos constituintes sintáticos dentro das sentenças é muito usada, permitindo que as sentenças sejam expressas nas mais diferentes formas. Muitos problemas surgem também com relação à regência, uma vez que muitos verbos possuem mais de uma transitividade e regem mais de uma preposição. Os tratamentos de concordância nominal e verbal, e a recuperação de referentes dos pronomes anafóricos, também constituem dificuldades para o *parsing*.

Assim, uma ferramenta de revisão gramatical nunca pode ser considerada completa, uma vez que a contemplação de todos os erros cometidos durante a escrita é uma tarefa quase impossível. O constante aperfeiçoamento do escopo atingido depende de muito esforço por parte de lingüistas e projetistas. Qualquer ajuda que possam trazer os modelos empíricos já é um grande avanço. Tais modelos podem contribuir, principalmente, para a extração de generalidades dos padrões lingüísticos que não são percebidos num processo de codificação manual do conhecimento.

3. Métodos Empíricos de Parsing

Dentre os sistemas de processamento de línguas naturais desenvolvidos atualmente, grande é a quantidade daqueles dedicados à construção de analisadores sintáticos (*parsers*). O processo de análise sintática (*parsing*) de uma sentença consiste na combinação sucessiva de símbolos dela pertencentes, com o objetivo de reconhecer sua estrutura sintática, validada pela gramática. Essa estrutura sintática é formada por segmentos denominados constituintes. Por exemplo, a união de um artigo com um nome forma um constituinte chamado sintagma nominal. Para que uma sentença seja bem formada, é necessário que ela tenha no mínimo um sintagma nominal (ex: “o gato”) mais um sintagma verbal, que pode ser representado por um verbo (ex: “correu”).

Para realizar seu trabalho, o *parser* necessita da ajuda de dois componentes essenciais: um léxico e uma gramática. O léxico consiste de uma base de palavras categorizadas com suas classes gramaticais e possivelmente com informações morfossintáticas e semânticas a seu respeito.

A gramática consiste na descrição das regras formadoras de uma língua ou um subconjunto delas. Para a construção da gramática é necessário que haja um formalismo a ser seguido. O grande obstáculo encontrado na construção do *parsing* está na formação de tal gramática, que pode ser feita manualmente, o que torna o processo bastante custoso devido à grande complexidade da língua; ou através de indução ou extração de conhecimento, a partir de grande número de exemplos de sentenças. Esse último método, dito empírico, está sendo bastante empregado nos sistemas atuais.

Existem várias abordagens utilizando esse processo de aquisição automática de gramática, como descreveremos nas seções seguintes. São elas: a abordagem simbólica de aprendizado de máquina, a abordagem estatística e a abordagem neural conexionista.

Na aquisição simbólica de conceitos (aprendizado de máquina), o sistema aprende construindo uma representação simbólica de um dado conjunto de conceitos através da análise de exemplos e contra-exemplos desses conceitos. As representações geralmente estão na forma de expressões lógicas, árvores de decisão, regras de produção ou redes semânticas.

A abordagem neural consiste de uma rede de nós (ou neurônios) bastante simples, altamente conectados entre si. Exige pouco conhecimento inicial e o sistema aprende através de uma constante mutação dos parâmetros da rede, que são as conexões, ou pesos, entre os nós. A característica principal, e de maior contraste com os modelos simbólicos, é o fato de serem modelos matemáticos, que dita o caráter numérico dos métodos e algoritmos de aprendizado.

Já os métodos estatísticos utilizam-se de modelos bastante consolidados na área, como as cadeias de Markov, para calcular probabilidades sobre a ocorrência das estruturas sintáticas.

A seguir, descreveremos, em linhas gerais, alguns *parsers* para o inglês construídos sob as três abordagens citadas.

3.1. Propostas de *Parsing* baseados em Aprendizagem Simbólica

Os métodos simbólicos que têm sido usados nas pesquisas atuais incluem indução baseada em regras transformacionais (Brill, 1995, 1993), indução de árvores de decisão (Hermjakob & Mooney, 1997), lógica de programação indutiva (Zelle & Mooney, 1996a., 1996b), entre outros. Esses sistemas adquirem automaticamente conhecimento em alguma forma de regras ou

simplesmente relembram exemplos específicos do passado e tomam decisões baseadas na similaridade com essas instâncias armazenadas.

3.1.1. Modelo de Brill (1993)

3.1.1.1. Características

Brill descreve uma técnica para *parsing* que pode ser aplicada para textos reais. Através do aprendizado automático de um tipo de gramática transformacional baseada na adição e remoção de delimitadores (parênteses) para a formação ou separação de constituintes, garante-se um processo de *parsing* que constrói a árvore sintática binária com nós não-terminais não rotulados.

A técnica de indução de gramática proposta por Brill tem como ponto de partida um estado de conhecimento bastante primitivo da estrutura da frase. Com o uso de um algoritmo de aprendizado dirigido por erro (*error-driven*), o sistema aprende um conjunto de transformações através de comparações entre a estrutura obtida e a correta (Figura 3.1).

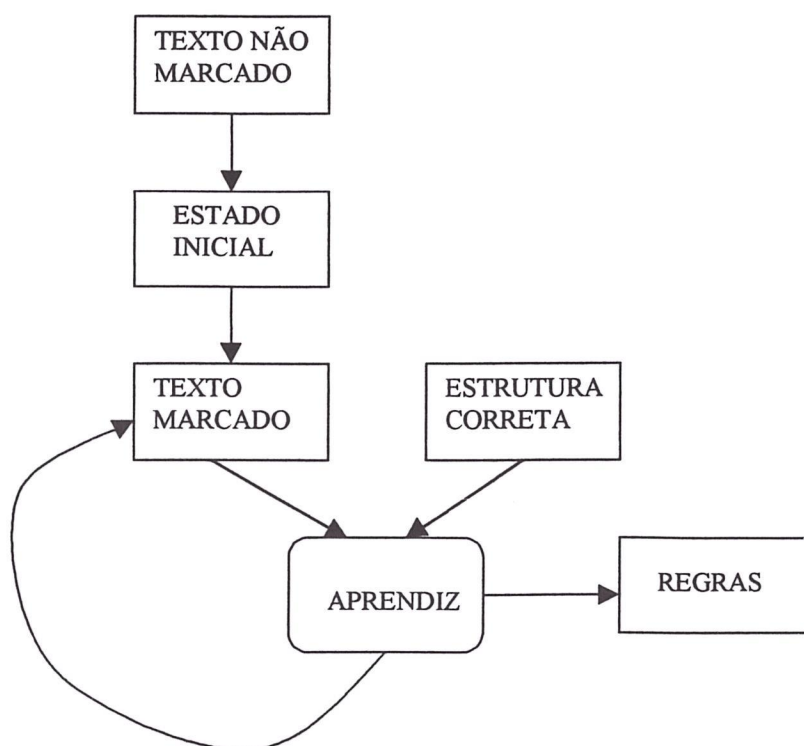


Figura 3.1. Arquitetura do Modelo de aprendizado de Brill (1993)

Uma vez que as transformações são aprendidas, o sistema é capaz de pegar sentenças com palavras marcadas e retornar uma estrutura de árvore binária com nós não-terminais não rotulados.

Por exemplo, dada a sentença “The dog barked”, o primeiro estágio é a obtenção da estrutura linear parentizada com alinhamento à direita, com o ponto final como sendo o de mais alta prioridade. Assim, temos:

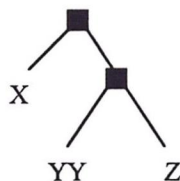
((The (dog barked)) .)

O segundo estágio consiste no aprendizado das transformações, que se dá através de simples adições ou remoções de parênteses (sempre com a ajuda de rotinas que mantêm o balanceamento). No caso, se uma transformação do tipo “apague o parêntese esquerdo do símbolo à direita do determinante” é aplicada, após os balanceamentos necessários teremos

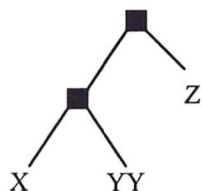
(((The dog) barked) .)

resultantes das regras: 1) apague o parêntese esquerdo; 2) apague parêntese direito que casa com o apagado anteriormente; 3) adicione um parêntese esquerdo à esquerda do constituinte imediatamente à esquerda do parêntese esquerdo removido e 4) se não há constituinte imediatamente à direita ou à esquerda, então a transformação falha.

Assim, uma sub-árvore da forma



Transforma-se em:



3.1.1.2. Aprendizado das transformações

São doze os modelos inicialmente propostos de transformações:

- (1-8) (adicione/remova) um parêntese (esquerdo/direito) à (esquerda/direita) do *tag* de marca X;
- (9-12) (adicione/remova) um parêntese (esquerdo/direito) entre os *tags* X e Y.

O processo de aprendizado das transformações segue da seguinte forma: aplicar cada um dos doze modelos; o que receber melhor pontuação torna-se o primeiro do conjunto; esta é então aplicada à estrutura linear; a transformação aplicada vai para o final da fila; recomeça o processo.

Várias medidas de sucesso podem ser usadas. No artigo, Brill usa a medida de porcentagem de constituintes (palavras entre parênteses) de sentenças que não sofrem conflito com constituintes no Penn Treebank (Marcus et al., 1993).

O método é bastante confiável e simples de ser aplicado. Diferentemente dos outros métodos de indução que usam estatísticas pesadas, esse é fracamente estatístico (somente inteiros são necessários e só existem operações como adição e comparação de inteiros). A gramática resultante é bastante simbólica. Ao invés de tentar encontrar uma gramática para maximizar a probabilidade do corpus de treinamento na esperança de que essa gramática casará com as descrições mais corretas, o aprendiz baseado em transformação pode facilmente usar qualquer medida de sucesso desejada no aprendizado.

3.1.1.3. Vantagens

Dentre as vantagens do método, destaca-se o fato de ser bastante simples, pois necessita de um conjunto relativamente pequeno de exemplos de treinamento, bem como pequeno também é o conjunto de transformações resultantes, com grau de confiabilidade alto. O *parser* resultante é totalmente simbólico.

3.1.1.4. Resultados

A partir de testes com o corpus *Texas Air Travel Information System* (ATIS) (Hemphill; Godfrey & Doddington, 1990), o sistema aprendeu 64 transformações contra 4096 regras livres de contexto do algoritmo *inside-outside* (Baker, 1979), e contou com 91.12% de confiabilidade contra 90.36%.

Em outros testes realizados, a medida foi de 90% de acerto, variando conforme o tamanho de sentenças e transformações a serem aprendidas.

3.1.1.5. Conclusão

A precisão do sistema é medida em termos de porcentagem de constituintes que não são conflitantes no conjunto de teste. Um número adicional de mudanças deve ser feito após cada transformação para manter o balanceamento dos parênteses. A inserção de parênteses é feita da direita para a esquerda. Com a inserção randômica de parênteses foram necessárias 147 transformações contra 64, com 87% contra 91% de eficiência.

Chega-se a 62% o número de classificação sem nenhum erro, para sentenças de tamanho dez. Quanto mais se aumenta o tamanho da sentença menor se torna a porcentagem de acerto.

Esse método baseado em transformações é usado em outros trabalhos do autor. Em (Satta & Brill, 1996) os autores propõem uma extensão para esse trabalho. As regras de transformação são pré-processadas, derivando um autômato de estado finito determinístico em forma de árvore. O algoritmo explora esse autômato de uma maneira que previne a necessidade de checagem das condições de uma regra quando ela não será aplicada. Isso melhora consideravelmente o tempo de execução do algoritmo de *parsing*, uma vez que só se gasta tempo com regras que podem ser aplicadas. Num outro trabalho (Mangu & Brill, 1997), os autores usam as regras baseadas em transformação em problemas de correção de erros ortográficos.

3.1.2. Modelo de Hermjakob & Mooney (1997)

3.1.2.1. Características

Os autores propõem um modelo de *parsing* baseado em contexto, que utiliza sentenças do Wall Street Journal para treinamento e teste. Com o uso de técnicas de aprendizado de máquina, o

sistema analisa exemplos e de uma maneira supervisionada gera um *parser* determinístico do tipo shift-reduce (Marcus, 1980), na forma de uma estrutura de decisão.

O sistema tem como ponto principal o uso de contexto, codificado em características que descrevem aspectos importantes como morfologia, sintaxe e semântica de cada estado da análise. O *parser* gerado transforma as sentenças de entrada em uma árvore de estrutura de frase com frames, que se tornam poderosos a ponto de serem usados para futuras aplicações como tradução, por exemplo.

Antes do processo de *parsing* propriamente dito, a sentença é segmentada em palavras e marcas de pontuação e é enviada ao analisador morfológico, que produz frames primitivos para as palavras. Uma palavra obtém um frame primitivo para cada rótulo sintático (*part of speech*) que possui, sendo a ambigüidade morfológica capturada dentro de cada frame.

O *parser* conta em sua estrutura com uma pilha e uma lista de entrada (Figura 3.2), ambas contendo árvores de frames de palavras ou frases. No exemplo ilustrado, temos que o topo da pilha é composto por um frame representando o sintagma nominal (np na Figura 3.2), anteriormente formado, “a book”. Logo em seguida, temos um verbo (“bought”). Uma operação de redução faz com que esses dois frames sejam combinados formando o sintagma verbal (vp) “bought a book”.

3.1.2.2. Vantagens

O fato de o processo de construção de *parsing* transformar as seqüências ordenadas em ações pequenas e ordenadas, como shift e reduce, faz com que seja mais fácil de ser entendido, uma vez que tem que ser supervisionado durante o treinamento.

Existem várias ações que o *parser* pode desempenhar, podendo ter vários argumentos cada, fazendo com que a linguagem de análise desenvolvida seja bastante poderosa.

Para que o sistema tomasse boas decisões de análise foi necessária a definição de uma linguagem de características. Essa linguagem expressa características em vários graus de abstração. As características servem para resolver conflitos durante o processo de análise, por exemplo.

3.1.2.3. Resultados

O sistema foi treinado com cerca de 105.000 sentenças de um corpus do Wall Street Journal, com tamanhos dos mais variados. A precisão variou bastante conforme o número de sentenças usadas no treinamento, produzindo taxas que variam entre 88% a 92.7%. O número de

características usadas também influencia na precisão, que se torna maior conforme aumenta o número de características que o sistema pode dispor.

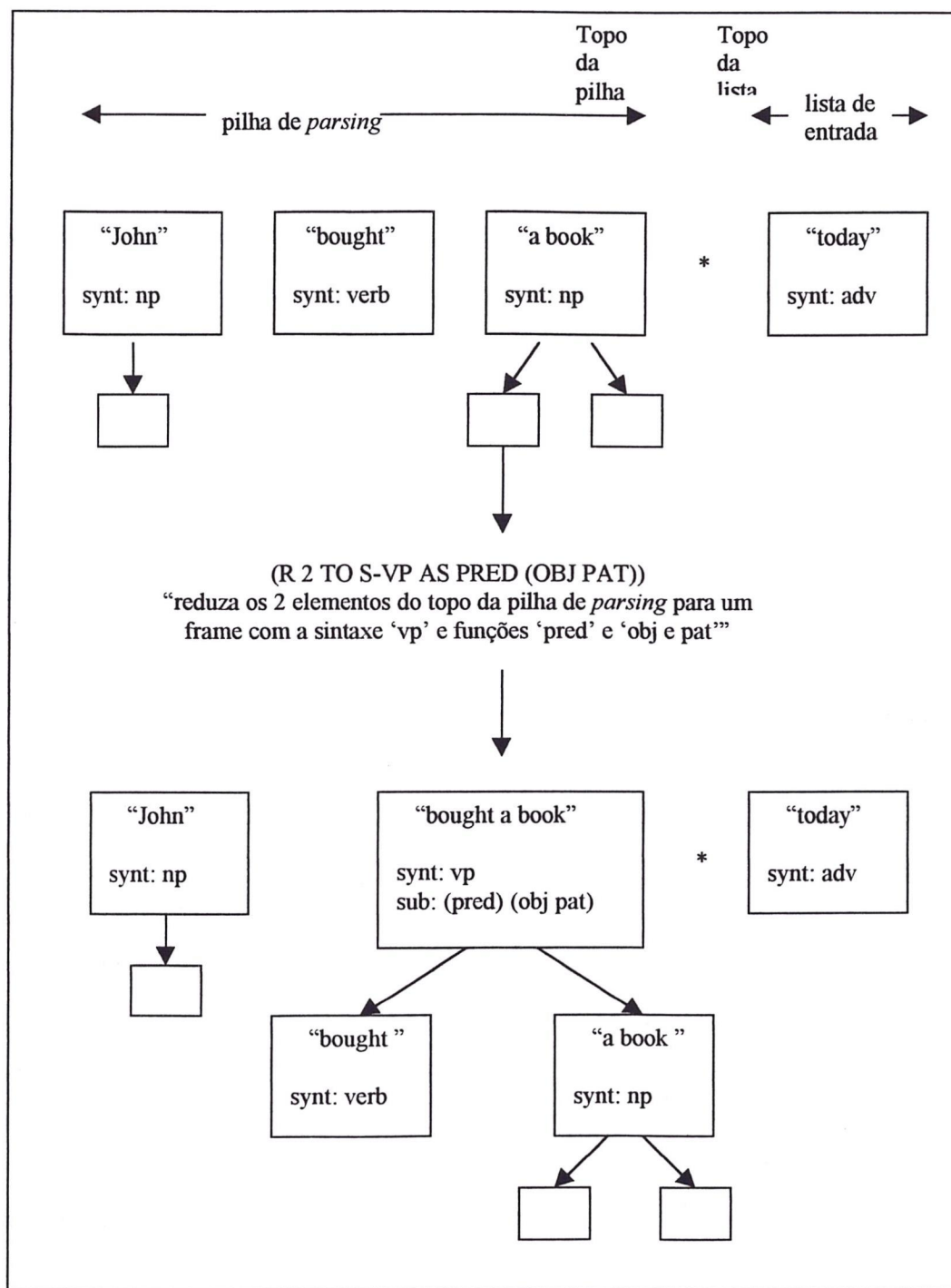


Figura 3.2. Exemplo de uma ação de análise (simplificada); as caixas representam *frames*.

3.1.2.4. Conclusão

O sistema tem por objetivo tentar construir uma ligação entre as abordagens de construção manual tradicionais, mas de pequena escala, com as abordagens estatísticas, que são de grande escalabilidade, mas pobre em contexto, para o trabalho com textos irrestritos.

Comparado com os métodos estatísticos, o sistema proposto conta com uma análise mais profunda e necessidade de uma maior supervisão. Os autores comparam seus resultados com os obtidos por Magerman (1995) (veja subseção 3.3.3), e concluíram que o uso de um conjunto de características maior torna o sistema mais eficiente.

3.1.3. Modelo de Zelle & Mooney (1996b)

3.1.3.1. Características

O trabalho proposto por Zelle & Mooney apresenta o sistema CHILL, que é um *parser* empírico. CHILL trata a aquisição da linguagem como o aprendizado de regras de controle (*search-control rules*) dentro de um programa lógico e usa técnicas de programação lógica indutiva (PLI) para essa aquisição.

Uma característica que diferencia das abordagens de aprendizado baseado em árvores de decisão e regras de transformação que, ao contrário destes, o conhecimento adquirido não é representado em forma proposicional.

A programação lógica indutiva se volta para o aprendizado de descrições de lógica de primeira ordem (programas em Prolog). Devido à expressividade dessa lógica, os métodos de PLI podem aprender conceitos relacionais e recursivos.

O sistema CHILL usa uma estrutura com representações estruturadas para o aprendizado. CHILL começa com uma estrutura de *parsing* bem definida e usa a PLI para aprender estratégias de controle dentro dessa estrutura.

CHILL recebe como entrada um conjunto de instâncias de treinamento consistindo de sentenças com suas análises desejadas. A saída é um *shift-reduce parser* em Prolog que mapeia as sentenças em suas análises.

Quanto ao seu funcionamento, CHILL aplica operações de *shift* e *reduce* sobre o estado atual da análise, representado pelo conteúdo de uma pilha e pela porção remanescente num buffer de

entrada. Os operadores de *parsing* são cláusulas que pegam a pilha atual e o buffer como argumentos e retornam uma pilha modificada e um buffer restante. A Figura 3.3. esboça seu funcionamento.

Durante o processo de geração de operadores, os exemplos aprendidos no treinamento são analisados para a extração de operadores que são requeridos para produzir as análises. Como por exemplo: um operador que faz a redução de dois itens da pilha adicionando o segundo como agente do primeiro, formando uma nova estrutura para o topo da pilha pode ser:

$op([Top, Second|Rest], In, [NewTop|Rest], In) :- reduce(Top, agt, Second, NewTop),$

onde os argumentos em *reduce* são respectivamente: pilha atual, buffer, pilha nova, novo buffer.

A partir da extração dos operadores, são extraídos exemplos de controle positivos e negativos dos quais as regras podem ser induzidas.

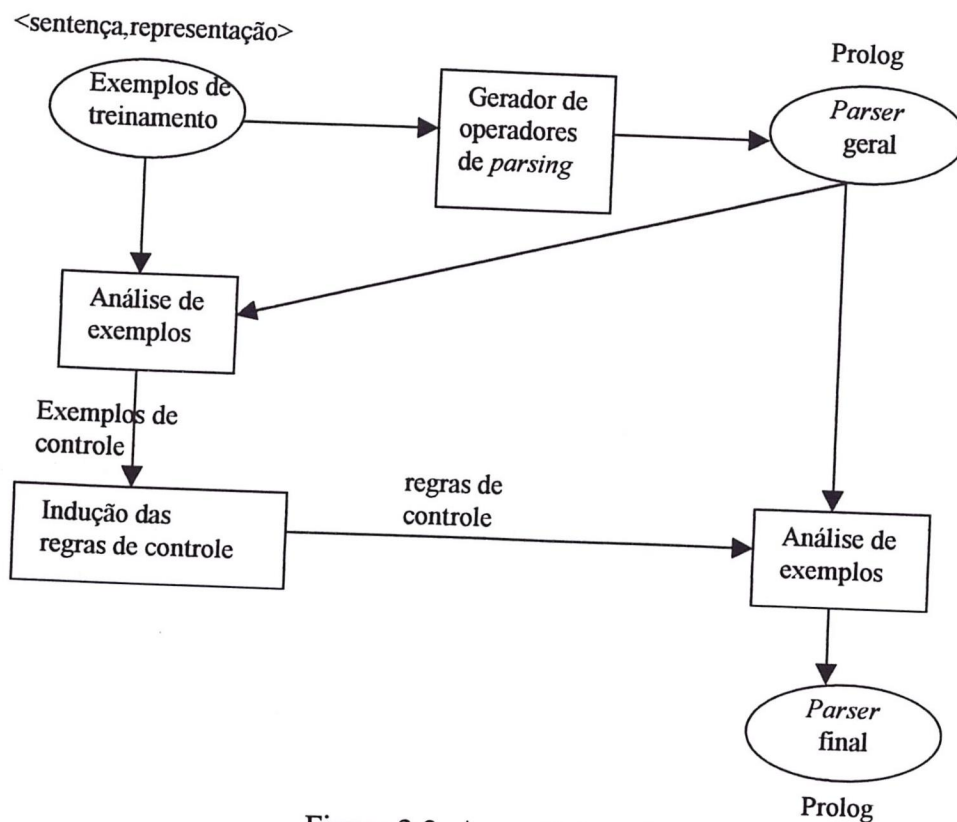


Figura 3.3. A arquitetura do CHILL

Para o exemplo do operador de agente acima, a sentença “the man ate the pasta” produzirá o exemplo positivo $op([ate, [man, det:the]], [the, pasta], A, B)$ e exemplos negativos como $op([man, the], [ate, the, pasta], A, B)$.

A fase de indução de regras de controle usa um algoritmo de indução de primeira-ordem para aprender uma regra de controle para cada operador. Por exemplo, uma regra de controle que pode ser aprendida para o operador agente é:

```
op([X,[Y,det:the]], [the|Z], A, B) :- animate(Y).  
animate(man).  
animate(boy).  
animate(girl).
```

que diz que a redução de agente se aplica quando uma pilha contém dois itens, sendo o segundo completado por um sintagma nominal, cujo núcleo é animado.

A combinação da cláusula com a regra de controle resulta na cláusula:

```
op([A,[B,det:the]], [the|C], [D], [the|C] :- animate(B), reduce (A, agt, [B,det:the], D).  
animate(boy).  
animate(man).
```

O parser resultante no final da operação conta com cada operador extraído conforme o exemplo acima.

3.1.3.2. Vantagens

A principal vantagem do uso de técnicas de PLI é a flexibilidade resultante, dado o poder das regras de primeira ordem. O algoritmo de indução pode extrair automaticamente as porções relevantes dos contextos estruturados e construir novos predicados para representar novas classificações e categorias de frase que são necessárias para o desempenho satisfatório do *parsing*.

3.1.3.3. Resultados

Os testes foram feitos usando uma das primeiras versões do TreeBank, com 729 sentenças de tamanho médio de 10.3 palavras etiquetadas com categorias lexicais. Após o treinamento com 525 sentenças, CHILL construiu 40% do total das árvores de *parsing* de forma completamente correta. Usando uma média de precisão, chegou a 84% em estudos com

parentização automática. Em (Brill, 1993), descrito nas seções anteriores, os resultados com as mesmas sentenças foram de 62% e 91.12% respectivamente.

3.1.3.4. Conclusão

Segundo os autores, uma das principais atrações da LPI é a facilidade com que ele pode ser integrado aos métodos simbolistas tradicionais de *parsing*.

Além disso, a estrutura de shift e reduce é fundamental para o seu sucesso em aprender tarefas realistas de processamento de língua.

3.2. Propostas de *Parsing* baseados em métodos estatísticos

Muitos dos trabalhos recentes no processamento empírico de língua têm envolvido técnicas estatísticas de treinamento para modelos probabilísticos, como *HMMs* e as *gramáticas livres de contexto probabilísticas* (Charniak, 1993, 1997). Esses métodos associam probabilidades às transições de uma máquina de estado finito ou às produções de uma gramática formal, e estimam esses parâmetros probabilísticos baseados nos dados disponíveis no treinamento utilizado.

Se esse conjunto de treinamento for previamente anotado com a estrutura a ser aprendida, o aprendizado consiste simplesmente na contagem dos vários eventos existentes no conjunto de dados. Se esse não for anotado, uma estratégia de estimativa de maximização pode ser usada (como o algoritmo *inside-outside* para treinar gramáticas livres de contexto probabilísticas (Baker, 1979)).

Tais medidas estatísticas podem ser usadas para analisar caminhos mais prováveis na geração de seqüências aprendidas ou para determinar a frequência com que cada seqüência de *n*-palavras (*n*-grams) aparece na língua.

Os modelos apresentados a seguir mostram alguns trabalhos que estão sendo desenvolvidos com esse enfoque.

3.2.1. Modelo de Soderland & Taira (1999)

A teoria dos campos (field theory) é um paradigma para *parsing* que trata as palavras não como objetos passivos de regras gramaticais, mas como partículas exercendo forças de atração em direção a outras partículas. Essa força é estimada por um modelo estatístico que é indexado pelas características sintáticas e semânticas das palavras envolvidas.

O sistema de *parsing* proposto pelos autores faz parte de um grande sistema que parte de textos de relatórios sobre assuntos médicos, e procura produzir uma representação interna, visando o processo de interpretação dos textos.

A partir das descrições, o *parser* identifica quais palavras são modificadoras de outras palavras e cria um grafo de dependência.

Por exemplo, dada a sentença “the large mass within the posterior segments of the right upper lobe has increased in size” produzirá o grafo de dependência mostrado na Figura 3.4, onde um arco dirigido da palavra A para a palavra B no grafo de dependência indica que A é modificador de B.

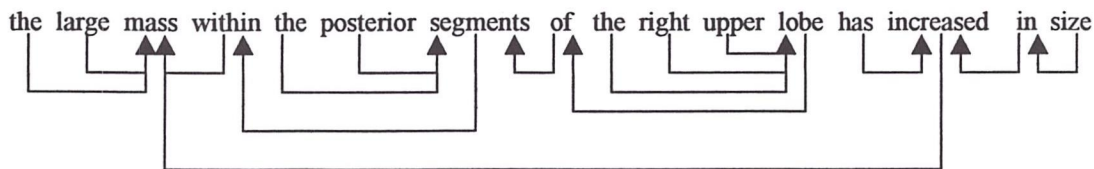


Figura 3.4. Arcos mostrando dependências na sentença (Soderland and Taira, 1999).

Esse tipo de relação é usado para todo tipo de dependência sintática como complementos verbais, ligações dos sintagmas preposicionais, etc.

O processo se inicia com uma sentença que não tem dependência entre as palavras. Ela é vista como uma sequência de partículas (palavra + características sintáticas e semânticas) isoladas, cada uma exercendo uma certa dependência de atração entre partículas adjacentes. Um arco é adicionado entre um par de partículas com ressonância mais forte (estimada de um conjunto de treinamento marcado). A ressonância entre uma partícula A e uma partícula B é definida como a probabilidade que A se liga em B como um modificador.

O *parser* reavalia atrações dos campos após o arco ter sido adicionado e novamente seleciona o par com ressonância mais forte. Cada iteração adiciona um arco construindo gradualmente estruturas complexas.

Essas estatísticas são indexadas por características gramaticais e semânticas das palavras, que são obtidas de um léxico.

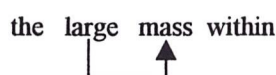
Ex: “mass” tem características gramaticais *nome/singular*

E tem características semânticas *anormal/objeto físico, invenção...*

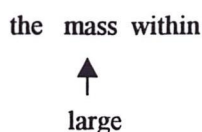
Por exemplo, de acordo com o conjunto de treinamento a ressonância é alta entre um adjetivo denotando tamanho e um nome denotando objeto físico. Assim, explica-se a ligação entre “large” e “mass”.

A ressonância entre partículas é estimada com um modelo estatístico parametrizável baseado em trigramas. No processo inicial, pegam-se trigramas para cada conjunto de três palavras adjacentes. Cada trigrama recebe uma ressonância. No trigrama de mais alta ressonância é adicionado um arco para a sentença. As dependências de longa distância entre partículas A e B são feitas ignorando-se modificadores intermediários que foram associados por qualquer partícula.

Assim, a primeira dependência que o sistema produz é:



Logo após o estabelecimento da dependência, a palavra passa a não fazer mais parte na formação do trigrama, permitindo assim, que as palavras restantes sejam combinadas em novos trigramas.



O problema do trigrama surge quando analisamos as frequências com que aparecem no corpus de treinamento. Por exemplo, o trigrama (“the”, “large”, “mass”) não tem nenhuma ocorrência no corpus. No entanto, se generalizarmos com suas características haverá vários exemplos de “the” seguido por adjetivos denotando tamanho (tal como a palavra “large”) e substantivos.

Um pré-processamento é feito para desambiguação e são feitas mudanças nas características gramaticais das palavras conforme necessário.

O fato de as estatísticas serem indexadas por características gramaticais e semânticas permite ao *parser* lidar com falhas nos dados de treinamento generalizando as características das palavras faltantes.

3.2.1.1. Resultados

Alcança boa performance. Média de 80% com 100 sentenças no treinamento e 90% com 1000.

3.2.1.2. Vantagens

O *parser* alcança boa performance usando conjuntos de treinamento relativamente pequenos. Proporciona um mecanismo de degradação brando para generalizar palavras não vistas no treinamento.

3.2.1.3. Conclusão

A teoria dos campos (field theory) usa o meio termo entre indexar as estatísticas em categorias gramaticais (que são contexto insuficiente para decisões de *parsing*), e a indexação entre palavras exatas (que exigem um conjunto de treinamento grande e falham em eficiência).

3.2.2. Modelo de Charniak (1997)

O autor apresenta um analisador sintático no qual a gramática e os parâmetros probabilísticos são induzidos a partir do Treebank (Marcus et al., 1993). O produto de tal analisador consiste na mesma forma de árvore sintática utilizadas para o treinamento.

O modelo proposto é probabilístico no sentido de que ele retorna a análise sintática π de uma sentença s que maximiza $p(\pi|s)$. Assim, o *parser* opera atribuindo probabilidades $p(\pi,s)$ para a sentença s sobre toda as possíveis análises de π (ou no mínimo todas as que ele constrói), e então escolhe aquela para a qual a probabilidade $p(\pi,s)$ é maior.

Assim, por exemplo, consideremos a sentença tomada como exemplo "Corporate profits rose.", cuja árvore sintática está representada na Figura 3.5. Para cada nó não terminal é identificado seu núcleo (que é o item léxico mais importante) (ex: para **np** o núcleo é o nome **n**).

Uma vez que os núcleos dos constituintes são os mesmos que de seus constituintes imediatos, esses são especificados de forma bottom-up. Formalmente, o núcleo é determinado por uma função determinística da regra gramatical usada para construir o constituinte.

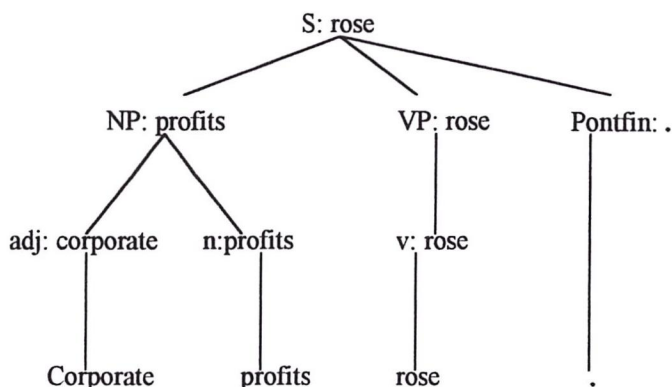


Figura 3.5. Análise sintática de uma sentença (Charniak, 1997).

A partir da atribuição dos núcleos de cada constituinte, o sistema encontra as possíveis árvores sintáticas para a determinada sentença, e escolhe a de mais alta probabilidade. Por exemplo, supondo que queremos determinar a probabilidade do constituinte **NP** em "corporate profits". O primeiro passo é determinar a probabilidade de seu núcleo; então a probabilidade da forma do constituinte dado o núcleo, e assim recursivamente, para encontrar a probabilidade de seus constituintes imediatos. Assim, determina-se a probabilidade de que um **NP** tem o núcleo "profits" dado que é um **NP** e que o constituinte acima é um **S** cujo núcleo é "rose". Ou $p(\text{profits}|\text{rose}, \text{NP}, \text{S})$.

O sistema usa uma gramática livre de contexto e encontra um conjunto de análises (*parses*) de alta probabilidade para a sentença. Antes do *parsing*, o analisador é treinado com o corpus para a obtenção de uma gramática livre de contexto (uma *treebank grammar*) e atribuição das probabilidades necessárias observadas (várias probabilidades são consideradas levando-se em conta vários eventos condicionados combinados, como somente o tipo do constituinte, ou então o tipo do constituinte mais o tipo de seu pai, ou então como descrito acima). Isso tanto é válido para os constituintes como para a obtenção da probabilidade de uma regra gramatical.

3.2.2.1. Vantagens

Duas características diferem esse sistema: a primeira é o fato de o aprendizado ser não supervisionado, uma vez que o corpus usado como treinamento é considerado correto, e, portanto, apto a fornecer todas as probabilidades necessárias.

A segunda característica que contribui com sua eficiência está na abrangência na qual são calculadas as probabilidades, uma vez que essas são condicionadas a vários eventos. Isso permite o uso do *parser* por outros sistemas, pois o sistema consegue computar a probabilidade geral da sentença.

O fato de o sistema ter uma gramática explícita traz várias vantagens como: 1) um nível de controle mais aprofundado. Por exemplo, o fato de a palavra "give" pedir tanto objeto direto como indireto e o mesmo não acontecer com "put", por exemplo, é tratado com duas probabilidades $p(vp \rightarrow \text{verb NP NP}|\text{give}) = 0.25$ e $p(vp \rightarrow \text{verb NP NP}|\text{put}) = 0$.

3.2.2.2. Resultados

Os testes foram realizados com conjunto de treinamento de 1 milhão de palavras e teste com 50.000 palavras. Um constituinte rotulado produzido pelo *parser* é considerado correto se existe um constituinte no Treebank com: 1) o mesmo ponto de partida, 2) o mesmo ponto final, e 3) o mesmo rótulo.

Vários testes foram realizados, com diversas combinações de probabilidades e os resultados tiveram sucesso, em média, em 86.5% dos casos.

3.2.2.3. Conclusão

O sistema começa com pouco conhecimento e reúne as estatísticas que necessita a partir do corpus de treinamento.

Os níveis que são levados em consideração são apenas o núcleo do constituinte e o núcleo de seu pai, mas não de seu avô.

O sistema elege como análise correta aquela com mais alta probabilidade de acordo com um grau estabelecido.

Apesar dessas características, o sistema não tem um passo de etiquetagem explícita e não há como tratar ocorrências de várias classes para uma mesma palavra. O sistema considera como etiqueta "correta" aquela que está na análise de mais alta probabilidade.

3.2.3. Modelo de Magerman (1995)

Magerman apresenta um *parser* estatístico baseado em técnicas de aprendizado com árvores de decisão, e que constrói uma árvore sintática completa para qualquer sentença.

Segundo o autor, o processo de análise sintática pode ser visto como uma seqüência de decisões de desambigüizações, dentre elas a determinação dos rótulos das palavras, escolhas entre as estruturas de constituintes possíveis e a seleção de rótulos para esses constituintes.

O autor trata do problema da descoberta automática desses critérios de desambigüização para todas as decisões tomadas durante o processo de *parsing*, dado o conjunto de características possíveis que podem agir como desambigüizadores. Os candidatos são as palavras na sentença, os relacionamentos entre as palavras e os relacionamentos entre os constituintes já construídos no processo de *parsing*.

As decisões são tomadas de uma forma não determinística com base na probabilidade de cada escolha. As probabilidades são estimadas usando modelos de árvore de decisão estatísticos, onde a probabilidade de uma árvore sintática completa (T) de uma sentença (S) é o produto de cada decisão (d_i) condicionada a todas as decisões anteriores:

$$P(T | S) = \prod_{d_i \in T} P(d_i | d_{i-1} d_{i-2} \dots d_1 S)$$

A árvore sintática escolhida é aquela cuja seqüência de decisões produz a maior probabilidade cumulativa (Figura 3.6). As decisões em consideração envolvem identificar os constituintes e seus rótulos em sentenças de língua natural.

São três as tarefas: identificar as características que são relevantes para cada decisão, atribuir probabilidades para as possíveis escolhas e decidir qual escolha selecionar com base nos valores das características relevantes.

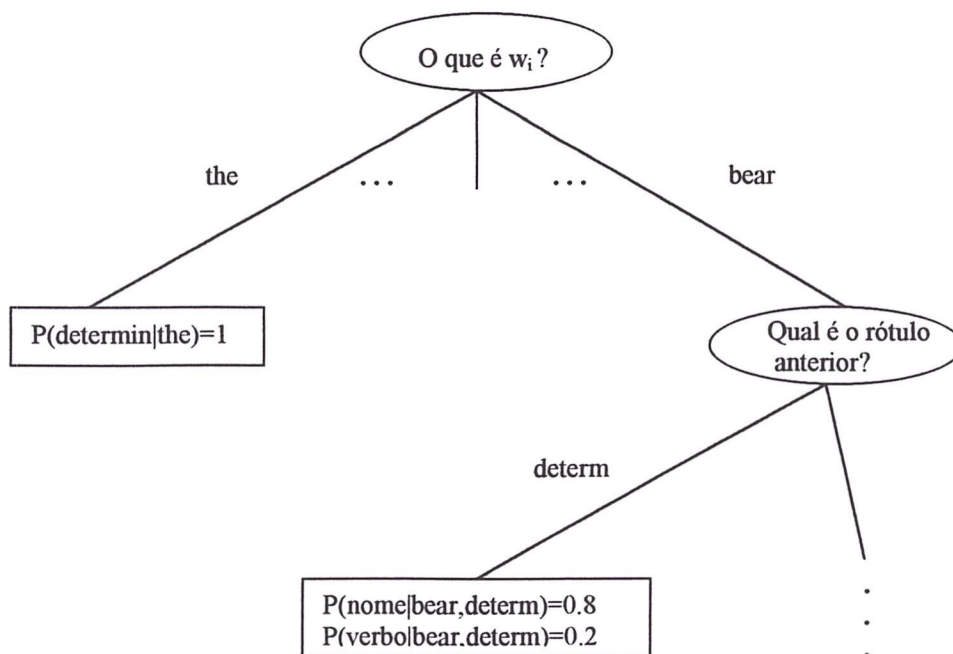


Figura 3.6. Árvore de decisão parcial de determinação de rótulos (Magerman, 1995).

Para o processo de *parsing*, a árvore sintática de uma sentença é construída começando-se com as palavras como folhas da árvore, e então, rotulando e estendendo esses nós até que uma árvore mínima seja construída, de uma forma bottom-up e left-to-right.

3.2.3.1. Vantagens

Segundo o autor, a vantagem do modelo proposto está no fato de que ele produz um *parser* preciso sem a ajuda de uma base de conhecimento complicada ou uma gramática, e que conta apenas com uma informação lingüística bastante limitada.

Outra vantagem está no fato de que o modelo baseado em árvores é capaz de desambigüizar informações léxicas de longa distância, uma vez que a tecnologia de árvores de decisão pode ser usada para construir e estimar modelos que seletivamente escolhem elementos do contexto que contribuem para as decisões de desambigüização.

O sistema não faz simplesmente uma atribuição prévia dos rótulos das sentenças e escolhe aquele com a melhor sequência. Ao invés disso, ele usa um modelo probabilístico para

determinar os rótulos para palavras, e considera todas as seqüências possíveis de acordo com a probabilidade com a qual são atribuídas.

3.2.3.2. Resultados

Em testes realizados com o Penn Treebank (Marcus et al., 1993), o sistema proposto por Magerman alcançou +- 86% de precisão para sentenças de 40 palavras ou menos, e +- 91% de precisão para sentenças com tamanho entre 10 e 20 palavras.

Apesar dos resultados obtidos serem bastante satisfatórios, falta ainda ser mostrado que a cobertura abrangente de tal sistema pode melhorar a performance das aplicações de processamento de texto.

3.2.3.3. Conclusão

De acordo com o autor, independentemente de quais técnicas são usadas para o processo de desambigüização, toda a informação necessária para a realização de tal processo deve estar disponível. As palavras são claramente necessárias para tomar as decisões, e em alguns casos a informação estrutural de longa distância é também necessária. Modelos estatísticos para *parsing*, segundo o autor, devem considerar muito mais características de uma sentença do que podem ser manuseadas por modelos n-grams.

O modelo proposto ilustra como grandes quantidades de informação contextual podem ser incorporadas em um modelo estatístico para *parsing* aplicando-se algoritmos de aprendizado de árvores de decisão a partir de um grande corpus marcado.

Em (Collins, 1999), o autor traça comparações sobre o modelo de Magerman e também o de Charniak, da subseção anterior, com o trabalho por ele desenvolvido.

3.3. Propostas de *Parsing* Conexionistas

O processamento de sentenças consiste em um grande desafio para a pesquisa com modelos conexionistas, que tem por propósito serem alternativas aos sistemas simbólicos.

Duas classes de modelos provêm tais alternativas (Christiansen & Chater, 1999), ambas enfocando o aprendizado a partir da experiência. Na primeira classe, o aprendizado se dá a partir

de sentenças etiquetadas. As redes são treinadas com sentenças, cada qual associada a uma estrutura gramatical particular, e a tarefa é atribuir estruturas gramaticais apropriadas para novas sentenças. Assim, as estruturas são inferidas não por observação, mas, via dados de treinamento.

A segunda classe de tais modelos tem por objetivo o aprendizado de estruturas sintáticas a partir de seqüências de palavras. Forte representante desse ramo é o trabalho de Elman (1990) com redes recorrentes¹⁰, incluindo também o de Tabor & Tanenhaus (1999) e Lawrence, Fong & Giles (1996). Tais redes mostraram similaridades com o comportamento humano em simulações com estruturas encaixadas, concordância, entre outros, o que sugere que possam adquirir algumas regularidades gramaticais.

Somam-se ainda trabalhos como os de Christiansen (1998), Christiansen & Chater, (1994), Henderson (1994, 1996), Lane (1998), Henderson & Lane (1998a, 1998b), cuja proposta é estender as redes de Elman para o tratamento de estruturas mais complexas da língua.

As redes recorrentes têm como característica o fato de lidarem com seqüências temporais, uma vez que as entradas são recebidas de forma incremental. Assim, esse tipo de arquitetura neural é capaz de tratar sentenças de tamanho variado e também é capaz adquirir generalizações sobre as posições estruturais das palavras nas sentenças.

A seguir, apresentamos exemplos de como as redes recorrentes estão sendo usadas para o tratamento da língua natural.

3.3.1. Modelo de Lawrence, Fong & Giles (1996)

Os autores propõem um modelo neural para classificar sentenças quanto a sua gramaticalidade. A proposta de seu trabalho é investigar se as redes neurais têm o poder de produzir os mesmos julgamentos feitos pelos seres humanos quando aprendendo uma língua. Não há distinção entre os princípios universais da língua e os parâmetros aprendidos, que são particulares de cada língua (Chomsky, 1986).

A motivação para o trabalho partiu de comprovadas experiências por outros autores da eficiência das redes neurais, principalmente as recorrentes, na aplicação de problemas lingüísticos, ainda que usados para resolver problemas bastante pequenos (Stolcke, 1990; Allen, 1983; Elman, 1984; Harris & Elman, 1984; St. John & McClelland, 1990, etc.).

¹⁰ Traduzimos como forma simplificada do inglês Simple Recurrent Networks (SRNs)

O sistema proposto utiliza-se somente das classes gramaticais das palavras para treinamento. Essas classes são obtidas de uma maneira completamente livre de contexto sem qualquer tratamento de desambigüização.

3.3.1.1. Vantagens

Uma vantagem do modelo é que ele usa uma gramática mais complexa em relação a trabalhos anteriores, em que as gramáticas são bastante reduzidas (Giles et al., 1995; Clouse et al., 1994).

Os autores se baseiam no fato de que as redes recorrentes têm o poder representacional suficiente para tratar as estruturas hierárquicas encontradas nas línguas naturais (Elman, 1990), o que, segundo (Pereira & Schabes, 1992), os modelos estocásticos baseados em descrições de estado-finito (n-grams ou modelos de Markov) não teriam.

3.3.1.2. Resultados

Os dados de entrada da rede são formados por um conjunto primário de 522 exemplos positivos e negativos retirados de livro texto, organizados em pares mínimos como:

I am eager for John to be here. (sentença correta)

** I am eager John to be here.* (sentença incorreta)

Devido ao pequeno tamanho do exemplo, os dados brutos foram convertidos dentro de 8 categorias principais:

<i>Categoria</i>	<i>Exemplos</i>
Nomes (N)	<u>John</u> , <u>book</u>
Verbos (V)	<u>hit</u> , <u>be</u> , <u>sleep</u>
Adjetivos (A)	<u>eager</u> , <u>old</u> , <u>happy</u>
Preposições (P)	<u>Without</u> , <u>from</u>
Complemento (C)	I thought <u>that</u> ... / I am eager <u>for</u> ...
Determinate (D)	<u>the</u> man / <u>each</u> man
Advérbio (ADV)	I <u>sincerely</u> believe/ <u>Why</u> did John want..
Marcador (MRKR)	possessivos('s, <u>of</u> , <u>to</u>), John's mother

Uma subcategorização foi aplicada sobre nomes, verbos, adjetivos e preposições, devido a resultados melhores apresentados pela rede (ex: verbos intransitivos (sleep) são colocados em lugar diferente de verbos transitivos (hit), e assim por diante.), resultando, assim, 9 classes para verbos, 4 para nomes, 4 para adjetivos e 2 para preposições.

exemplos:

<i>Sentença</i>	<i>codificação</i>	<i>Estatus gramatical</i>
I am eager for John to be here	n4 v2 a2 c n4 v2 adv	1
	n4 v2 a2 c n4 p1 v2 adv	1
I am eager John to be here	n4 v2 a2 n4 v2 adv	0
	n4 v2 a2 n4 p1 v2 adv	0

O conjunto das sentenças foi dividido em treinamento e teste. Os 23 símbolos foram codificados em uma janela de tamanho fixo de segmentos contendo 8 entradas separadas. As subcategorizações foram codificadas linearmente para representar valores específicos de determinada entrada (exemplo: não nome = 0; classe nome1 = 0.5; classe nome2 = 0.667; classe nome3 = 0.833; classe nome4 = 1). A ordem linear foi definida usando julgamento da similaridade entre as várias subcategorias. Duas saídas foram usadas representando classificações gramaticais e não gramaticais. O número de exemplos positivos e negativos é igual de modo a garantir que os modelos não produzam resultados baseados na proporção de positivos para negativos.

Principais resultados obtidos, em porcentagem de classificação correta:

<i>Rede</i>	<i>Treinamento</i>	<i>Teste</i>
MLP	55	54
FGS	56	59
BT-FIR	56	54
Elman	100	74
W&Z	92	71

Onde MLP é o modelo multilayer perceptron, FGS é o modelo de redes recorrentes de (Frasconi, Gori, Maggini & Soda, 1992), BT-Fir é o modelo Back & Tsoi (1991), Elman o modelo parcialmente recorrente (Elman, 1990), e W&Z o modelo de rede recorrente de (Williams & Zipser, 1989).

3.3.1.3. Conclusão

Os autores não chegam a conclusões quanto à capacidade de generalização dos modelos neurais propostos, uma vez que os dados usados para o treinamento foram poucos e esparsos. Contudo, acreditam que o modelo de Elman seja bastante indicado para tal tarefa.

3.3.2. Modelo de Henderson & Lane (1998)

3.3.2.1. Características

Este modelo de *parsing* tem sua origem em Henderson (1994). Trata-se de uma arquitetura conexionista que, de acordo com seus idealizadores, é o modelo que mais se aproxima do ideal para tratar alguns problemas do tão almejado processamento automático de língua natural.

Chamadas de redes sincronizadas simples¹¹, essas redes são uma extensão natural das redes recorrentes de Elman (1990), pois usam uma técnica chamada *Temporal Synchrony Variable Binding (TSVB)* (Shastri & Ajjanagadde, 1993) que lhes dá a habilidade adicional de generalizar também sobre constituintes em diferentes posições estruturais, além de generalizar sobre palavras em diferentes posições na sentença.

Essas habilidades combinadas é que fazem as SSNs adequadas para o *parsing* sintático.

Para melhor entendimento, algumas definições são apresentadas a seguir como a proposta das TSVBs e sua aplicação nas redes sincronizadas simples.

3.3.2.2. Temporal Synchrony Variable Binding (TSVB)

A TSVB é uma extensão do poder representacional das redes conexionistas tradicionais. Os neurônios em uma rede convencional indicam a presença de uma característica no ciclo computacional pela força de sua ativação. A idéia central da TSVB é que os neurônios podem também *pulsar* dentro de cada ciclo.

Neurônios pulsando em sincronia representam características de uma entidade (constituente); neurônios pulsando sem sincronia representam características de diferentes entidades.

Devido ao fato de os pesos das ligações serem constantes em todos os pulsos num ciclo, cada entidade terá o mesmo processamento aplicado a ela. Essa propriedade é usada em (Henderson, 1996) para demonstrar que redes conexionistas usando TSVB generalizam a informação aprendida sobre um constituinte sintático para outros constituintes sintáticos, quando aplicadas para a tarefa de língua natural.

¹¹ Traduzimos de uma forma simplificada do inglês Simple Synchrony Networks (SSNs)

Por exemplo, uma rede treinada para descrever palavras como nomes, quando visto somente na posição de sujeito das sentenças, generalizará aquela informação para as mesmas palavras quando aparecem na posição de objeto. Isso ocorre obedecendo-se ao conceito de sistematicidade apresentado a seguir.

3.3.2.3. A definição de Sistematicidade

Devido ao fato de as generalizações serem expressas em termos de constituintes, tal arquitetura prediz a prevalência de regularidades sobre os constituintes. Por exemplo, o padrão de palavras que formam um sintagma nominal (SN) é muito complexo, contudo esse padrão é virtualmente idêntico tanto para sintagmas nominais que funcionam como sujeito ou objeto. Assim, se “João” pode ser um SN funcionando como sujeito, e “Maria” um SN funcionando como objeto (como em “João ama Maria”), então também será possível para “Maria” ser um SN-sujeito e “João” um SN-objeto (como em “Maria ama João”).

No paradigma conexcionista, as habilidades responsáveis pelas regularidades são testadas através do aprendizado das generalizações e não em termos de representação, como nos paradigmas clássicos. Uma arquitetura conexcionista consegue tal regularidade se ela pode generalizar de um conjunto de treinamento para um conjunto de teste de acordo com essa regularidade. Esse critério é usado para formalizar o conceito de sistematicidade (Hadley, 1994).

Henderson mostra em (Henderson, 1996) que as redes conexcionistas que usam TSVB generalizam informação sobre palavras de uma posição sintática para outra.

3.3.2.4. Variable Binding

As arquiteturas convencionais têm duas dimensões para representar informações de estado, que são as unidades de ativação e as unidades de identidades (constituintes). A chave da arquitetura proposta é que em adição à ativação e identidade, os neurônios têm à disposição uma terceira dimensão para representar a informação. Quando um neurônio está ativo, ele não está simplesmente **on**, ele pulsa. Um pulso tem uma ativação, um identidade, e um tempo. Essa dimensão temporal é que é usada para prover a terceira dimensão que é necessária para representações simbólicas.

Níveis de ativação são usados para representar valores reais, as identidades são usadas para representar predicados, e o tempo de um pulso de ativação da unidade é usado para representar para qual entidade o predicado é verdadeiro.

Tipicamente, as representações simbólicas usam variáveis para representar entidades para as quais predicado é verdadeiro. A identidade de uma variável não especifica por si qualquer informação sobre a entidade, mas serve para estar junto a diferentes predicados sobre essa entidade que são verdadeiros. Esse processo é chamado **variable binding**. Essa arquitetura usa a sincronia temporal dos pulsos das unidades de ativação para representar essas ligações. Se duas unidades estão disparando em modo síncrono, então elas estão representando predicacões sobre a mesma entidade. Esta técnica é chamada **temporal synchrony variable binding**.

Por exemplo: dado que temos as informações: quadrado, triângulo, listrado e pontilhado. Para representar em termos simbólicos o fato de que o quadrado que temos é listrado e o triângulo é pontilhado usamos a notação:

$$\exists x, \exists y, \text{listrado}(x) \wedge \text{quadrado}(x) \wedge \text{pontilhado}(y) \wedge \text{triangulo}(y)$$

Na arquitetura proposta, a representação da ligação de tais predicados é feita através dos pulsos (Figura 3.7). Diferentes predicados são representados com diferentes unidades (neurônios). Se duas unidades estão pulsando ao mesmo tempo, então elas estão representando predicacões sobre as mesmas coisas. Assim, de acordo com a figura, para alguma coisa que não sabemos (x), temos a informação de que é listrado e também quadrado.

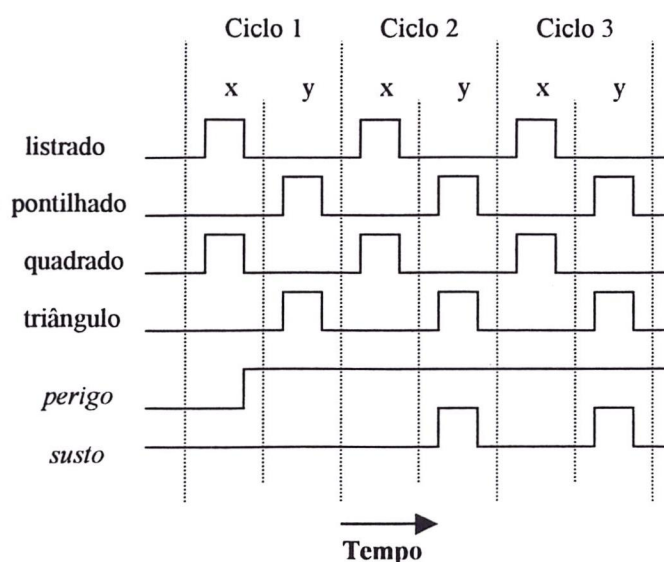


Figura 3.7. Representação de TSVB e a aplicação sequencial de duas regras.

As informações representadas em um padrão temporal de ativação podem ser modificadas de um período para outro usando as ligações entre as unidades. Como em outras arquiteturas conexionistas, o fluir da ativação através das ligações produz novos padrões de ativação para as unidades. Os padrões das ligações têm o efeito de testar o padrão de ativação, e modificá-lo se um casamento é encontrado.

Dessa maneira, as ligações (links) implementam *regras*. Devido ao fato de a arquitetura fazer um ciclo pelas variáveis, suas regras generalizam sobre elas. Pela mesma razão, as ligações têm em cada acesso à informação sobre uma variável de cada vez.

Por exemplo, para representar regras nos moldes simbolistas temos:

$$\forall x, \text{listrado}(x) \wedge \text{quadrado}(x) \Rightarrow \text{perigo}$$

$$\forall x, \text{perigo} \wedge \text{triangulo}(x) \Rightarrow \text{susto}(x)$$

Na arquitetura conexionista, podemos observar os pulsos dos predicados *perigo* e *susto* e a aplicação das regras no decorrer do tempo (Figura 3.7).

3.3.2.5. Parâmetros Aprendidos versus Variable Binding (VB)

A definição de sistematicidade requer que a rede faça uso de informações sobre as palavras, sobre as posições sintáticas, e sobre os pares de palavras com posições sintáticas. Os pares são manifestados nas sentenças que são entradas da rede e tal tarefa força que esses pares sejam manifestados na saída da rede.

Essa informação sobre esses pares deve ser comunicada da entrada para a saída, e conseqüentemente, deve ser representada de alguma maneira pelo padrão de ativação da rede. Essa é uma instância do problema de VB. A rede deve representar a ligação entre a informação sobre uma palavra e a informação sobre a posição sintática daquela palavra.

Em abordagens tradicionais, essa ligação é representada através de variáveis. Num exemplo simplificado, a ligação de “Maria” com a posição sintática de sujeito em “Maria ama João” poderia ser representada por $\text{Maria}(x) \wedge \text{sujeito}(x)$.

As exigências de sistematicidade aplicadas na representação de VB são bastante diferentes das exigências aplicadas na representação de palavras e posições sintáticas. Para que a rede generalize de um processamento de um conjunto de sentenças para o processamento de outro conjunto, os parâmetros que são determinados usando o conjunto de treinamento (isto é, os

pesos das ligações) têm que representar informação que também é verdadeira para o conjunto de teste. Para uma sistematicidade forte, tanto a informação aprendida sobre as palavras quanto a informação aprendida sobre as posições sintáticas devem ser verdadeiras no conjunto de teste.

Em contraste, devido ao fato de as palavras e as posições sintáticas serem unidas de forma diferente nos conjuntos de treinamento e teste, qualquer informação aprendida sobre as VBs não será verdade no conjunto de teste. Desde que a tarefa que requer informação sobre palavras e posições sintáticas seja aprendida corretamente, os parâmetros que representam essa informação não devem ser dependentes de VB.

3.3.2.6. Representando as Variable Bindings

Segundo o autor, a técnica de TSVB apresentada por Shastri & Ajjanagadde (1993) é atualmente a única proposta para o uso de dimensão de tempo para representar VBs. Nesse modelo, uma VB é representada pela especificação do período de tempo durante o qual o processamento envolvendo a variável deve acontecer. Se duas unidades (neurônios) estão representando informação sobre a mesma variável, então eles produzem ativação de saída ao mesmo tempo (ou seja, de maneira sincronizada).

Então, se uma dada palavra está em uma dada posição sintática, os neurônios que a representam produzem ativação de saída ao mesmo tempo em que os neurônios que representam a posição sintática.

O processamento ocorre quando essa ativação flui através das ligações da rede. Independentemente do período de tempo em que o processamento ocorre, serão usados os mesmos pesos das ligações.

Assim, não importa em qual VB uma palavra ou posição sintática participa, os mesmos parâmetros aprendidos serão aplicados a ela. Em outras palavras, os parâmetros aprendidos (pesos) são independentes das VBs. A informação que a rede aprendeu sobre uma palavra ou uma posição sintática em um conjunto de pares palavra-posição será aplicada automaticamente à mesma palavra ou posição sintática em diferentes pares.

Por esse motivo o uso de TSVB resulta em uma rede que generaliza informação sobre palavras de uma posição sintática para outra, e exibe uma forte sistematicidade.

3.3.2.7. Redes Sincronizadas Simples

A rede SSN é formada pela combinação de dois elementos, dos quais o primeiro é uma rede recorrente (SRN) (Elman, 1990).

O segundo elemento usado é Temporal Synchrony Variable Binding (TSVB) (Shastri & Ajjanagadde, 1993), cuja idéia essencial é que as unidades existentes dentro de cada ciclo da rede podem *pulsar*. Essas unidades que pulsam podem ser usadas para fazer um ciclo através de um conjunto de entidades. O SSN usa o elemento SRN para aprender sobre os padrões através do tempo, e o elemento TSVB para representar entidades; ambos aspectos importantes do aprendizado de língua requerido para esses experimentos.

O SSN usa duas SRNs (com uma camada de saída combinada), conforme ilustra a Figura 3.8. Uma das SRNs é feita de unidades que não pulsam, e o processamento procede como descrito acima. A outra SRN é feita de unidades que pulsam. O processamento para a SRN que pulsa procede como uma SRN padrão, exceto que o fluxo de ativação é computado para *cada* entidade. Em outras palavras, o processamento é repetido por um número fixo de vezes em cada ciclo, uma vez para cada entidade.

As ativações das unidades ocultas (hidden) das duas SRNs são ambas transferidas para a camada de combinação das unidades pulsantes. Esta combina os valores de ativação da SRN não pulsante com os valores de ativação de cada entidade da SRN pulsante. A ativação então flui para as unidades de saída, que computam ativações de saída para cada entidade.

O algoritmo para treinamento de SSN é baseado na extensão do Backpropagation Through Time (BPTT) (Rumelhart et al., 1986). Quando treinando uma rede recorrente, BPTT começa por fazer uma cópia da rede todo o tempo, para cada passo na sequência.

O resultado é uma rede *feed-forward*, com os links recorrentes carregando ativação entre cópias da rede nos diferentes tempos. Essa rede é então treinada usando o algoritmo *backpropagation* padrão, com mudanças de pesos na rede de acordo com a diferença entre a saída obtida e a saída desejada. A diferença com BPTT é que cada ligação tem que ser copiada, para cada ciclo. Quando treinando, cada mudança no peso de uma cópia das ligações deve ser feita para outras cópias.

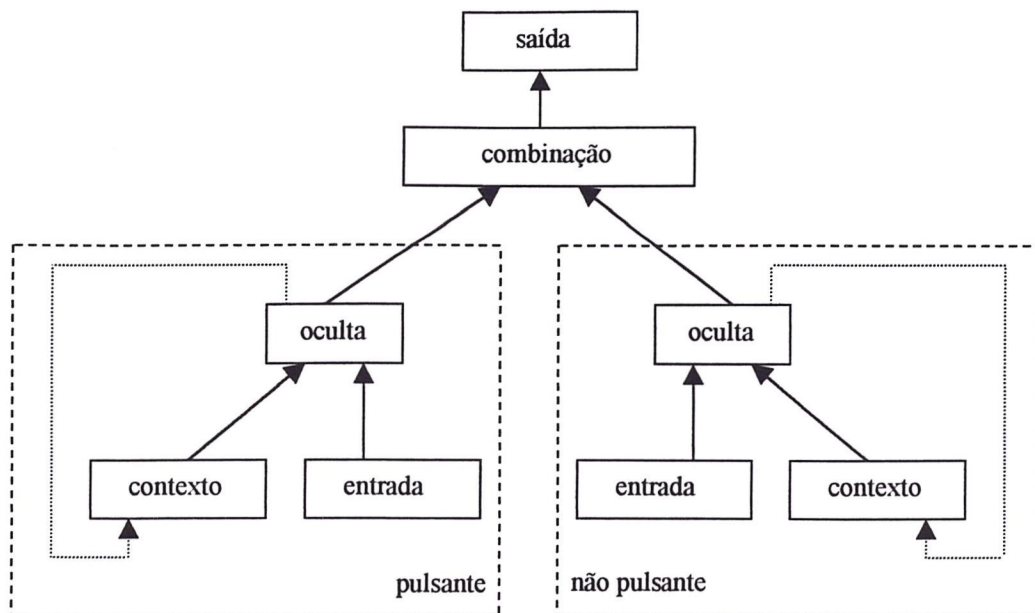


Figura 3.8. A Rede Sincronizada Simples (Lane, 1998).

BPTT pode ser estendida para redes usando TSVB fazendo-se uma cópia da rede tanto para cada passo na seqüência quanto para cada entidade usada. Logo, as unidades da SRN não pulsantes serão copiadas uma vez por cada ciclo como com BPTT padrão, e aquelas na SRN pulsantes serão copiadas uma vez por ciclo, mas uma vez por entidade.

O resultado é novamente uma rede *feed-forward*, exceto que agora há mais cópias das ligações (links) associadas com unidades pulsantes. Novamente, mudanças de peso devem ser feitas para cada cópia de uma ligação.

Como exemplo da representação das informações de entrada e saída, vide estrutura na Figura 3.9.

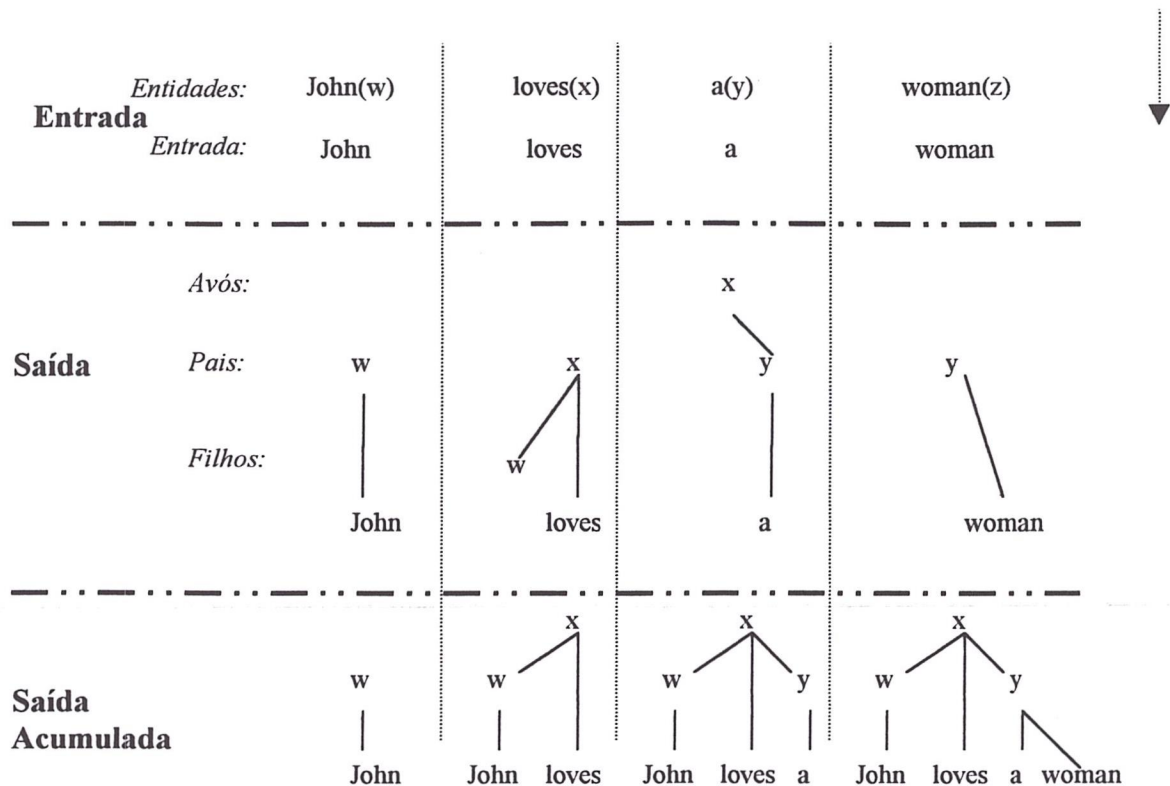


Figura 3.9. Representação das informações de entrada e saída para a sentença "John loves a woman". As fases são mostradas como variáveis (w, x, y, e z).

3.3.2.8. Vantagens

O uso de TSVB em uma rede conexionista tem duas importantes conseqüências:

- A saída de rede pode representar uma estrutura. As ativações das unidades de saída em uma fase podem representar informações sobre um constituinte em particular na estrutura, incluindo seu relacionamento estrutural com outros constituintes;
- As redes TSVB generalizam a informação aprendida sobre um constituinte para outros constituintes. Devido ao fato de os diferentes períodos de tempo (fases) serem usados para representar diferentes constituintes, e os mesmos pesos das ligações serem usados a todo o tempo, a mesma informação aprendida é aplicada para cada constituinte.

3.3.2.9. Conclusão

Segundo os autores, o modelo proposto estende as abordagens conexionistas quando fazem com que a ligação temporal das variáveis em sincronia represente os constituintes de uma maneira que permita que regras generalizem sobre eles. Devido a essa habilidade, a arquitetura pode levar vantagem sobre a natureza estrutural da língua natural, enquanto mantém as propriedades que fazem dos modelos conexionistas ferramentas importantes para o processamento cognitivo.

No entanto, a arquitetura tem limitações que impõem restrições computacionais no processo de análise sintática. Mas, apesar das limitações, a proposta se apresenta bastante satisfatória para tal propósito.

4. Conclusões

O uso de ferramentas de revisão gramatical de textos já é hoje uma realidade que faz parte do cotidiano da grande maioria dos usuários de computadores.

Várias ferramentas estão à disposição no mercado, inclusive o revisor ReGra, cujo propósito é de ser uma ferramenta de uso geral trabalhando com domínios irrestritos.

No entanto, poucas são as pesquisas acadêmicas que visam ao desenvolvimento de sistemas com tais características. O que mais se aproxima são as propostas de sistemas de *parsing*, que podem servir como base para o desenvolvimento das ferramentas de revisão gramatical.

As propostas atuais para *parsing* são as mais variadas, utilizando, em geral, métodos empíricos que extraem automaticamente o conhecimento necessário a partir de grandes corpora de exemplos.

Por outro lado, temos o revisor ReGra, que é construído nos moldes tradicionais, uma vez que necessita da codificação manual das regras que utiliza. Para a construção e validação de tais regras, são utilizadas sentenças contidas no corpus descrito nas seções anteriores. Essas características fazem com que o revisor seja uma ferramenta bastante robusta e eficiente.

Contudo, existem ainda vários pontos falhos no revisor, não se sabe se por conflitos existentes na complexa interdependência entre as regras, ou se por dificuldade de tratamento de determinadas estruturas sintáticas da língua portuguesa.

É natural, portanto, que nosso próximo trabalho seja a investigação dos pontos fortes das abordagens utilizadas atualmente nos sistemas de inferência automática, como os modelos estatísticos de Charniak (1993, 1997), as redes sincronizadas simples de Handerson (1994, 1996, 1998), os modelos de aprendizado através de programação lógica indutiva de Zelle & Mooney (1996a, 1996b) e vários outros modelos, no sentido de viabilizar o uso das características empíricas para a construção de revisores gramaticais, convergindo numa proposta de modelo empírico, ou híbrido, desses revisores. Estudos de caso e protótipos serão construídos para a revisão gramatical do português do Brasil.

Referências

- Aluísio, S. M. (2000). Construção de um Etiquetador para o Português do Brasil. Relatório Técnico do ICMC (em publicação).
- Allen, J. F. (1995). *Natural Language Understanding*. Menlo Park, Calif.: Benjamin /Cummings
- Allen, R. B. (1983). Sequential connectionist networks for answering simple questions about a microworld. In *5th Annual Proceedings of the Cognitive Science Society*, 489-495.
- Back, A. & Tsoi, A. (1991), 'FIR and IIR synapses, a new neural network architecture for time series modeling'. In *Neural Computation* 3(3), 375-385.
- Baião, M. A. S. G. C. (1994). Analisador por Grafo: superação de incorreções na escrita do Português. Tese de Mestrado. Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, Departamento de Informática. Lisboa, Portugal.
- Baker, J. (1979). Trainable grammars for speech recognition. In *Speech communication papers present at the 97th Meeting of the Acoustical Society of America*.
- Bonfante, A. G. & Nunes, M. G. V. (1999). Connectionist Learning Applied to a Brazilian Portugues Grammar Checking. In *Proceedings of the First Workshop on Natural Language Processing and Neural Networks (NLPNN'99)*, Beijing, China, Nov-99: 9-13.
- Bonfante, A. G. (1997). Uso de Redes Neurais para a Correção Gramatical do Português: Um Estudo de Caso. Tese de Mestrado, ICMC, USP-São Carlos.
- Brill, E. & Mooney, R. J. (1997). An Overview of Empirical Natural Language Processing. *AI Magazine*, 13-24, Winter 97.
- Brill, E. (1995). Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, 21(4): 543-565.

- Brill, E. (1993) Automatic Grammar Induction and Parsing Free Text: A Transformation-Based Approach. *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, 259-265. Columbus, Ohio.
- Charniak, E. (1999). A Maximum-Entropy-Inspired Parser. Technical Report CS99-12 Department of Computer Science, Brown University.
- Charniak, E.; Goldwater, S. & Johnson, M. (1998). Edge-based best-first chart parsing. In *Proceedings of the Sixth Workshop on Very Large Corpora*.
- Charniak, E. (1997). Statistical Parsing with a Context-free Grammar and Word Statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*.
- Charniak, E. (1997b) Statistical techniques for natural language parsing. In *AI Magazine*.
- Charniak, E. (1993) *Statistical Language Learning*. MIT Press.
- Chen, C. & Honavar, V. (1997). A Neural Network Architecture for Syntax Analysis. Feb. 9.
- Chomsky, N. A. (1986). *Knowledge of Language: Its Nature, Origin, and Use*. Prager.
- Christiansen, M. H. & Chater, N. (1999). Dynamical models of sentence processing. In *Special Issue of Cognitive Science – Connectionist Models of Human Language Processing: Progress and Prospects*. Vol. 23(4). Eds. Morten H. Christiansen, Nick Chater & Mark S. Seidenberg.
- Christiansen, M. H. (1998). Improving and Generalization in Neural Networks through the Acquisition of Multiple Related Functions. In *Proceedings of the Fourth Neural Computation and Psychology Workshop: Connectionist Representations*. Eds. J.^a Bullinaria, D.G. Glasspool & Houghton. London, UK, Springer-Verlag.
- Christiansen, M. H. & Chater, N. (1994). Generalization and Connectionist Language Learning. In *Mind and Language*, 9, 273-287.
- Church, K. & Mercer, R. L. (1993). Introduction to the special issue on computational linguistics using large corpora. *Computational Linguistics*, 19 (1), 1-24.
- Clouse, D.; Giles, C. L.; Horn, B. & Cottrell, G. (1994). Learning Large DeBruijn automata with feed-forward neural networks. Technical Report CS94-398, Computer Science and Engineering, University of California at San Diego, La Jolla, CA.
- Collins, M. (1999). Head-driven Statistical Models for Natural Language Parsing. Tese de Doutorado, Universidade da Pennsylvania.
- Dell, G. S.; Chang, F. & Griffin, Z. M. (1999). Connectionist Models of Language Production: Lexical Access and Grammatical Encoding. In *Special Issue of Cognitive Science – Connectionist Models of Human Language Processing: Progress and Prospects*. Vol. 23(4). Eds. Morten H. Christiansen, Nick Chater & Mark S. Seidenberg.
- DTS Software Ltda. (1995). Manual do Revisor DTS Windows. Rio de Janeiro: DTS.
<http://www.dts.com.br/>

- Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179-211.
- Elman, J. L. (1984). Structured representations and connectionist models. In *6th Annual Proceedings of the Cognitive Science Society*, 17-25.
- FLiP 97 (1999). FLiP – Ferramentas para a Língua Portuguesa. Editora Porto / Priberam Informática. <http://www.portoeditora.pt/>
- Frasconi, P.; Gori, M.; Maggini, M. & Soda, G. (1992). Unified integration of explicit rules and learning by example in recurrent networks. In *IEEE Transactions on Knowledge and Data Engineering*.
- Giles, C. L.; Horn, B. & Lin, T. (1995). Learning a class of large finite states machines with a recurrent neural networks, *Neural Networks*, 8(9), 1359-1365.
- Hadley, R. F. (1994). Systematicity in connectionist language learning. In *Mind and Language*, 9(3):247-272.
- Harris, C. L. & Elman, J. L. (1984). Representing variable information with simple recurrent networks. In *6th Annual Proceedings of the Cognitive Science Society*, 635-642.
- Hemphill, C.; Godfrey, J. & Doddington, G. (1990). The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA Speech and Natural Language Workshop*.
- Henderson, J. & Lane, P. (1998a). A connectionist architecture for learning to parse. In *Proceedings of the Association of Computational Linguistics*.
- Henderson, J. & Lane, P. (1998b). Simple Synchrony Networks: Learning to Parse Natural Language with Temporal Synchrony Variable Binding. In *Proceedings of ICANN 1998*, Skövde, Sweden.
- Henderson, J. (1996). A connectionist architecture with inherent systematicity. In *Proceedings of the Eighteenth Conference of the Cognitive Science Society*, 574-579, La Jolla, CA.
- Henderson, J. (1994). Connectionist syntactic parsing using temporal variable binding. *Journal of Psycholinguistic Research*, 23(5):353-379.
- Hermjakob, U. & Mooney, R. J. (1997). Learning Parse and Translation Decisions from Examples with Rich Context. In: *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics*, 482-489. Somerset, N. J.: Association for Computational Linguistics.
- Horne, B. G. & Giles, C. L. (1995). An experimental comparison of recurrent neural networks. In *Neural Information Processing Systems 7*, eds. G. Tesauero, D. Touretzky, T. Leon, p.697, MIT Press.
- Itautec Softwares (1999). Redação Língua Portuguesa. Itautec. <http://www.itautech.com.br/produtos/soft/rlp/>
- Lane, P. (1998). Simple Synchrony Networks: Learning Generalisations across Syntactic Constituents. In *Proceedings of ICANN 1998*, Skövde, Sweden.
- Lawrence, S.; Fong, S. & Giles, C. L. (1996). On the Applicability of Neural Network and Machine Learning Methodologies to Natural Language Processing. In *Workshop on New Approaches to Learning for Natural Language Processing, IJCAI-95*, Montreal, Canadá, 1-8.

- Lexicon Informática (1999). Gramática Eletrônica. Lexicon Informática Ltda.
<http://www.lexikon.com.br/gramat/gramat.htm>
- Lins, R. D.; Camelo, H. A. L.; Moura, R. S. (1999) Um SOS para a Língua Portuguesa. *IV Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR-99)*. 115-128.
- Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 276-283.
- Manning, C. D. & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts.
- Mangu, L. & Brill, E. (1997). Automatic Rule Acquisition for Spelling Correction. In *Proceedings of ICML*.
- Marcus, M.; Santorini, B. & Marcinkiewicz, M. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2), 313-330.
- Marcus, M. P. (1980). *A Theory of Syntactic Recognition for Natural Language*. MIT Press.
- Martins, R. T.; Rino, L. H. M.; Montilha, G.; Nunes, M. G. V. (1999). Dos modelos de resolução da ambigüidade categorial: O problema do SE. *IV Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR-99)*. 115-128.
- Martins, R. T.; Hasegawa, R.; Nunes, M. G. V.; Montilha, G.; Oliveira Jr., O. N. (1998) Linguistic Issues in the development of ReGra: a Grammar Checker for Brazilian Portuguese. *Natural Language Engineering*, 4, 287-307, Cambridge University Press.
- Miikkulainen, R. (1993). *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory*. MIT Press, Cambridge, MA.
- Mooney, R. (1996). Inductive Logic Programming for Natural Language Processing. In *Proceedings of the Sixth International Inductive Logic Programming Workshop on Logic Programming*, Stockholm, Sweden, Springer Verlag.
- Moura, R. S. (1996). SOS: um sistema Orto-Sintático para a Língua Portuguesa. Tese de Mestrado, Universidade Federal de Pernambuco, Recife.
- Nunes, M. G. V.; Vieira, F. M. C.; Zavaglia, C.; Sossolote, C. R. C.; Hernandez, J. (1996) A construção de um Léxico para o português do Brasil. In *II Workshop em Processamento do Português Escrito e Falado*, 61-70, Curitiba, PR.
- Pardo, T. A. S. & Rino, L. H. M. (2000). Manual do ReGra – Revisor Gramatical. Relatório Técnico do ICMC (em publicação).
- Parekh, R. & Honavar, V. (1996). Grammar Inference, Automata Induction and Language Aquisition. In *CGI96*.

- Pereira, F. & Schabes, Y. (1992). Inside-Outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for the Computational Linguistics*, Newark, De.
- Pollack, J. B. (1990). Recursive Distributed Representations. In *Artificial Intelligence* 46: 1, 77-105.
- Reference Software (1995). SNNS Stuttgart Neural Network Software. V4.1. User's Guide.
- Reference Software International (1992). The #1 Grammar & Style Checker Gram.Mat.Ik 5 (for windows) User's Guide. San Francisco, CA, USA.
- Reilly, R. G. & Sharkey, N. E. (Eds.) (1992). *Connectionist Approaches to Natural Language Processing*. Lawrence Erlbaum and Associates, Hillsdale, NJ.
- Rumelhart, D. E.; Hinton, G. E. & Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing*, Vol.1, Eds. D.E. Rumelhart and J.L. McClelland, MIT Press, Cambridge, MA.
- Satta, G. & Brill, E. (1996). Efficient Transformation-Based Parsing. In *Proceedings of the Association for Computational Linguistics*.
- Seidenberg, M. S. & MacDonald, M.C. (1999). A Probabilistic Constraints Approach to Language Acquisition and Processing. In *Special Issue of Cognitive Science – Connectionist Models of Human Language Processing: Progress and Prospects*. Vol. 23(4). Eds. Morten H. Christiansen, Nick Chater & Mark S. Seidenberg.
- Shastri, L. & Ajanagadde, V. (1993). From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*.
- Smolensky, P. (1999). Grammar-based Connectionist Approaches to Language. In *Special Issue of Cognitive Science – Connectionist Models of Human Language Processing: Progress and Prospects*. Vol. 23(4). Eds. Morten H. Christiansen, Nick Chater & Mark S. Seidenberg.
- Soderland, S. G. & Taira, R. K. (1999). A Field Theory Approach to Natural Language Parsing. Unpublished.
- St. John, M. F. & McClelland, J.L. (1990). Learning and Applying Contextual Constraints in Sentence Comprehension. In *Artificial Intelligence*, 46:217-257.
- Steedman, M. (1999). Connectionist Sentence Processing in Perspective. In *Special Issue of Cognitive Science – Connectionist Models of Human Language Processing: Progress and Prospects*. Vol. 23(4). Eds. Morten H. Christiansen, Nick Chater & Mark S. Seidenberg.
- Stolcke, A. (1990). Learning feature-based semantics with simple recurrent networks. Technical Reports TR-90-015, International Computer Science Institute, Berkeley, California.
- Sun-Microsystems Manual – Lex – a Lexical Analyser Generator. Chapter 9.
- Sun-Microsystems Manual – Yacc – Yet Another Compiler-Compiler. Chapter 10.

- Tabor, W. & Tanenhaus, M.K. (1999) Dynamical models of sentence processing. In *Special Issue of Cognitive Science – Connectionist Models of Human Language Processing: Progress and Prospects*. Vol. 23(4). Eds. Morten H. Christiansen, Nick Chater & Mark S. Seidenberg.
- Waltz, D. L. (1978). Na English Language Question-Answering System for a Large Relational Database. *Communications of the Association for Computing Machinery*, 21(7): 526-539.
- Wermter, S.; Rillof, E. & Scheler, G. (1996). *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*. Springer, Heidelberg, New York, March.
- Williams, R. & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks, *Neural Computation*, 1(2), 270-280.
- Winograd, T. (1970). *Understanding Natural Language*. San Diego, Calif.: Academic.
- Woods, W. A (1977). Lunar Rocks in Natural English: Explorations in Natural Language Question Answering. In *Linguistic Structures Processing*, eds. A Zampoli. New York: Elsevier.
- Woods, W. A. (1970) A Transition Network Grammars for Natural Language Analysis. In: *CACM, Special Issue on Computational Linguistics*, 13(2), 591-606.
- Writing Tools Group, Inc. (1991). *Using Correct Grammar for Windows*. Sausalito, CA, USA.
- Zelle, J. M. & Mooney, R. J. (1996a.). Learning to Parse Data Base Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1050-1055. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Zelle, J. M. & Mooney, R. J. (1996b.). Comparative Results on Using Inductive Logic Programming for Corpus-based Parser Construction. In *Symbolic, Connectionist, and Statistical Approaches to Learning for Natural Language Processing*, Springer-Verlag.