Analisador Sintático Estatístico Orientado ao Núcleo-Léxico Para a Língua Portuguesa

Fabiano de Carvalho e Sousa

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO GRAU DE MESTRE
EM
CIÊNCIA DA COMPUTAÇÃO

Área de Concentração: **Ciência da Computação**
Orientador: **Prof. Dr. Marcelo Finger**

São Paulo - outubro – 2003

Analisador Sintático Estatístico Orientado ao Núcleo-Léxico Para a Língua Portuguesa

Este exemplar corresponde à redação
final da dissertação devidamente
corrigida e defendida por
Fabiano de Carvalho e Sousa
e aprovada pela comissão julgadora.

São Paulo, outubro de 2003.

Banca examinadora:

- Prof. Dr. Marcelo Finger (Orientador), IME-USP
- Prof. Dra. Renata Wassermann, IME-USP
- Prof. Dra. Maria das Graças Volpe Nunes, ICMC-USP

Resumo

O desenvolvimento do Corpus Tycho Brahe do Português histórico anotado sintaticamente motivou a criação de uma ferramenta automática para a geração das árvores sintáticas das sentenças de um texto. Com este fim, utilizamos um método já desenvolvido e aplicado para a mesma tarefa aplicada ao Inglês, que utiliza um modelo estatístico orientado ao núcleo-léxico através da definição das dependências léxicas presentes na sentença. Neste trabalho, estudamos tal método aplicado à língua Portuguesa, chegando até a implementação de dois sistemas automáticos para análise sintática. O primeiro utiliza o modelo baseado em dependências léxicas e o segundo baseia-se em regras gramaticais livre de contexto probabilísticas. Desta forma, esperamos não só auxiliar o processo de anotação sintática no corpus Tycho Brahe como também apresentar os primeiros resultados comparativos de acurácia do método escolhido aplicado ao Português.

Abstract

The construction of the Tycho Brahe Corpus of Historical Portuguese, in which texts are going to be displayed in their original format and in a syntactically parsed format, has served as motivation for the development of an automated tool for parsing Portuguese texts. For that, we have applied a parsing method based on a probabilistic model lexical-head driven that incorporates lexical dependencies. Originally developed for English, this model has been adapted to Portuguese, leading to the implementation of two automated-learning parsing systems. One is based on a model with a probabilistic context free grammar and the other is enhanced with lexical dependencies. A comparison between the two methods is presented; the programs implementing these methods were trained with the initial sections of the manually annotated parsed texts of the Tycho Brahe Corpus.

Oferecimento

A Deus, toda honra e toda glória,
agora e para sempre, por todos
os séculos dos séculos...

Agradecimentos

Agradeço ao meu orientador, ***Dr. Marcelo Finger****, por sua ajuda, paciência e confiança.*

À ***equipe de Sintaxe Histórico do IEL/UNICAMP****, pelo material utilizado no trabalho.*

Aos ***professores de mestrado do IME/USP*** *pelos ensinamentos passados.*

Aos meus pais**, por tudo que me ofereceram.**

À minha esposa **e aos** nossos filhos**, por tudo que são.**

Ao pessoal do escritório**, pela força e compreensão.**

Ao Pe Joselino **e** Pe. Ângelo**, por seus conselhos e incentivos.**

A todos que **de alguma forma** colaboraram **com a realização deste trabalho, principal-mente com suas orações.**

Também agradeço à ajuda dos santos**, em particular** São Josemaría Escrivá **e de** Maria, Nossa Senhora**, minha mãe do céu.**

E finalmente, ao misericordioso Sacratíssimo Coração de Jesus **junto a** Deus **por todas as graças derramadas em minha vida, em especial às que ocorreram no período do meu programa de mestrado.**

Sumário

Capítulo 1
Introdução

1.1 Descrição do problema

Sistemas de extração de informação, tradução automática e reconhecimento de fala são exemplos de aplicações relacionadas a processamento lingüístico computacional. Tais aplicações dependem em parte de um *analisador sintático*[1], cuja acurácia influencia diretamente a qualidade dos resultados obtidos naquelas tarefas. A função de um analisador sintático é basicamente associar uma estrutura de árvore sintática a uma dada sentença de entrada. O algoritmo que realiza tal objetivo pode ser tanto determinístico como probabilístico.

Algoritmos determinísticos consistem na aplicação em série de regras gramaticais viáveis sobre a sentença de entrada, transformando-a na árvore sintática final. Essas regras gramaticais viáveis pertencem a um conjunto finito fixo geralmente construído a mão por um especialista. O maior problema de tais sistemas é a geração de mais de uma árvore para uma sentença de entrada. Neste caso, não existe regra bem definida para se escolher qual é a árvore de resposta. Esta e outras falhas incentivaram o desenvolvimento de algoritmos probabilísticos para o problema, onde o critério de escolha faz parte do próprio modelo matemático usado.

Algoritmos probabilísticos constroem a função de transformação a partir de um conjunto de treinamento que consiste em pares de exemplos 'sentença-árvore sintática correspondente'. Um outro conjunto de pares de exemplos, significativamente menor que o anterior, é usado para teste e avaliação da acurácia do algoritmo. Este conjunto é chamado de conjunto de teste. Algoritmos probabilísticos usam métodos estatísticos para estimar valores de parâmetros a partir dos dados de treinamento. Um exemplo de parâmetro é a probabilidade de aplicação de uma determinada regra gramatical dentro de um determinado contexto.

A maior parte da literatura e dos trabalhos realizados na área são destinados à língua inglesa. Este trabalho visa criar um analisador sintático para o caso específico do Português. Deste modo, não há garantias de que qualquer método desenvolvido até então para o Inglês funcionará com boa acurácia para o Português. Mas ao mesmo tempo, também não há garantias de que tal método funcionará mal. O que seguramente muda são os parâmetros usados pelo algoritmo. Portanto, pode-se aproveitar a experiência existente na área em Inglês, para se adaptar um método relativamente bom para o Português.

---

[1] Em inglês *Natural Language Parsing.*

Da mesma forma, alguns exemplos apresentados neste trabalho, apesar de serem em Português, utilizam as etiquetas sintáticas e regras estruturais definidas em trabalhos orientados para a língua inglesa. Isso não invalida o efeito ilustrativo dos exemplos e contorna o problema de que ainda não existe um conjunto de etiquetas sintáticas e regras estruturais bem definidas e completo para o Português. Tal trabalho de especificação está em fase de conclusão pela equipe de Sintaxe Histórico do IEL/UNICAMP.

O método a ser adaptado neste trabalho é o método probabilístico orientado ao núcleo-léxico que foi desenvolvido por Michael Collins, em sua tese de doutorado em Ciência da Computação na Universidade da Pennsylvania em 1999 [Collins 99]. Este método além de ser um dos mais recentes, tem um grau de acurácia significativamente superior aos demais existentes, como as gramáticas livres de contexto probabilísticas[HU 79] e *SPATTER* [Magerman 95]. Infelizmente esses índices de precisão não são exatos, pois dependem tanto da base de dados usada para o treinamento como para teste, que são diferentes de um método para o outro.

O método escolhido tenta realizar a tarefa a partir das informações léxicas mais relevantes presentes na sentença. Isso é feito através de uma parametrização do problema que considera dois pontos críticos:

- Com quais objetos lingüísticos os parâmetros devem estar relacionados? Como as árvores devem ser quebradas em fragmentos menores?
- Como essa escolha pode ser feita para se obter um modelo estatístico?

Esse método pode ser aplicado para o caso do Português, com especial cuidado para a escolha e adaptação dos objetos lingüísticos.

1.2 Motivação

A motivação mais imediata é auxiliar no desenvolvimento do córpus[2] Tycho Brahe para um projeto que pretende estudar a evolução do Português Clássico (PEC) para o Português Europeu Moderno (PEM) [AlvFinger 99]. Os textos utilizados neste projeto são de autores nascidos no período desde o século XVI até XIX. Com a construção dos textos sintaticamente anotados, será possível, por exemplo, estabelecer como a mudança prosódica dos padrões rítmicos do Português levou a mudanças sintáticas, em especial na colocação dos pronomes clíticos.

[2] Utilizaremos nesta dissertação a tradução córpus própria para a língua portuguesa tanto para o termo *corpus* como o plural *corpora* comuns na literatura técnica em inglês.

Além desta aplicação imediata, existem pelo menos mais três que podem se beneficiar dos resultados deste trabalho: a) Sistemas de extração e recuperação de informação; b)Sistemas de tradução automática; c) Sistemas de reconhecimento da fala.

a) Um sistema de extração de informação poderia aplicar o analisador sintático sobre um conjunto textos de jornais extraídos por décadas, presentes em um banco de dados. Acrescentando-se informações léxicas de mapeamento sintático para regras semânticas, seria possível, por exemplo, extrair do sistema todos os pares *vencendor-perdedor*, onde vencedor e perdedor são jogadores de tênis. Um analisador sintático com o léxico dos verbos *vencer/perder* poderia realizar boa parte desta tarefa.

b) Um dos problemas mais sérios para sistemas de tradução automática está em como mapear a ordem de palavras de uma sentença de uma língua para a ordem correta em outra língua. Por exemplo, Português e Japonês tem ordem sintática de palavras bastante distintas, como na sentença abaixo:

Português: O jornalista disse que Guga venceu Sampras ontem

Japonês: Ontem Guga Sampras venceu o jornalista que disse

Tendo-se a estrutura gramatical, o mapeamento de ordem pode ser descrito por definições recursivas simples de regras gramaticais do Português para o Japonês. Assim, por exemplo, a regra da língua portuguesa (*VP* $\rightarrow$ *VBD NP*) seria mapeada para (*VP* $\rightarrow$ *NP VBD*). As etiquetas *NP, VP, VBD* presentes no exemplo são usadas no banco de árvores[3] da *Penn Wall Street Journal* e significam respectivamente *'noun phrase'*, *'verb phrase'* e *'past-tense verb'*.

c) Os sistemas de reconhecimento de fala mais modernos usam modelos lingüísticos de 'trigrama' para avaliar a probabilidade de uma sentença como o produto de termos, cada termo composto por uma janela de três palavras consecutivas. Por exemplo:

Sentença falada: Ele mora no Brasil e na Argentina

Sentença reconhecida: Ele mora no Brasil está na Argentina

A segunda frase está gramaticalmente errada, porém possivelmente possui uma classificação melhor que a original já que nenhuma de suas janelas de três palavras consecutivas (trigramas) possui probabilidade extremamente baixa de ocorrer independentemente. Em outras palavras, todas as janelas de três palavras da sentença reconhecida poderiam ter sido observadas no córpus de treinamento do sistema. Perceba que as duas sentenças candidatas possuem os mesmos trigramas, exceto os seguintes {*<no Brasil e>*, *<Brasil e na>*, *<e na Argentina>*} na primeira sentença e {*<no Brasil está>*, *<Brasil está na>*, *<está na Argentina>*} na segunda. Os

[3] Em inlês *tree banks*: coleção de pares sentenças e árvores sintáticas anotadas e corrigidas manualmente. Ver mais detalhes em *tree banks grammars* [Charniak 96].

dois conjuntos de trigramas possuem sua nota de probabilidade de ocorrência própria e isto será usado como único critério para decisão de qual sentença será reconhecida. Os trigramas da sentença errada não possuem nenhuma seqüência de 3 palavras incomuns e na prática seriam possíveis de se obtidos em um treinamento sobre um córpus suficientemente extenso. Portanto, a sentença errada tem boa chance de ser a reconhecida pelo critério de trigrama, pois este não tem em seu contexto de avaliação a construção gramaticalmente inviável gerada pelo tetragrama *<mora no Brasil está>* ou *<no Brasil está na>*. Esse contexto seria observado por exemplo em um treinamento sobre tetragramas, no qual tais ocorrências teriam notas de ocorrência bastante baixas. Entretanto, o problema ainda existe em tamanhos de janela maiores já que é possível se imaginar outros exemplos deste problema aplicados para o modelo de tetragrama ou pentagrama e superiores (sem mencionar o aumento do problema de dados esparsos, isto é, de que quanto maior a janela, mais dados de treinamento são necessários para cálculos confiáveis dos parâmetros do modelo).

Como será visto mais adiante, o analisador sintático deste trabalho também pode ser usado para classificar probabilisticamente seqüências de palavras dentro da língua. O objetivo inicial dele é descobrir a árvore sintática mais provável para a entrada. Isto é feito dando notas para todas as árvores significativas geradas para a sentença e retornando aquela com nota de probabilidade maior. Somando-se todas as notas de cada árvore gerada obtém-se a probabilidade da sentença ser gerada na língua. Em outros termos, sendo τ o conjunto de todas as árvores sintáticas possíveis no modelo usado, Σ o conjunto de todas as sentenças geradas pelo modelo e $\Im(S)$ o conjunto de árvores que geram a sentença S e que o analisador sintático é capaz de calcular $\wp(T|S)$, temos:

$$\sum_{T \in \tau} \wp(T) = 1 \qquad \sum_{S \in \mathbf{S}, \wp(S) \geq 0} \wp(S) = 1 \tag{1.1}$$

$$\wp(S) = \sum_{T \in \Im(S)} \wp(T \mid S), \quad \forall T \in \tau \text{ e } \forall S \in \Sigma \tag{1.2}$$

Uma pequena simplificação para efeitos de desempenho é usada na prática, para eliminar subestruturas de árvores que possuem notas muito baixas, abaixo de um limiar de descarte. Assim, há uma perda de massa de probabilidade não computada, mas em hipótese isto não deveria alterar a classificação obtida. Essa atribuição de probabilidade poderia corrigir o engano gramatical ocorrido no modelo de trigramas já que o algoritmo tem a vantagem de capturar dependências estatísticas entre palavras que não estão dentro de uma mesma janela.

1.3 Objetivo

O objetivo deste trabalho é desenvolver um analisador sintático para o Português com o maior grau de acurácia possível a fim de auxiliar no processo de anotação sintática dos textos do córpus Tycho Brahe.

O modelo a ser utilizado é uma adaptação do probabilístico orientado ao núcleo-léxico desenvolvido para o Inglês em [Collins 99]. Ao mesmo tempo foi possível avaliar o desempenho do modelo usado para o Português sob a representação lingüística proposta. O estudo levanta possíveis argumentos para os resultados obtidos e tenta verificar a validade das hipóteses originalmente usadas no modelo, como por exemplo, se existe também no Português o efeito da influência local dos núcleos-léxicos nas árvores sintáticas (Capítulo 4 apresenta esta e outras hipóteses utilizadas pelo modelo).

Os textos utilizados para treinamento e teste do analisador foram fornecidos por colaboração do grupo de Sintaxe Histórico do IEL/UNICAMP. O processo de anotação é iterativo; à medida que se aplica o analisador sintático sobre mais textos, esses são corrigidos e então usados para se treinar novamente e aumentar a acurácia do analisador. A cada iteração, o tamanho do córpus anotado sintaticamente aumenta e o algoritmo melhora sob certos limites.

1.4 Visão Geral da Dissertação

O Capítulo 2 apresenta alguns conceitos e resultados matemáticos utilizados como base de entendimento e implementação do analisador sintático orientado ao núcleo-léxico. Inicialmente ele descreve alguns tópicos relevantes como representação de relações sintáticas, o conceito de modelo bem-definido, processo de ajustagem e função de verossimilhança. Parte então para uma breve descrição de alguns modelos matemáticos para análise sintática que historicamente encaminharam para a elaboração do método orientado ao núcleo-léxico. E por fim, apresenta o algoritmo de Carta de Análise Ascendente que será posteriormente adaptado ao modelo final usado nesta dissertação. Todo este material é retirado de trabalhos feitos para a língua inglesa.

O Capítulo 3 descreve especificamente os dois trabalhos realizados por Collins utilizando a idéia de decompor a árvore sintática a partir da palavra principal de cada subestrutura. Esta idéia é fundamentada pela hipótese de que no Inglês os itens lexicais núcleos de uma sentença exercem uma influência localizada nos demais itens lexicais e portanto afetam diretamente um domínio limitado da estrutura da árvore. Apesar dos dois analisadores sintáticos de Collins possuírem resultados de desempenho bastante próximos entre si, o capítulo também apresenta as diversas diferenças existentes. Por fim, apresenta-se o que destes trabalhos desenvolvidos ao Inglês será aplicado para o Português nesta dissertação.

O Capítulo 4 é focado no córpus Tycho Brahe que é utilizado para treinamento e testes do trabalho desenvolvido nesta dissertação. Além de apresentar suas características, são levantados alguns pontos de análise que interferem na aplicação do método orientado ao núcleo-léxico sobre tal córpus. Aspectos como estilo de anotação, restrições do córpus, particularidades do Português, mudanças na representação das relações sintáticas e verificação de hipóteses do método utilizado são listados a fim de preparar o entendimento ao que foi implementado no Capítulo 5.

O Capítulo 5 descreve a arquitetura e implementação dos sistemas de análise sintática desenvolvidos nesta dissertação e aplicados para o córpus apresentado no Capítulo 4.

O Capítulo 6 apresenta os resultados obtidos pelos analisadores descritos no Capítulo 5 e esboça alguns pontos de análise que justificam o desempenho obtido.

No Capítulo 7 são realizadas as conclusões de todo o trabalhado e resultados descritos nos capítulos anteriores. Além disso, apresenta-se um conjunto de sugestões para trabalhos futuros nesta linha de desenvolvimento como por exemplo: 1) aprofundamento do estudo, 2) verificação de hipóteses, 3) desenvolvimento de ferramentas auxiliares ou de refinamento, 4) aplicação de esforço em heurísticas sobre o método desenvolvido e 5) a integração de novas técnicas para obtenção de resultados superiores.

Capítulo 2
Arcabouço

Este capítulo apresenta um resumo de conceitos e trabalhos realizados para a tarefa de analisadores sintáticos automáticos para o Inglês, mas com o foco orientado a preparar um mínimo de base para o entendimento do modelo orientado ao núcleo-léxico elaborado por [Collins 99]. Este entendimento engloba tanto a mecânica do algoritmo como o próprio histórico de trabalhos anteriores que levaram a sua maturação. Nesta área, assim como em outras correlatas (reconhecimento de fala, por exemplo) existem muitos outros trabalhos baseados em características lingüísticas e idéias originais que não interferem de forma direta no entendimento do modelo orientado ao núcleo-léxico, e portanto tais esforços não são considerados neste resumo. Os leitores que já estão familiarizados com métodos estatísticos para análise sintática podem seguir direto para o Capítulo 3.

Inicialmente serão apresentados os conceitos de representação e de analisador sintático, com destaque maior para as suas componentes de projeto principais: modelo e o algoritmo de aplicação. Tais componentes possuem aspectos relevantes para a determinação da qualidade do analisador. Após isso, será apresentado um breve histórico com os principais métodos probabilísticos já utilizados em trabalhos similares que de certa forma influenciaram para a elaboração do método base desta dissertação. Outros conceitos chaves como lexicalização de árvores, tipos de erros presentes nos modelos estatísticos, modelo bem-definido, verossimilhança, método de ajustagem de parâmetros e medidas de acurácia também serão apresentados à medida de sua necessidade para a compreensão dos pontos seguintes. Ao fim, é feita uma conclusão do capítulo.

2.1 Estruturas para Análise Sintática e sua Representação

A estrutura usada neste trabalho para codificar os vários níveis de informação sintática de uma sentença é chamada de árvore sintática. A figura 2.1 apresenta um exemplo desta estrutura, já com alguns elementos de representação usados no trabalho.

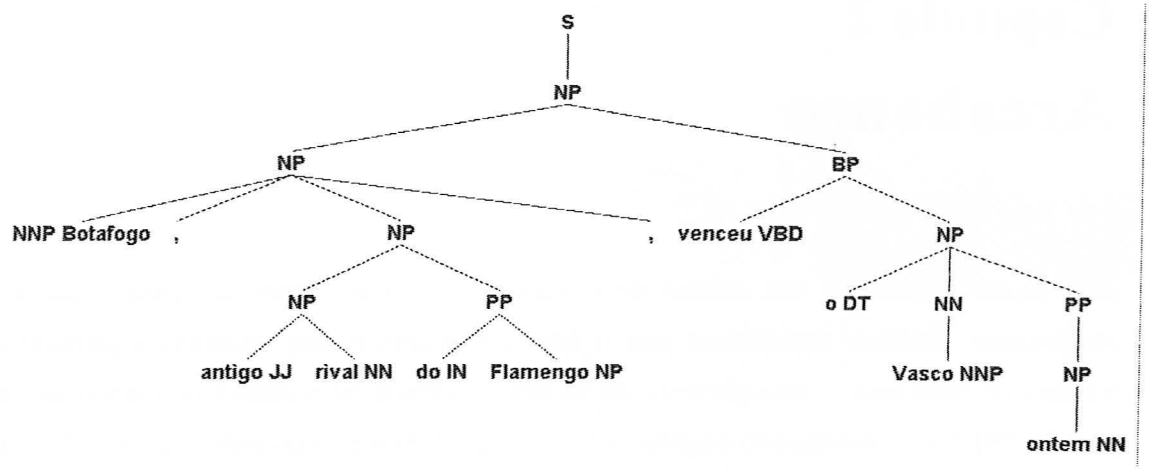


Figura 2.1: Um exemplo de árvore sintática

Trata-se de uma estrutura de árvore, comum em computação, na qual as informações relevantes são obtidas a partir das etiquetas dos constituintes, dos agrupamentos gerados em cada nível e das relações presentes nestes agrupamentos. No caso da figura 2.1, percebemos que os nós terminais possuem a informação das categorias morfossintáticas de cada palavra da sentença. Assim, '*Botafogo*' é um *NNP* (nome próprio), '*venceu*' é um *VBD* (verbo no passado) e '*antigo'* é um *JJ* (adjetivo). Também é possível ter-se a descrição sintática de agrupamentos de palavras, como por exemplo '*Botafogo, antigo rival do Flamengo*' é um *NP* (frase nominal). E por fim, pode-se obter diretamente as diversas relações predicados-argumentos presentes nos níveis da árvore. Por exemplo, na regra (*S NP VB*) sabe-se que o *NP* é o sujeito do verbo presente dentro de *VP* e portanto, por esta regra, '*Botafogo, antigo rival do Flamengo*' é o sujeito de '*venceu'*. Sendo '*Botafogo'* o núcleo do constituinte *NP* em questão, chega-se nele como o núcleo do sujeito. Outras relações como por exemplo verbo-objeto entre '*venceu'* e '*Vasco'* também podem ser retiradas da árvore.

Algumas decisões interferem no que se chama representação da árvore sintática, isto é, a escolha de como se representa tal árvore. Por exemplo, quais são as etiquetas morfossintáticas e os não-terminais possíveis nas árvores; existirá ou não núcleos-léxicos anexados a cada nó não-terminal; os itens lexicais são constituídos apenas pela palavra ou possuem outras informações anexadas como raízes morfológicas, flexão, sinônimos, ou códigos binários[4] derivados por técnicas de agrupamento.

No caso deste trabalho a representação consistirá nos elementos apresentados na figura 2.1, com o item lexical contendo a informação de sua etiqueta morfossintática e com passagem de núcleos-léxicos dos constituintes filhos para os pais (processo conhecido como lexicalização da árvore, descrito na seção 2.3). De fato, no método orientado ao núcleo-léxico, a árvore é codificada por um agrupamento de relacionamentos de dependência com 2 itens lexicais cada,

[4] Em inglês *binary string.*

um principal e o outro modificador. Cada arco da árvore entre etiquetas com núcleos-léxicos diferentes gera um destes relacionamentos. Isso será explicado no Capítulo 3, mas no momento é necessário saber que se trata de um mapeamento 1 para 1 entre árvores sintáticas e grupos de relacionamentos de dependências (considerando-se a restrição fixa de que assim como os arcos de uma árvore nunca se cruzam, os relacionamentos entre dois itens lexicais também não se cruzam entre si).

2.2 Analisador Sintático

Um analisador sintático tem como função determinar para uma dada sentença de entrada sua árvore de estrutura sintática. No caso desta dissertação, as palavras da sentença de entrada já estão rotuladas com suas respectivas classes gramaticais. Essa rotulação pode ser gerada automaticamente através de uma aplicação externa, chamada de *etiquetador de categorias morfossintáticas*[5], como por exemplo, a desenvolvida em [AlvFinger 99].

A tarefa pode ser tratada como um problema de aprendizado computacional supervisionado que tem a seguinte forma. Assumindo-se que a tarefa consiste em se aprender uma função f: $\chi \rightarrow \gamma$ onde χ é o conjunto das entradas possíveis, γ é o conjunto das saídas possíveis. Os dados de treinamento são um conjunto de n pares entrada e saída, $<x_1, y_1>$, ... $<x_n, y_n>$ onde $x_i \in \chi$, $y_i \in \gamma$, e $y_i = f(x_i)$.

Numa aproximação probabilística o problema é transformado de uma função diretamente aprendida f: $\chi \rightarrow \gamma$, para o aprendizado de uma função probabilística *Nota:* χ x $\gamma \rightarrow$ [0, 1]. *Nota(x, y)* pode ser tanto uma probabilidade condicional $\mathcal{P}(y|x)$ ou uma probabilidade conjunta $\mathcal{P}(x,y)$. Definido a função *Nota*, $f(x)$ podem ser definidos como o membro mais provável de γ sob esta distribuição de probabilidade:

$$f(x) = \arg\max_{y \in Y} Nota(x, y) \qquad (2.1)$$

Num modelo parametrizado existe um vetor Θ que é um argumento adicional da função *Nota*, ou seja a função passa a ser $Nota(x, y|\Theta)$. Esta função tem estrutura fixa e com todas suas possíveis variações descritas pelo espaço de parâmetros Ω, o espaço de possíveis valores para Θ. Neste caso, o problema de aprendizado consiste em se ajustar a estimativa do parâmetro $\hat{\Theta} \in \Omega$, a partir do conjunto de exemplos de treinamento.

Para o caso específico de analisadores sintáticos probabilísticos, o modelo consiste na escolha de uma parametrização de árvores, isto é, dividir o problema em partes menores para se

[5] Em inglês *part-of-speech tagger*.

estimar uma *Nota(A,S)* (A = árvore, S = sentença de entrada) de probabilidade para cada par árvore-sentença da linguagem. Escolhido o modelo, a árvore assimilada para uma dada sentença de entrada S é tal que maximiza a *Nota(A, S)* dentro do subdomínio definido, isto é: $A_{melhor}(S) = \arg\max_{A} Nota(A,S)$. *Nota(A,S)* pode ser tanto probabilidade conjunta $\mathcal{P}(A,S)$ como probabilidade condicional $\mathcal{P}(A|S)$.

Deste modo, o projeto de um analisador sintático pode ser subdividido em duas partes. A primeira é o modelo, que é a forma como se quebra em eventos menores a geração da árvore para se definir as *Notas(A,S)* a todo par (A,S) e o cálculo das estimativas para o parâmetro $\hat{\Theta}$ (que no caso são os valores de cada uma das probabilidades existentes nos eventos menores que constroem uma árvore segundo o modelo). A segunda consiste em um algoritmo que implementa a procura da $Árvore_{melhor}$ para qualquer sentença de entrada S. No caso do analisador desenvolvido nesta dissertação, este algoritmo é fortemente baseado no método amplamente difundido na literatura de processamento de língua natural, chamado Carta de Análise [Charniak 93]. O algoritmo monta a árvore de baixo para cima, calculando a nota das subestruturas montadas de acordo com as dependências léxicas existentes. Quando se junta 2 subestruturas formando uma maior, a sua nota é calculada aproveitando-se as notas das subestruturas participantes e a nota da nova dependência gerada. Alguns critérios de parada, busca, ordem de união e eliminação de constituintes de baixa nota, são usados para melhorar a eficiência. O algoritmo de Carta de Análise e suas adaptações ao analisador deste trabalho serão descritos na seção 2.10 .

Essas duas partes estão tratadas em separado nas seções a seguir.

2.3 Lexicalização da Árvore Sintática

Alguns modelos de análise sintática utilizam em sua representação itens lexicais em cada um dos constituintes presentes nos nós da árvore. Normalmente os córpus existentes para treinamento e testes não possuem tais itens lexicais nos níveis não-terminais de suas árvores e às vezes nem possuem etiquetas em tais níveis. O córpus utilizado para o Português nesta dissertação está apresentado no Capítulo 4, mas de forma breve é importante saber que ele já possui todas etiquetas em seus nós e que os não-terminais não são lexicalizados. Assim, tal processo não só é importante para entender alguns modelos subseqüentes como também foi implementado nos trabalhos descritos no Capítulo 5.

A lexicalização de uma árvore sintática é feita a partir da definição de um constituinte núcleo para cada uma das regras gramaticais existentes. Assim, uma regra do tipo $P \rightarrow E_1 E_2 ... E_n$, é alterada para indicar qual dos constituintes E_i é o principal. O item lexical deste

constituinte principal será o item lexical do símbolo não-terminal da esquerda (*P*). Por exemplo, na regra: VP → VB-D NP ADV, o sublinhado em VB-D indica que ele é o constituinte principal e portanto, quando esta regra for aplicada no trecho '*comprou comida ontem*', o item lexical *comprou* será passado para o VP. A estrutura de árvore lexicalizada correspondente é a seguinte:

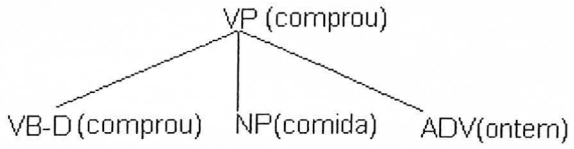


Figura 2.2: Exemplo de lexicalização. O item lexical do núcleo do constituinte principal passa ao pai

Portanto, o processo de lexicalização é feito a partir da base da árvore em direção ao topo. A definição das regras de escolha do núcleo para cada regra gramatical existente no córpus em Português e outros comentários relativos a este processo neste córpus são apresentados no Capítulo 4.

2.4 Aspectos Relevantes da Modelagem

O modo de se quebrar a análise e síntese da árvore sintática é determinante para o sucesso de um analisador sintático. Essa quebra consiste em caracterizar a geração da árvore sintática como uma seqüência de eventos < $Evento_1$ ... $Evento_n$ > a partir da qual a nota da árvore sintática *A* para a sentença *S* será calculada da seguinte forma: $Nota(A,S) = \prod_{i=1...n} Nota(Evento_i)$. Perceba que não necessariamente existe a hipótese de independência entre os *n* Eventos de geração, já que $Nota(Evento_i)$ pode depender de um ou mais ou todos $Evento_j$ tal que $j < i$.

A parametrização escolhida deve levar dois critérios em consideração:

Poder de discernimento Os parâmetros devem conter informações contextuais suficientes para realizar decisões de distinção com o maior grau de confiança possível. Problemas de ambigüidade para geração da árvore sintática são bastante comuns, mas os mais relevantes são de dois tipos:

- Ambigüidade de anexo preposicional

  Por exemplo: *O menino viu a menina com o telescópio.*

  Esta sentença aceita duas árvores sintáticas, a primeira expressa o sentido de que o menino estava usando o telescópio para ver a menina e a segunda expressa o sentido de que a menina estava com o telescópio e foi vista pelo menino. Decidir qual das duas árvores candidatas é a mais provável para a sentença deve ser

possível através de um contexto bem definido. Neste caso, basta por exemplo que o contexto seja capaz de captar a preferência por anexação local, isto é, a preferência pela anexação ao predicado mais próximo, que então já se teria um critério suficientemente pesado para a decisão. Esta preferência, observada no Inglês e provavelmente também no Português, pode ser captada por exemplo com medidas de distância entre as palavras participantes das relações distintas.

- Ambigüidade de coordenação

Por exemplo: *cachorros em casa e gatos*

A coordenação no trecho acima pode ser tanto entre '*cachorros em casa*' e '*gatos*', como entre '*casa*' e '*gatos*'. A decisão do analisador sintático não tem fundamentação semântica, mas sim estatística, e portanto, o contexto escolhido é relevante para a qualidade da decisão. Neste caso, o uso no contexto de informações sobre os itens lexicais envolvidos já ajuda muito a decidir por qual coordenação usar. Parâmetros obtidos sobre treinamento em um córpus suficientemente grande indicariam que é muito mais provável existir uma relação de coordenação entre os itens lexicais '*gatos*' e '*cachorros*' do que entre '*gatos*' e '*casa*'. As informações dos itens lexicais também podem ser usadas para o processo de 'ajustagem de parâmetros', apresentado no item 2.8, que diminuiria a necessidade de córpus maiores de treinamento. Se por outro lado, usássemos apenas informações sobre a categoria morfossintática das palavras envolvidas e ignorássemos qualquer outro dado sobre seus itens lexicais, o poder de discernimento nesta ambigüidade seria nulo, já que as três palavras no caso têm a mesma etiqueta morfossintática (é o que ocorre no modelo GLCP a ser visto no item 2.9.2).

Resolver um problema de ambigüidade fica mais fácil se o modelo consegue 'enxergar' construções impossíveis ou improváveis dentro de uma subestrutura de uma das árvores candidatas. O quanto o modelo consegue 'enxergar' é determinado pelas informações contextuais utilizadas para a decisão. Um ponto que Michael Collins sustenta para o relativo sucesso de seu modelo é exatamente na escolha destes contextos. Ele afirma que devido ao efeito de 'localidade' existente nas palavras das sentenças em Inglês, a sua escolha de decomposição da árvore sintática centralizada nos itens lexicais núcleos permite a seleção de contextos 'enxutos' e com grande poder de discernimento. Tal afirmação será estudada na introdução do Capítulo 3 desta dissertação.

Compacidade Dado um nível de poder de discernimento adequado, um modelo deve atingi-lo usando a quantidade mínima possível de parâmetros, ou seja, o modelo deve ser o mais compacto possível. Quanto mais parâmetros, mais amplo fica o seu domínio de análise e portanto, mais dados de treinamento são necessários para se treinar o modelo. Um modelo com muitos parâmetros facilmente sofre de grande erro de falta de amostragem, definido no item 2.8. Em poucas palavras, significa que os parâmetros obtidos num treinamento insuficiente representam as probabilidades presentes apenas naquele córpus e não podem ser estendidos ao contexto mais geral da língua a qual o córpus pertence.

As duas características acima evoluem em sentidos opostos. Quanto mais parâmetros seu modelo tem, maior seria, teoricamente, o seu poder de discernimento, e menor seria a sua compacidade. No nível mais extremo, a geração de uma árvore é composta por um único passo, o da própria geração de toda a árvore. Este caso peca pela compacidade, já que o número de parâmetros a ser estimado seria do tamanho do domínio de sentenças possíveis da língua, ou seja, extremamente grande, senão infinito. A quantidade de dados de treinamento seria igualmente extensa e inviável. Deste modo, um bom modelo deve ser obtido pelo equilíbrio das duas características, visando sempre eliminar parâmetros redundantes (que diferenciam as mesmas ambigüidades).

2.5 Estimativa de Maximização da Verossimilhança

O modelo utilizado neste trabalho baseia-se numa função probabilística cuja estrutura é fixa, mas que pode ser variada de acordo com um vetor de parâmetros Θ que define cada uma das probabilidades existentes nos eventos que constituem a geração de uma árvore. Assim, um aspecto importante para a implementação do modelo é justamente o método de estimativa deste vetor Θ a partir dos dados de treinamento. O método utilizado no analisador deste trabalho é uma variação da estimativa de maximização da função de verossimilhança. Esta maximização pode ser entendida de maneira resumida abaixo, mas para uma compreensão mais completa e inclusive com descrição de suas propriedades deve-se ler [BD 77].

Considere o caso simples de um espaço de eventos discreto ζ e P uma função de distribuição de probabilidade sobre este espaço tal que:

(1) $0 \leq P(A) \leq 1$ para todo $A \in \zeta$;

(2) $\sum_{A \in \zeta} P(A) = 1$.

A função P pode se parametrizada com a inserção de um vetor de parâmetros Θ no seu contexto, isto é, temos a probabilidade de um evento A dado um valor de parâmetro Θ como

sendo $P(A|\Theta)$. O parâmetro Θ pertence ao espaço Ω de todos os possíveis Θ tal que $\{\Theta| P(A|\Theta)\}$ é uma medida de probabilidade sobre o espaço ζ conforme definido pelos critérios (1) e (2) descritos acima. A função de verossimilhança é usada como critério para a escolha da estimativa do parâmetro Θ que melhor aproxima a função P da distribuição de probabilidade real para o espaço treinado. Assumindo-se uma seqüência de n eventos amostrados $\Lambda = <A_1, A_2, \ldots A_n>$ e que tais eventos são independentes uns dos outros, temos que a função de maximização da verossimilhança L é dada por:

$$L(\Lambda \mid \Theta) = \prod_{i=1\ldots n} P(A_i \mid \Theta) \qquad (2.2)$$

A estimativa de maximização de verossimilhança $\hat{\Theta}_{MV}$ é o valor de Θ no espaço de parâmetros Ω que maximiza a função de verossimilhança , isto é:

$$\hat{\Theta}_{MV} = \arg\max_{\Theta \in \Omega} L(\Lambda \mid \Theta) \qquad (2.3)$$

2.6 O Conceito de Modelo Bem-Definido

Na prática então, um modelo de análise sintática define uma escolha de como se quebrar os membros do espaço de entrada e saída Sentença X Árvores em sub-eventos menores, como por exemplo a geração de cada subestrutura da árvore independentemente. Após isso, o modelo deve especificar seus parâmetros e esta escolha é fundamental para o sucesso prático da implementação do modelo. Para tratar deste aspecto, define-se então como modelo bem-definido o modelo que possui as seguintes duas características:

(1) A estrutura do modelo $Nota(x, y|\Theta)$ para todos os valores de $\Theta \in \Omega$ devem ser ou uma probabilidade condicional $P(y|x)$ (isto é: $\forall x \in \chi \ \sum_{y \in \gamma} Nota(x,y) = 1$) ou uma probabilidade conjunta $P(x,y)$ (isto é: $\sum_{x \in \chi, y \in \gamma} Nota(x,y) = 1$).

(2) A estimativa de maximização da verossimilhança para o modelo deve ser derivada de forma fechada ou por uma solução iterativa como por exemplo usando método de estimativa EM ou Maximização de Entropia.

Estes dois critérios são importantes portanto no momento de se escolher entre possíveis modelos para um mesmo problema. A maioria dos modelos propostos em análise sintática respeita tais condições a fim de se obter resultados de precisão melhores. No Capítulo 3 desta dissertação é feita uma reflexão sobre falhas no primeiro modelo orientado ao núcleo-léxico de Collins quanto a atender tais condições.

De qualquer forma, outro resultado interessante é a existência de uma classe de modelos na qual as duas condições são atendidas de forma simples. Nesta classe, a estrutura do modelo define um distribuição de probabilidade condicional $P(a|b)$ onde b é entrada e a é saída; e a estimativa de maximização da verossimilhança é igual a $\frac{Contagem(a,b)}{Contagem(b)}$, onde *Contagem(a,b)* é a quantidade de eventos no domínio de treinamento em que a entrada foi b e a saída foi a, e *Contagem(b)* é a quantidade de eventos no domínio de treinamento em que a entrada foi b (independentemente de qual foi a saída). Esta classe é composta pelos modelos que atendem as seguintes condições:

(1) O vetor de parâmetros do modelo $\Theta = \{ p_1 , p_2 , \ldots, p_n \}$ é a combinação de m distribuições $\Omega_1, \Omega_2, \ldots, \Omega_m$. Cada Ω_i é um subconjunto de inteiros $\{1, 2, \ldots, n\}$ tal que Ω_s formem uma partição de $\{1, 2, \ldots, n\}$. $\{ p_i | \, i \in \Omega_j \}$ são os parâmetros da j-ésimo subconjunto, tal que:

$$\sum_{i \in \Omega_j} p_i = 1 \qquad (2.3)$$

Consideremos que Ω^i é o subconjunto que contém p_i, isto é $p_i \in \Omega^i$.

(2) Se a verossimilhança do espaço de eventos pode ser escrita como:

$$L(\Lambda \mid \Theta) = \prod_{i \in \Omega_1} p_i^{C(i,\Lambda)} \prod_{i \in \Omega_2} p_i^{C(i,\Lambda)} \ldots \prod_{i \in \Omega_m} p_i^{C(i,\Lambda)} \qquad (2.4)$$

onde $C(i,\Lambda)$ é a contagem do evento que corresponde a p_i na amostragem Λ.

Se estas condições são satisfeitas, maximizar 2.4 sob as restrições 2.3 significa que a estimativa de maximização de verossimilhança para cada p_i é dada por:

$$\hat{p}_{iMV} = \frac{C(i,\Lambda)}{\sum_{j \in \Omega^i} C(j,\Lambda)} \qquad (2.5)$$

Este resultado apresentado em [Collins 99] é bastante útil para o trabalho desta dissertação pois muitas das tarefas presentes no analisador sintático implementado utilizam modelos que atendem tais condições e cujos parâmetros, portanto, podem ser obtidos de forma direta a partir do córpus de treinamento. Sua prova pode ser vista em [Collins 99], página 40.

2.7 Fontes de Erros nas Estimativas

Vamos considerar de mais de perto a questão de se estimar os parâmetros dos subconjuntos Ω_I quando o conjunto de dados é esparso através da "suavização" das estimativas de maximização de verossimilhança (MV). Esta seção apresenta o problema de forma resumida e a seguinte apresenta o método de ajustagem aplicado. Para uma revisão mais completa veja [Jelinek 90].

Todos os modelos apresentados na seção 2.9 na prática envolvem estimativa de parâmetros de distribuições de probabilidades presentes em si. São parâmetros da forma $P(Y|X_1, X_2, \ldots, X_n)$ que requerem estimativas para cada valor de Y dentro do seu domínio γ de possibilidades e para cada combinação de valores de suas n variáveis de contexto $X_1, X_2, \ldots, X_n$ que são membros dos conjuntos $\chi_1, \chi_2, \ldots, \chi_n$, respectivamente. Por simplicidade usamos X para denotar $X_1, X_2, \ldots, X_n$; membro do conjunto $\mathbf{X} = X_1 \times X_2 \times \ldots \times X_n$. Assim, temos:

$$\hat{P}_{MV}(Y \mid X) = \frac{Contagem(Y, X)}{Contagem(X)} \qquad (2.6)$$

Em espaços altamente-dimensionais *Contagem*(X) pode ser bastante baixa ou até mesmo 0, deixando 2.6 imprecisa ou indefinida. Este é o problema de dados esparsos, isto é, o conjunto de dados de treinamento não é suficientemente grande para nos definir estimativas seguras para todos os parâmetros $P(Y|X)$. Uma estratégia geral para se contornar tal problema é o uso de estimativas MV calculadas a partir de distribuições de *ordem menor*. Parâmetros de ordem menor são distribuições de probabilidade que estão condicionados em sub-contextos de $\mathbf{X}$, e que portanto devem ter contagens de ocorrência de seus eventos mais altas e confiáveis. Por exemplo, se n =3, temos então 7 distribuições de *ordens menores*: $P(Y|X_1)$, $P(Y|X_x)$, $P(Y|X_1, X_2)$, $P(Y)$ e assim por diante. A utilidade de estimativas de *ordens menores* pode ser motivada pela compreensão das duas fontes de erro de estimativa explicadas a seguir. Para esta análise iremos considerar o caso de uma distribuição binomial, no caso particular em que $|Y| = 2$, mas cujas conclusões podem ser estendidas para os casos em que $|Y|>2$.

Digamos que $\hat{p}^n$ é uma estimativa de p baseada numa amostragem de tamanho n. Defini-se o erro de estimativa $\hat{p}^n$ como:

$$Err(\hat{p}^n) = E_p[(\hat{p}^n - p)^2] \qquad (2.7)$$

Onde E_p refere-se ao valor da esperança com respeito a probabilidade p. Para esta seção iremos considerar que o objetivo do método de estimativa é minimizar a definição de *Err* acima. Outras funções de erro podem e às vezes são usadas, mas isso não modificará as conclusões que chegaremos ao fim.

Na pg. 117 de [BD 77], prova-se que $Err(\hat{p}^n)$ é dado pela soma de suas componentes:

$$Err(\hat{p}^n) = Err_1(\hat{p}^n) + Err_2(\hat{p}^n) \tag{2.8}$$

$$Err_1(\hat{p}^n) = (\breve{p}^n - p)^2 \tag{2.9}$$

$$Err_2(\hat{p}^n) = E_p[(\hat{p}^n - \breve{p}^n)^2] \tag{2.10}$$

Onde $\breve{p}^n$ é definido como $E_p[\hat{p}^n]$. Assim Err_1 é o quadrado do erro base de estimativa $\breve{p}^n - p$ e Err_2 é a medida do erro de amostragem. Para tamanhos de amostragem pequenos, $\hat{p}^n$ com mais probabilidade irá variar de seu valor médio $\breve{p}^n$. À medida que *n* cresce, Err_2 decresce e tende para 0 quando $n \to \infty$.

Um resultado de [BD 77] é que a estimativa MV $\hat{P}_{MV}(Y \mid X)$ tem Err_1 igual a 0. Porém, como o tamanho da amostra geralmente é pequeno, Err_2 costuma ser muito grande e portanto prejudica muito a acurácia dos métodos. Por outro lado se usássemos $\hat{P}(Y \mid X) = \hat{P}_{MV}(Y)$, uma estimativa MV de *Y* baseada no sub-contexto vazio de **X**, teríamos com quase toda certeza $Err_1 > 0$, mas sua componente Err_2 seria bem menor que o valor de Err_2 para $\hat{P}_{MV}(Y \mid X)$.

Entre estes dois casos extremos existem diversos sub-contextos $\Phi(\mathbf{X})$ não vazios de **X**. À medida que o sub-contexto diminui, Err_1 cresce e Err_2 decresce. Assim, pode-se usar uma estratégia de estimativa tal que $\hat{P}(Y \mid X) = \hat{P}_{MV}(Y \mid \Phi(X))$ na qual escolhemos $\Phi(X)$ que leva ao melhor compromisso entre Err_1 e Err_2 e portanto minimizando a esperança de erro. Porém, melhor que esta estratégia, existe o método de ajustagem de parâmetros que é descrito na seção seguinte.

2.8 Método de Ajustagem de Parâmetros

Uma técnica melhor para redução do erro de estimativa consiste em se fazer a média ponderada de estimativas MV sob diferentes sub-contextos. Este método se chama ajustagem de parâmetros. Vamos usar como exemplo uma estratégia com *n* níveis de sub-contextos $\Phi = \Phi_1\Phi_2 \ldots \Phi_n$. Consideremos para esses contextos que:

1. $\hat{P}_i(Y \mid X)$ é definido como $\hat{P}_{MV}(Y \mid \Phi_i(X)) = \dfrac{Contagem(Y, \Phi_i(X))}{Contagem(\Phi_i(X))}$ (2.11)

2. $\Phi_1(X)$ é suficientemente pequeno para que $\hat{P}_1(Y \mid X)$ esteja definido para todos os casos, isto é, $\forall\ X \in \mathbf{X}$, $Contagem(\Phi_1(X)) > 0$ (geralmente isso significa que $\Phi_1(X)$ é o conjunto vazio).

3. Se i < j então $\Phi_i(X) \subset \Phi_j(X)$.

Assim, à medida que i aumenta Err_1 decresce e Err_2 diminui, criando um compromisso entre as duas medidas na seqüência $\hat{P}_1 \ldots \hat{P}_n$. Uma combinação linear destas estimativas usando-se pesos λ_i associados a cada nível pode definir uma estimativa suavizada da seguinte forma:

$$\tilde{P}_1 = \hat{P}_1$$

$$\tilde{P}_i = \lambda_i \hat{P}_i + (1-\lambda_i) \tilde{P}_{i-1} \text{ para cada } 1 < i \leq n \text{ e } 0 \leq \lambda_i \leq 1$$

O coeficiente λ_i pode ser visto como medida do quanto confiamos na estimativa $\hat{P}_i$ a mais do que na suavizada do nível inferior (i – 1). $\lambda_i = 0$ significa que a estimativa não tem confiança nenhuma enquanto $\lambda_i = 1$ significa que a estimativa é bastante confiável (isto é, as contagens de eventos são significativamente altas) e portanto é preferível utilizar apenas esta medida e ignorar a suavizada de nível inferior por conter erro de base maior (Err_1).

O problema passa então a ser como estimar os valores λ_i que levam a uma ajustagem de parâmetros com media de erro minimizada. Para isso, [Jelinek 90] descreve o método de otimização para verossimilhança de dados separados[6]. Outra forma é a utilização de fórmulas heurísticas para cálculo direto de λ_i. Abaixo é apresentada uma destas fórmulas (uma das mais simples):

- $\lambda_i = 0$ se $Contagem(\Phi_i(X)) = 0$
- ou $\lambda_i = \dfrac{Contagem(\Phi_i(X))}{Contagem(\Phi_i(X)) + C_i}$ se $Contagem(\Phi_i(X)) > 0$

O valor de C_i é otimizado com uso de dados separados e indica o quão lento é a convergência de λ_i para 1 à medida que $Contagem(\Phi_i(X))$ vai para ∞.

[Bikel et al 97] sugere uma outra fórmula de cálculo que leva em conta a diversidade de $\Phi_i(X)$ denotada por $D(\Phi_i(X))$.[7] [CG 96] sugere uma fórmula que normalmente tem um bom desempenho e usa uma medida chamada *ContagemUnica*[8].

---

[6] Em inglês *held-out data.*

[7] $D(\Phi_i(X)) = |\gamma(\Phi_i(X))|$ onde $\gamma(\Phi_i(X)) = \{y \mid Contagem(y, \Phi_i(X)) > 0\}$.

[8] $ContagemUnica(\Phi_i(X)) = |\gamma^*(\Phi_i(X))|$ onde $\gamma^*(\Phi_i(X)) = \{y \mid Contagem(y, \Phi_i(X)) = 1\}$.

Exemplo:

O Capítulo 5 de [Collins 99] relata a experiência de se aplicar o método de ajustagem de parâmetros para o problema de decisão de anexação nominal ou verbal de uma preposição nominal no Inglês. O problema pode ser descrito de forma resumida como o de se decidir se um anexo preposicional com núcleo nominal $n2$ e preposição p deve estar anexado ao verbo v ou ao nome $n1$ num exemplo de sentença onde este tipo de ambigüidade comum ocorre. Assim, dada uma tupla $<p, n_1, v, n_2>$, o sistema deve ser capaz de estimar $\hat{p}(A \mid p, n_1, v, n_2)$, onde A = 0 indica anexação verbal e A = 1 indica anexação nominal e por fim decidir qual anexação usar (neste caso, aquela cuja $\hat{p}(A \mid p, n_1, v, n_2) > 0{,}5$ ou caso $\hat{p}(0 \mid p, n_1, v, n_2) = \hat{p}(1 \mid p, n_1, v, n_2) = 0{,}5 \Rightarrow A = 1$).

O capítulo chega à conclusão de que neste problema em particular os valores das estimativas com sub-contextos maiores são preferíveis a de contextos menores mesmo se as contagens daquelas forem baixas, como por exemplo *Contagem* ($\Phi_i(X)$) = 1. Além disso, ele consegue ordenar os termos da tupla por relevância como critério de decisão: a preposição é o termo mais relevante para se tomar a decisão, depois igualmente relevantes estão $n1$ e v, e por fim $n2$. O trabalho apresenta dados relevantes para análise de performance e escolha dos sub-contextos e por fim ele apresenta o seguinte algoritmo de ajustagem de parâmetros:

1. Se *Contagem*$(p, n_1, v, n_2) > 0$

$$\hat{p}(1 \mid p, n_1, v, n_2) = \frac{\text{Contagem}(1, \text{p}, \text{n}_1, \text{v}, \text{n}_2)}{\text{Contagem}(\text{p}, \text{n}_1, \text{v}, \text{n}_2)}$$

2. Senão se *Contagem*(p, n_1, v) + *Contagem*(p, v, n_2) + *Contagem*$(p, n_1, n_2) > 0$

$$\hat{p}(1 \mid p, n_1, v, n_2) = \frac{\text{Contagem}(1, \text{p}, \text{n}_1, \text{v}) + \text{Contagem}(1, \text{p}, \text{v}, \text{n}_2) + \text{Contagem}(1, \text{p}, \text{n}_1, \text{n}_2)}{\text{Contagem}(\text{p}, \text{n}_1, \text{v}) + \text{Contagem}(\text{p}, \text{v}, \text{n}_2) + \text{Contagem}(\text{p}, \text{n}_1, \text{n}_2)}$$

3. Senão se *Contagem*(p, v) + *Contagem*(p, n_1) + *Contagem*$(p, n_2) > 0$

$$\hat{p}(1 \mid p, n_1, v, n_2) = \frac{\text{Contagem}(1, \text{p}, \text{v}) + \text{Contagem}(1, \text{p}, \text{n}_1) + \text{Contagem}(1, \text{p}, \text{n}_2)}{\text{Contagem}(\text{p}, \text{v}) + \text{Contagem}(\text{p}, \text{n}_1) + \text{Contagem}(\text{p}, \text{n}_2)}$$

4. Senão se *Contagem*$(p) > 0$

$$\hat{p}(1 \mid p, n_1, v, n_2) = \frac{\text{Contagem}(1, \text{p})}{\text{Contagem}(\text{p})}$$

5. Senão $\hat{p}(1 \mid p, n_1, v, n_2) = 1{,}0$ (por padrão decide-se por anexação nominal).

Os coeficientes λ_i estão presentes em cada nível da suavização do algoritmo. Só como ilustração, perceba que no nível 3 os coeficientes são os seguintes (dado que d = *Contagem*(p, v) + *Contagem*(p, n_1) + *Contagem*(p, n_2)):

$$\lambda_1 = \frac{\text{Contagem(p, v)}}{\text{d}} \qquad \lambda_2 = \frac{\text{Contagem(p, n}_1\text{)}}{\text{d}} \qquad \lambda_3 = \frac{\text{Contagem(p, n}_2\text{)}}{\text{d}}$$

2.9 Métodos de Análise Sintática

Nesta seção são apresentados alguns métodos utilizados em análise sintática para o Inglês, a maioria probabilísticos, em ordem crescente de complexidade, partindo de *gramáticas livres de contexto*[9] (GLCs) até o método orientado ao núcleo-léxico proposto por [Collins 99]. O foco principal é apresentar as vantagens, desvantagens e resultados existentes para cada método.

2.9.1 Gramáticas Livres de Contexto (GLCs)

Uma GLC pode ser definida por uma quádrupla (T, N, P, S), onde:

- T é o alfabeto que define as cadeias da linguagem, ou seja, é o conjunto de símbolos terminais.
- N é o conjunto de não-terminais onde $T \cap N = \varnothing$.
- P é o conjunto de produções da forma $a \rightarrow b$, onde $a \in N$ e $b \in (T \cup N)^*$.
- $S \in N$ é o símbolo inicial.

A linguagem definida por uma gramática é o conjunto de cadeias de terminais que podem ser derivadas a partir de uma seqüência de operações de reescrita com o símbolo inicial S. As operações de reescrita são definidas pelas produções da forma $a \rightarrow b$, onde a é substituído por b. Exemplos podem ser vistos em [Cáccamo 98].

GLCs são usadas com relativo sucesso em análise na teoria de linguagens formais, onde o tamanho da gramática, expresso pelo número de regras de produção e não-terminais, é relativamente pequeno. No caso de línguas naturais, o tamanho da gramática é bem maior, o que

[9] Em inglês *Context-Free Grammars (CFGs)*.

degrada sensivelmente a eficiência dos algoritmos tradicionais. A literatura não apresenta resultados muitos seguros para a avaliação do sucesso relativo dos resultados da aplicação de GLC para análise sintática. Estes métodos foram implementados quando ainda não se disponibilizava de bancos de árvores para avaliação de acurácia e seus domínios de testes eram bastante restritos.

O maior problema para o uso de GLCs em análise sintática de línguas naturais está no fato do modelo ser muito pouco sensível ao contexto superficial da sentença. Isso gera problemas, por exemplo, na representação de certas relações superficiais de concordância, que são indispensáveis para avaliar a viabilidade da estrutura gerada. Outro aspecto negativo está no fato de que o treinamento automático de uma GLC deve ser feito com a utilização de exemplos negativos, pois caso contrário, a GLC gerada será permissiva, isto é, aceitará muitas sentenças que não são gramaticalmente corretas.

2.9.2 Gramáticas Livres de Contexto Probabilísticas[10] (GLCPs)

Pelo modelo de GLCP [Charniak 93], uma árvore sintática é representada como uma seqüência de *n* eventos, onde o i-ésimo evento significa a i-ésima regra livre de contexto aplicada para se obter a árvore. A probabilidade da árvore é calculada como o produto das probabilidades de seus *n* eventos, que no caso, são as probabilidades de cada produção gramatical aplicada. A probabilidade de uma sentença qualquer pelo modelo do GLCP pode ser calculada como o somatório das probabilidades de todas as árvores possíveis para a entrada em questão.

Alguns resultados reportados com o uso de GLCPs para análise sintática são relativamente medíocres. [Charniak 97] treinou e testou uma GLCP usando o banco de árvores do *Penn Wall Street Journal*, obtendo uma acurácia de 70.6%/74.8% cobertura/precisão. As definições destas medidas de acurácia estão apresentadas na seção 2.11, mas no momento basta saber que quanto maior estes índices, melhor é o desempenho do algoritmo.

Apesar deste resultado ruim, GLCPs têm vantagens úteis em relação a métodos mais simples, como as GLCs ou modelos de trigramas. A primeira vantagem está em ter-se um método direto para se distinguir entre árvores candidatas. Escolhe-se sempre a árvore mais provável. A segunda vantagem é a falta de necessidade de exemplos negativos para o treinamento. Isso é contornado pelo fato de que gramáticas super-geradoras assimilam probabilidades menores às sentenças gramaticalmente corretas do córpus de treinamento, pois há perda de probabilidades não nulas em sentenças não gramaticais. Portanto, a gramática que

[10] Em inglês *Probabilistic Context-Free Grammars (PCFGs).*

maximiza a verossimilhança em relação ao córpus de treinamento será a melhor, não sendo necessário o uso de exemplos negativos.

As principais desvantagens das GLCPs são a falta de sensibilidade a dependências léxicas e a falta de sensibilidade a preferências estruturais, como por exemplo, preferência por ramificações à direita ou à esquerda.

A primeira deficiência significa que a decisão entre duas estruturas depende apenas das probabilidades da regras utilizadas, e não tem nenhuma relação com os itens lexicais em questão. Por exemplo, a frase '*trabalhadores venderam sacos de um químico*' pode ter duas estruturas sintáticas válidas, uma na qual '*de um químico*' se anexa ao verbo através da regra $VP \rightarrow VP\ PP$ e a outra em que se anexa ao nome através da regra $NP \rightarrow NP\ PP$. A decisão de qual estrutura é a mais provável é feita apenas pela probabilidade $P(VP \rightarrow VP\ PP)$ e $P(NP \rightarrow NP\ PP)$, pois todas as outras regras são iguais. Ou seja, se $P(VP \rightarrow VP\ PP) > P(NP \rightarrow NP\ PP)$, a árvore utilizada será a de anexação verbal, senão será escolhida a de anexação nominal. Suponhamos que $P(VP \rightarrow VP\ PP) < P(NP \rightarrow NP\ PP)$. É fácil ver que qualquer frase com a seqüência de categorias morfossintáticas *<NNS VBD NNS IN DT NN>* será assinalada com a árvore de anexação nominal. Isso acarreta em erro, por exemplo, na frase '*trabalhadores empilharam sacos dentro da lata*', que tem a mesma seqüência de categorias morfossintáticas, mas é de anexação verbal.

Erros similares ocorrem em casos de coordenação, anexação de pronomes relativos e composição de nomes. Outro exemplo é a coordenação '*cachorros em casa e gatos*' apresentada na seção 2.2. As duas estruturas de coordenação possuem as mesmas regras gramaticais, aplicadas apenas em ordem diferente, e portanto, teriam a mesma probabilidade. Neste caso, o método de GLCP não teria como decidir.

A deficiência de falta de sensibilidade de preferências estruturais pode ser verificada em anexação adverbial, na qual o advérbio geralmente altera o local de anexação (verbo, locução adverbial) mais recente. Esta preferência é do tipo anexação à direita. As duas possíveis formas de anexação têm as mesmas regras gramaticais e portanto a GLCP não tem como distinguir.

2.9.3 Método Proposto por Charniak

O método proposto por [Charniak 97] limita a geração de árvores a apenas utilizar as regras gramaticais vistas no treinamento. As regras são lexicalizadas, e sua aplicação é dividida em duas partes. A primeira atribui probabilidade para a utilização da regra em si, e a segunda avalia a probabilidade dos itens lexicais que preenchem a folhas. Assim, para uma produção do tipo $E(h) \rightarrow L_n(l_n)...L_1(l_1)HR_1(r_1)...R_m(r_m)$, onde H é o símbolo núcleo do lado direito e h, $l_1...l_n$,

$r_1 ... r_m$ são os itens lexicais associados aos símbolos da produção, a probabilidade é estimada como:

$$\mathcal{P}(L_n(l_n)...L_1(l_1)HR_1(r_1)...R_m(r_m) \mid E(h)) = \mathcal{P}(L_n...L_1HR_1...R_m \mid E(h)) \times \prod_{i=1...n} P_l(l_i \mid L_i, P, h) \times$$

$$\times \prod_{i=1...n} P_r(r_i \mid R_i, P, h) \qquad (2.12)$$

Um processo de interpolação é utilizado, cujo nível mais baixo é dado por apenas $\mathcal{P}(L_n...L_1HR_1...R_m \mid E)$. Deste modo, regras gramaticais não vistas nos dados de treinamento terão probabilidade zero. Isso é ruim, pois quando o estilo de anotação utilizado é plano (minimiza ao máximo o número de níveis da árvore) temos muita variedade de regras gramaticais. Algumas ocorrerão apenas nos dados de teste e nunca nos de treinamento, e portanto árvores sintáticas corretas serão erroneamente descartadas. A acurácia deste modelo foi avaliada sobre o mesmo domínio de teste dos trabalhos *SPATTER* (seção 2.9.4.1) e de dependências léxicas (seção 2.9.4.2). Os resultados obtidos foram superiores aos dos trabalhos anteriores, atingindo 86,7% / 86,6% de cobertura/precisão. Outra vantagem do modelo é que ele define uma distribuição de probabilidade conjunta $P(A,S)$ enquanto os outros dois definem uma condicional $P(A|S)$. Estatisticamente isto é relevante dependendo do tipo de problema ao qual o modelo é aplicado, como por exemplo em sistemas de reconhecimento de fala e de reconhecimento de caracteres.

O trabalho de Charniak também enfoca o aprendizado automático de gramáticas livres de contexto a partir de uma base não anotada manualmente, com relevância ao aspecto matemático do modelo. Trata também de técnicas de decisão entre anexação preposicional verbal ou nominal, de referência de pronomes relativos e sugere a utilização de informações semânticas, além das estruturais, para auxiliar na distinção. Por fim, como base para uma nova linha de modelos, Charniak apresenta técnicas de agrupamento por propriedades[11] das palavras que podem ser utilizados para interpolação de probabilidades num domínio menos esparso que o dos itens lexicais, além de apresentar técnicas para decidir o sentido de uma palavra de múltiplos sentidos, através do contexto ou de informações externas. Ambas as técnicas são muito úteis para sistemas de tradução automática, e estão intimamente relacionadas com a análise sintática das sentenças.

[11] Em inglês *clustering*.

2.9.4 Métodos Baseados em História[12]

Esse tipo de método foi proposto pela primeira vez em 1992 por [Black et al 92]. Consiste em se fazer um mapeamento entre o espaço de entrada e saída X x Y (no caso, sentença e árvore) para uma seqüência de decisões $<d_1, d_2 ... d_n>$. O modelo pode ter tanto uma distribuição conjunta como condicional. Seja $x \in X$ (uma sentença) e $y \in Y$ (uma árvore), temos que:

distribuição conjunta: $$\mathcal{P}(x, y) = \mathcal{P}(<d_1, d_2 ... d_n>) = \prod_{i=1...n} P(d_i \mid d_1 ... d_{i-1}) \quad (2.13)$$

distribuição condicional: $$\mathcal{P}(y \mid x) = \mathcal{P}(<d_1, d_2 ... d_n> \mid x) = \prod_{i=1...n} P(d_i \mid d_1 ... d_{i-1}, x) \quad (2.14)$$

Condicionar a decisão d_i a todas as decisões prévias de d_1 até d_{i-1}, levaria a um problema de estimativa de muitos parâmetros, o que exigiria muitos dados de treinamento. Assim, na prática se agrupam histórias em classes equivalentes através de uma função Φ, obtendo-se probabilidades condicionadas a $\Phi(d_1 ... d_{i-1})$ ao invés de $d_1 ... d_{i-1}$. O projeto de um modelo baseado em história pode ser dividido nos 3 passos seguintes:

1. **Representação.** Conforme apresentado na seção 2.1, escolhe-se como representar a árvore sintática. Por exemplo, quais serão as etiquetas morfossintáticas e os não-terminais presentes na árvore; quando ou não se terão núcleos-léxicos anexados aos nós; como representar as palavras: diretamente ou através de estruturas como radicais mais flexões ou até mesmo através de técnicas de agrupamento por propriedades.

2. **Decomposição.** Neste passo se define o mapeamento um-para-um entre árvores A e seqüência de decisões $<d_1, d_2 ... d_n>$ em uma ordem canônica. Dado o mapeamento, o modelo assume então uma distribuição de probabilidade conjunta ou condicional de acordo com as fórmulas 2.12 e 2.13 respectivamente. No caso de probabilidade conjunta as decisões geralmente mapeiam a derivação da árvore de cima para baixo (gerativo); e no caso de probabilidade condicional a seqüência de decisões é associada a um processo de recuperação da árvore (geralmente de baixo para cima).

3. **Hipóteses de independência.** Neste passo define-se a função Φ de agrupamento que reduz os contextos de cada decisão em classes equivalentes, mas de proporções aceitáveis para o treinamento e estimativa de parâmetros. A função Φ pode ser definida tanto a mão como automaticamente através de técnicas de aprendizado computacional (veja SPATTER na seqüência). O modelo final torna-se por este passo:

distribuição conjunta: $$\mathcal{P}(S, A) = \prod_{i=1...n} P(d_i \mid \Phi(d_1 ... d_{i-1})) \quad (2.14)$$

[12] Em inglês *History-Based Methods.*

distribuição condicional: $$\mathcal{P}(A|S) = \prod_{i=1...n} P(d_i \mid \Phi(d_1 ... d_{i-1}, S)) \quad (2.15)$$

Esse tipo de modelo foi usado em muitos dos recentes analisadores sintático. Dois deles são mencionados a seguir.

2.9.4.1 Analisador Sintático SPATTER

[Magerman 95] desenvolveu um analisador sintático chamado de *SPATTER* que obteve resultados significativamente superiores aos de GLCP aplicado ao mesmo domínio. Este método baseia-se fortemente em informações léxicas, atingindo resultados de 84,0%/84,3% cobertura/precisão sobre o banco de árvores do *Penn Wall Street Journal*. GLCP atingia valores em torno de 72%.

O trabalho desenvolvido apresentou grandes avanços na área. O primeiro avanço foi de escala, pois todas as sentenças com até 40 palavras foram analisadas num domínio bem mais amplo do que os anteriores. Além disso, o analisador foi completamente obtido automaticamente pelo treinamento a partir do banco de árvores, sem necessidade de uma gramática escrita à mão. A escolha dos sub-contextos de decisão (função Φ apresentada em 2.9.4) foi implementada através de árvores de decisão para procura de hipóteses de independência.

De acordo com o seu próprio prefácio e a luz das três fases presentes no projeto de um método baseado em História (seção 2.9.4), Magerman adotou basicamente o seguinte raciocínio em seu analisador: um especialista de lingüística seria usado para definir a representação (fase 1) que contivesse todos os elementos de árvore-sentença que podem ser úteis para a resolução de ambigüidades; árvores de decisão seriam usadas então no passo 3 para se identificar quais características são realmente significativas como critério de decisão. O problema desta estratégia está na pouca importância que se dá a fase 2 (de decomposição) do modelo. Os parâmetros escolhidos para o *SPATTER* em alguns momentos perdem informações de distinção importantes e em outros momentos fragmenta desnecessariamente os dados de treinamento nas ramificações de suas árvores de decisão. Collins insistirá na importância do passo 2, com especial cuidado na escolha dos parâmetros que quebram e classificam a árvore, como argumentação para o relativo sucesso de seu analisador sintático (leia na introdução do Capítulo 3).

2.9.4.2 Método Orientado ao Núcleo-Léxico

O trabalho desenvolvido em [Collins 99] partiu de um modelo puramente baseado em dependências léxicas, com o qual já se obteve uma acurácia em torno de 85,0,% um pouco superior ao do método *SPATTER*. Esse primeiro modelo, desenvolvido em 1995, tinha a vantagem de ser muito mais simples do que o anterior. A partir dele, novos elementos lingüísticos foram inseridos no modelo na tentativa de torná-lo mais exato, mas paralelamente, tornando-o mais complexo. A seqüência de propostas de parametrização foi basicamente a seguinte:

- **Dependências**

Uma dependência é uma relação entre duas palavras de uma sentença, uma chamada núcleo e outra modificadora, denotado por *<modificador → núcleo>*. Numa sentença de *n* palavras existem *n* dependências, cada uma tem uma probabilidade associada de ocorrer.

Exemplo: *< Guga → venceu >*

- **Dependências + Direção**

A cada dependência é acrescentada a informação de ordem de posição do modificador para o núcleo, denotado por *<modificador → núcleo, direção>*. Deste modo, existe um mapeamento, um para um, entre árvores sintáticas e representação por dependências.

Exemplo: *< Guga → venceu, ESQUERDA >*

- **Dependências + Direção + Relação**

A informação de relação especifica a relação gramatical existente entre as duas palavras, denotado por *<modificador → núcleo, direção, relação>*. Ela consiste em um conjunto de três símbolos não-terminais retirados da árvore sintática. Por exemplo, a dependência *< Guga → venceu, ESQUERDA, <NPR, S, VP> >* significa um relacionamento onde: *Guga* é o modificador; *venceu* é o núcleo; *Guga* está à esquerda de *venceu*; e a relação é *<NPR, S, VP>* (relacionamento sujeito-verbo).

- **Dependências + Direção + Relação + Subcategorias**

Uma árvore é representada por *m* probabilidades de subcategorias, junto com *n* dependências definidas como anteriormente. Subcategorias assimilam probabilidades para eventos da seguinte forma: *existência de dois complementos NP à direita do verbo DAR*, ou, *qual a probabilidade do verbo DAR ter um único sujeito NP a sua esquerda?*.

Exemplos: *<VP, VB-D, venceu, DIREITA, { NPR }>,*
<VP, VB-D, venceu, ESQUERDA, { }>, <S, VP, venceu, DIREITA, { }>,
<S, VP, venceu, ESQUERDA, { NPR } >

- **Dependências + Direção + Relação + Subcategorias + Distância**

A cada dependência é adicionada a informação de distância entre o modificador e o núcleo. Essa distância é uma medida tanto sobre a trecho entre as duas palavras na sentença superficial (a original) como na estrutura da árvore. Deste modo, o modelo pode aprender, por exemplo, preferência por anexação mais próxima (ramificação à direita). Na próxima seção serão apresentadas as definições de medida de distância mais usadas.

- **Dependências + Direção + Relação + Subcategorias + Distância + CMS**

Todos os itens lexicais são ampliados para se incluir a informação de etiqueta em *categoria morfossintática*[13] (CMS) associada. Isso ajuda a resolver o problema de domínio muito esparso sobre os itens lexicais, pois estatísticas sobre etiquetas de CMS podem ser usadas num *modelo de ajustagem*[14] de estimativas de probabilidades.

Outras pequenas modificações são acrescentadas ao modelo final de Collins para aumentar heuristicamente a acurácia em erros específicos, como coordenação e definição de *NPs* básicos. O melhor resultado obtido apresenta uma melhora de acurácia em torno de 2,4% em relação ao método mais simples de dependências (Dependências + Direção + Relação + Distância + CMS). Essa melhora aparentemente pífia se justifica pelo fato de que a medida de distância já é uma boa aproximação para a informação adicionada pelas subcategorias na maioria das ambigüidades sintáticas. Ou seja, o acréscimo de subcategorias no contexto de decisão só aumenta o poder de discernimento em alguns casos específicos. No Capítulo 3 é apresentado uma comparação entre as diferenças e vantagens dos dois modelos implementados por Collins baseados em dependências léxicas.

2.9.5 *Parsing* probabilístico para o Português do Brasil [Bonfante 03]

Alguns meses antes da finalização deste trabalho, [Bonfante 03] reportou resultados de sua pesquisa para investigar o comportamento de um método estatístico para análise sintática baseado no primeiro modelo gerativo de [Collins 99] aplicado às sentenças da língua portuguesa do Brasil. Este trabalho em muitas formas apresenta semelhanças com o modelo de dependências léxicas orientado ao núcleo-léxico desenvolvido no Capítulo 5 desta dissertação. O produto final desta tese é um ambiente chamado PAPO formado por vários módulos que executam três funções básicas: (1) o pré-processamento e preparação dos dados do conjunto de sentenças usado no treinamento, (2) a geração de dois modelos probabilísticos de análise (PAPO-I e PAPO-II), e (3) um analisador sintático final que usa um dos modelos gerados.

---

[13] Em inglês *Part-of-Speech tags.* . Também é traduzido para o Português como Parte do Discurso.
[14] Em inglês *Backed-Off Model.*

O córpus de treinamento utilizado neste trabalho foi obtido do banco de árvores CETENFolha, subconjunto de sentenças compiladas de textos jornalísticos do córpus NILC[15], etiquetado automaticamente com um analisador sintático externo, chamado "*Palavras*", que produz as análises em uma notação "flat", não hierárquica, baseada em gramáticas de restrições. Esta diferença do córpus utilizado acrescentou um ruído de erro nas árvores do *treebank* e a falta de balanceamento destas. Mas apesar das diferenças, este trabalho tem conclusões parecidas com as apresentadas no Capítulo 6, entre elas o fato dos resultados obtidos serem limitados mas promissores e que propriedades como modelo de anotação (sintática e morfossintática) e definição dos núcleos dos constituintes da árvore influenciam profundamente na performance do analisador sintático.

O trabalho realiza uma análise de resultados sobre a aplicação dos modelos PAPO-I e PAPO-II em 23 sentenças externas ao banco de árvores de treinamento. As sentenças tinham entre 5 a 9 palavras cada e a análise realizada observou três aspectos: o tempo de resposta, uma análise quantitativa resumida e uma extensa análise qualitativa de falha e sucesso em cada caso. A média de tempo de resposta ficou em 47s por sentença num AMD Atlon XP 1900 1.6 GHz com 512 de Ram. A precisão e revocação médias reportadas foram de 75%/60% para o PAPO-I e 79%/75% para o PAPO-II. Estes resultados são a primeira vista superiores aos reportados no capítulo 6 desta dissertação. Entretanto a comparação entre os trabalhos não é tão simples, já que o córpus utilizado em [Bonfante 03] é outro, o tamanho do seu conjunto de teste é significativamente menor e com menos palavras por sentença do que a deste trabalho. A análise mais expressiva realizada em [Bonfante 03] foi qualitativa sobre as 23 sentenças processadas. Percebeu-se que existiam três tipos principais de falhas presentes no processamento automático: 1) falha por carência de exemplos no banco de árvores de tamanho insuficiente; 2) falha por ruído na entrada do gerador de modelos provavelmente oriundo do pré-processamento do córpus; 3) falha por incapacidade de aprendizado de um tipo de estrutura de subordinação pelos modelos. De um modo geral, exceto em um dos 23 casos estudados, o sistema PAPO-I teve uma acurácia inferior ao PAPO-II, indicando uma superioridade do segundo em relação ao primeiro.

Além do ineditismo deste trabalho para o Português, ele também representa uma grande contribuição para a pesquisa na área, servindo como referência e base para outra linha de tarefas futuras em análise sintática para Português.

[15] Núcleo Interinstitucional de Lingüística Computacional

2.10 O Algoritmo de Carta de Análise Ascendente[16]

Dado uma gramática livre de contexto e uma sentença qualquer, o algoritmo de Carta de Análise é capaz de encontrar todas as árvores sintáticas possíveis para a sentença de entrada. Se a gramática tiver probabilidades associadas a cada uma de suas regras, o algoritmo ainda é capaz de calcular a nota de probabilidade de cada árvore gerada, segundo o modelo GLCP descrito anteriormente. Ele é ascendente porque gera as árvores de baixo para cima, isto é, a partir dos constituintes terminais até o símbolo não-terminal inicial da gramática. O objetivo desta seção é apresentar um resumo da funcionalidade e das estruturas de dados utilizadas no algoritmo. O algoritmo completo e exemplos de sua aplicação podem ser vistos em [Charniak 93], mas um estudo de outras técnicas bem estabelecidas está em [Winograd 83].

Três estruturas de dados são usadas no algoritmo: uma lista de chaves, uma tabela e um conjunto de arcos. A tabela é um conjunto de entradas compostas por nome do símbolo terminal ou do símbolo não-terminal, o índice da primeira palavra da entrada e o tamanho (em número de palavras) da entrada. Assim uma tabela é metade de uma matriz bidimensional, cujas entradas são indexadas pelo índice da palavra inicial e o tamanho.

A estrutura de lista de chaves é uma pilha, da qual só é possível fazer duas ações: adicionar e remover chaves do topo da pilha. A última estrutura, chamada de 'arcos', representa as regras que podem ser aplicadas sobre entradas da tabela para então gerar entradas maiores. Além disso, a cada passo do algoritmo, os arcos indicam quais regras e onde estão sendo aplicadas na entrada. Um arco tem as seguintes informações: (1) a regra a ser aplicada; (2) a posição de começo do primeiro elemento do lado direito da regra; (3) a posição onde o primeiro elemento não completado do lado direito da regra tem que começar; (4) o último elemento do lado direito da regra que já foi encontrado na entrada, indicando o próximo a ser encontrado.

Com essas três estruturas, o algoritmo é capaz de aplicar regras gramaticais sobre a tabela até obter o símbolo não-terminal inicial. Porém, como o objetivo é encontrar as árvores sintáticas, é preciso acrescentar informações adicionais para que o algoritmo seja capaz de recuperar as árvores geradas. Primeiramente, é preciso que cada constituinte armazene quais arcos foram usados para criá-lo, e cada arco deve indicar quais entradas da tabela foram usadas para satisfazer o lado direito dele. Além disso, cada símbolo não-terminal $N_{k,l}^{\ j}$ (significa um símbolo não-terminal do tipo N^j que gera parte da sentença de entrada desde a k-ésima até a l-ésima palavra) deve guardar o conjunto de arcos completados que podem criá-lo. Também deve guardar o conjunto de arcos completados que usam $N_{k,l}^{\ j}$ no seu lado direito. Com essas

[16] Em inglês *Chart-Parsing Bottom-Up*. Também é traduzido para o Português como Análise por Tabulação.

informações, é possível recuperar todas as árvores viáveis para a sentença partindo do símbolo não-terminal inicial e seguindo as aplicações possíveis.

O tempo de execução do algoritmo é, no pior caso, da ordem de n^3, onde n é o número de palavras da sentença. Entretanto, o número de possíveis árvores sintáticas cresce muito mais rápido que isso, já que depende inclusive da quantidade de regras gramaticais existentes. Isso torna ineficiente qualquer algoritmo que necessite da enumeração de todas as árvores. Portanto, no caso do analisador sintático algumas alterações são realizadas no processo da carta de análise de modo a eliminar na medida do possível as árvores candidatas pouco prováveis. Ou seja, a cada passo do algoritmo a tabela atual é analisada e suas subestruturas pouco prováveis são eliminadas. Isso reduz o crescimento do número de árvores sintáticas criadas na carta de análise.

O algoritmo é originalmente elaborado para a geração de árvores a partir de um conjunto de regras sintáticas com probabilidades, como no modelo GLCP. Para o modelo de dependência núcleo-léxico o critério de avaliação das subestruturas é alterado para refletir os parâmetros do modelo, mas a idéia de geração do algoritmo é a mesma. Outras adaptações para melhora de desempenho também são utilizadas, como uso de estratégia de procura por raio[17], limiares de descarte variáveis e estruturas de programação dinâmica para promover o descarte de subestruturas redundantes com notas mais baixas que outra análoga. O algoritmo final com suas adaptações é melhor descrito no Capítulo 3, seção 3.1.2.

2.11 Medidas de Avaliação de Acurácia de Modelo

A tarefa de medir a performance de um analisador sintático apresenta uma certa dificuldade prática, já que diferentes trabalhos geralmente usam diferentes métodos de medidas de erros para seus resultados obtidos. A mais aceita, ou mais usada às vezes com pequenas variações é a medida de PARSEVAL proposta em [Black et al 92]. Outra medida comum é o índice de colchetes atravessantes[18], que ao contrário da anterior representa uma medida de perda de performance e será apresentado a seguir.

A medida de PARSEVAL consiste em dois critérios para avaliação de recuperação de informação:

- $\text{Precisão} = \dfrac{\textit{quantidade de nós corretos}}{\textit{quantidade de nós produzidos}}$ (2.16)

[17] Em inglês *beam search*.
[18] Em Inglês *Crossing-Brackets*.

- Cobertura = $\frac{quantidade\ de\ nós\ corretos}{quantidade\ de\ nós\ na\ árvore\ do\ banco\ de\ árvores}$ (2.17)

Por exemplo, a árvore analisada e a árvore correta da sentença "the daring ugly dog barked at Mary" são respectivamente [[the_AT [daring_JJ ugly_JJ]] [dog_NN [barked_VBD [at_IN Mary_NP]]]] e [[the_AT [daring_JJ [ugly_JJ dog_NN]]] [barked_VBD [at_IN Mary_NP]]]. Neste caso, a etiqueta AT indica artigo definido, JJ adjetivo, NN nome comum no singular, VBD verbo no passado e NP nome próprio. Ambas as árvores são binárias e possuem 6 constituintes e portanto, neste caso particular, a medida de precisão e cobertura serão iguais. O número de constituinte corretos são 2, [barked at Mary] e [at Mary], e portanto as medidas de PARSEVAL são 2/6 = 0.33 .

Colchetes atravessantes é o número de constituintes produzidos que não estão compatíveis com a estrutura da árvore correta. No caso, ignoram-se as etiquetas de cada nó e só se consideram as palavras nas folhas e a estrutura da árvore gerada com a respectiva do banco de árvores. No exemplo supracitado, os constituintes na árvore correta são [at Mary] [barked at Mary], [ugly dog], [daring ugly dog], [the daring ugly dog] ,[the daring ugly dog barked at Mary]. Os constituintes da árvore gerada são [at Mary] [barked at Mary], [dog barked at Mary], [daring ugly], [the daring ugly], and [the daring ugly dog barked at Mary]. Existe um colchete atravessante neste caso, referido por [dog barked at Mary]. Veja na figura 2.3 a indicação do constituinte atravessante.

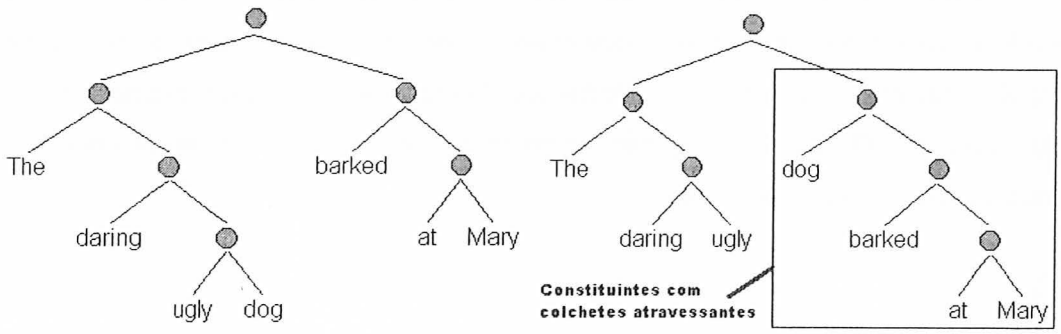


Figura 2.3: Exemplo de árvore sintática correta e a respectiva gerada apresentado em [Chen 98]

A figura 2.4 abaixo mostra mais alguns exemplos de colchetes atravessantes.

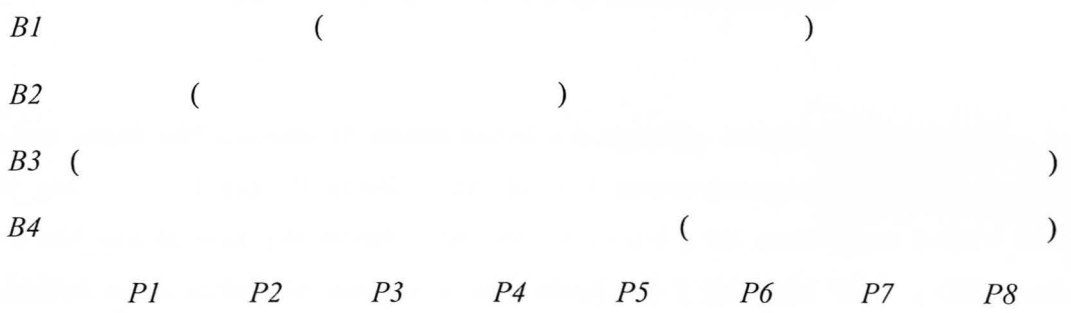


Figura 2.4: Exemplos de colchetes atravessantes. O colchete *B1* atravessa tanto *B2* como *B4*. Todas as outras permutações de colchetes são consistentes entre si. A idéia principal é que colchetes atravessantes não podem combinar entre si em uma única estrutura de árvore.

2.12 Conclusão

Este capítulo apresentou um breve resumo de muitos dos conceitos e técnicas pertinentes ao processo de desenvolvimento do analisador sintático orientado ao núcleo-léxico para a língua portuguesa. A literatura científica da área de processamento de linguagem natural (PLN) apresenta diversos outros trabalhos e resultados sobre outros mecanismos que podem se utilizados em tarefas análogas, mas que por bem foram ignorados neste resumo por não interferirem diretamente na compreensão do trabalho descrito no Capítulo 5.

Antes de prosseguir para a apresentação do trabalho realizado, o Capítulo 3 apresenta o método orientado ao núcleo-léxico desenvolvido em [Collins 99] com mais detalhes. O Capítulo 4 faz então uma análise do córpus em Português sobre o qual o treinamento e testes foram realizados. Os textos dos capítulos referidos assumem como corretamente compreendido o conteúdo deste capítulo de revisão.

Capítulo 3
Os Modelos Propostos pelo Método Orientado ao Núcleo-Léxico

Conforme apresentado no resumo da seção anterior, o método consiste em dois modelos. O segundo é baseado em história e tem resultado ligeiramente superior ao primeiro. Neste capítulo será resumido cada um destes modelos, apresentando justificativas para os respectivos acréscimos de informação contextual para fins de distinção. Os dois modelos serão comparados quanto as suas diferenças e por fim, serão apresentadas conclusões práticas para o andamento deste trabalho. Mas inicialmente, será apresentada a argumentação de Collins de defesa de seu modelo e também o mapeamento entre árvores e conjuntos de dependências.

Todos os modelos são aplicados para sentenças de entrada etiquetadas, isto é, cada palavra possui uma etiqueta de categoria morfossintática (CMS) associada. Além disso, as árvores geradas são lexicalizadas, o que significa que em cada nó, além do símbolo não-terminal existe um item lexical (palavra da sentença e sua respectiva etiqueta de CMS) núcleo, que é o principal para toda a estrutura dominada pelo nó.

3.1 Argumentação para o Uso de Dependências Léxicas

O analisador por dependências léxicas tem sua argumentação de bom desempenho baseada principalmente na boa escolha de parâmetros para se decompor a avaliação da árvore sintática diretamente por relacionamentos entre os itens lexicais. A proposta de Collins é que estes relacionamentos avaliados sobre os itens lexicais armazenam muitas informações necessárias para distinção. Ao mesmo tempo, as relações também guardam informações úteis para se ajustar os parâmetros, diminuindo o problema de dados esparsos (sub-treinamento).

A argumentação baseia-se também numa propriedade que as palavras de uma sentença têm chamada de localidade. Esta propriedade é análoga à propriedade de causalidade existente em Redes de Credibilidade[19]. Veja o seguinte exemplo adaptado de [RN 95] para se entender tal propriedade.

[19] Em inglês *Belief Networks*.

Seja o seguinte problema modelado com 5 variáveis binárias aleatórias: Uma pessoa tem uma casa com alarme anti-furto e está no trabalho. Ela tem dois vizinhos, João e Maria que são livres para chamá-la no trabalho caso o alarme dispare. O alarme pode ser disparado por dois motivos distintos: ou um assalto ou terremoto. A variável binária *A* indica quando ou não o alarme disparou. *E* indica se ocorreu um terremoto e *B* indica se ocorreu um assalto. *J* e *M* indicam quando João e Maria respectivamente chamaram o proprietário por telefone. O modelo assume a forma de P(*A, B, E, J, M*) que pode ser reescrito como uma cadeia de probabilidades condicionais. Vamos escolher a seguinte ordem de variáveis: *B E A J M*, então:

$$P(A, B, E, J, M) = P(B)\, P(E|B)\, P(A|E, B)\, P(J|A, E, B)\, P(M|J, A, E, B)$$

Podemos com conhecimento do caso assumir com tranqüilidade as seguintes hipóteses de independência de eventos:

1) $P(E|B) = P(E)$ (a chance de ocorrer um terremoto não depende em nada de ter ou não ocorrido um assalto)

2) $P(J|\, A, E, B) = P(J|\, A)$ (João telefonar independe da causa do disparo do alarme mas sim do alarme ter ou não disparado)

3) $P(M|A, E, B, J) = P(M|\, A)$ (Da mesma forma, Maria telefonar só depende do alarme ter disparado ou não. Estamos assumindo que João e Maria não se falam entre si, isto é, um nunca vai avisar ao outro quando já telefonou ou não.)

Assim, o modelo passa a ter a seguinte forma:

$$P(A, B, E, J, M) = P(B)\, P(E)\, P(A|E, B)\, P(J|A)\, P(M|A)$$

Veja que agora temos apenas 10 parâmetros a serem calculados (1 para $P(B)$ mais 1 para $P(E)$ mais 4 para $P(A|E, B)$ mais 2 para $P(J|A)$ mais 2 para $P(M|A)$). A quantidade de parâmetros por componente de probabilidade é dado pelas combinações possíveis de contexto, que no caso como todas as variáveis são aleatórias e binárias resulta em $2^{\text{tamanho do contexto}}$.

Agora imagine se tivéssemos usado a seguinte ordenação para as variáveis do modelo: *M J E B A*

$$P(A, B, E, J, M) = P(M)\, P(J|M)\, P(E|J, M)\, P(B|E, J, M)\, P(A|B, E, J, M)$$

Neste caso não podemos aplicar nenhuma hipótese de independência de acordo com o que conhecemos do problema (veja por exemplo $P(J|M)$, Maria ter chamado aumenta a chance do alarme ter disparado e portanto aumenta a chance de João ter telefonado também. Isso já não acontece em $P(M|A, E, B, J) = P(M|\, A)$ existente na ordenação anterior, já que apenas a

informação do alarme ter disparado já acrescenta todo contexto necessário para a distribuição de probabilidade, isto é, Maria ter telefonado ou não alteraria em nada o contexto). Por esta ordenação teríamos para o mesmo problema 31 parâmetros (1 para *P(M)* mais 2 para *P(J|M)* mais 4 para *P(E|J, M)* mais 8 para *P(B|E, J, M)* mais 16 para *P(A|B, E, J, M)*). Isto é mais de 3 vezes o necessário pela ordenação *B E A J M*.

O motivo desta diferença está na escolha da ordenação *B E A J M* levar em consideração a causalidade entre os eventos. Nela, insere-se primeiro os eventos que influenciam ou causam os seguintes e isso permite posteriormente a utilização de hipóteses de independência que diminuem a quantidade de parâmetros a serem calculados. Por este exemplo, pode-se ter uma breve idéia do quão crítico é a escolha de decomposição do problema em eventos menores para que a sua solução seja viável.

No caso do modelo para análise sintática, a localidade define o domínio no espaço sobre o qual a palavra núcleo influencia as demais palavras da sentença. Cada palavra da sentença afeta um domínio limitado na sentença. O propósito então é escolher uma decomposição que permita uma parametrização que reflita melhor a influência local dos itens lexicais núcleos. Assim, o item lexical núcleo deve ser gerado antes da estrutura que depende dele. Esta restrição leva-nos a uma derivação centrada nos núcleos da árvore. Com esta escolha de decomposição, aplicam-se hipóteses de independência naturalmente sobre relacionamentos que estão fora do domínio de influência (localidade), o que torna o modelo mais compacto sem prejudicar a sua acurácia.

3.2 Método Baseado em Dependências Léxicas

3.2.1 O Modelo Estatístico

O modelo se baseia no fato de que qualquer árvore sintática pode ser representada como um conjunto de *NPs* básicos *B* e um conjunto de dependências *D*. Desta forma, a árvore sintática assimilada a uma sentença de entrada *S* é aquela que maximiza:

$$\mathcal{P}(A|S) = \mathcal{P}(B, D|S) = \mathcal{P}(B|S) \times \mathcal{P}(D|S) \qquad (3.1)$$

NP básico ou minimal é um *NP* não recursivo, isto é, nenhum de seus filhos na árvore sintática é o símbolo não-terminal *NP*.

Uma dependência consiste em:

- um par de palavras <*modificador* → *núcleo*>
- da orientação entre as duas (posicionamento na sentença à esquerda ou à direita do núcleo)

- da relação presente (símbolo não-terminal do modificador, símbolo não-terminal do pai, símbolo não-terminal do núcleo na regra gramatical utilizada para a dependência)
- uma medida de distância entre os itens lexicais baseado na sentença de superfície ou nas subestruturas da árvore.

Uma sentença de entrada *S*, etiquetada em CMS, se apresenta da forma $< (p_1, t_1), (p_2, t_2) \ldots (p_n, t_n) >$, onde p_i é a i-ésima palavra da sentença e t_i é sua respectiva etiqueta de CMS. Dada uma sentença *S* e um conjunto *B*, a sentença reduzida $\overline{S}$ é a subseqüência de S que é formada através da retirada da pontuação e da redução de todos os *NPs* básicos para sua palavra núcleo. Exemplo:

S = João Paulo, o dono da DSC, pediu suas férias ontem.

B = [João Paulo], [o dono], [DSC], [suas férias], [ontem]

$\overline{S}$ = <João, NNP>, < dono, NN>, <da, IN>, <DSC, NNP>, <pediu, VBD>, <férias, NN>, <ontem, NN>

As dependências em *D* são definidas sobre a sentença reduzida e sua respectiva árvore sintática. Desta forma, temos:

$$\mathcal{P}(A|S) = \mathcal{P}(B, D|S) = \mathcal{P}(B|S) \times \mathcal{P}(D|S, B) \qquad (3.2)$$

Cada palavra da sentença reduzida modifica exatamente uma outra palavra, exceto a palavra núcleo da sentença (o item lexical de *S*). Definimos:

$AF(j) = (h_j, R_j)$, onde $j \in [1, m]$, *m* é o tamanho da sentença reduzida,
h_j é o índice do núcleo modifica pelo j-ésimo item lexical na sentença reduzida
R_j é a relação (conjunto de três símbolos não-terminais) entre o modificador e o núcleo da dependência

No caso do exemplo anterior, *férias* (sexto item lexical da sentença reduzida) modifica *pediu* (quinto item lexical da mesma) com relação *< NP, VP, VB-D >*, o que significa que *AF(6) = (5, < NP, VP, VB-D >)*

Temos que D = { *AF*(1), *AF*(2)... *AF*(*m*) } define a árvore *A* da sentença reduzida. O modelo assume que as dependências são independentes entre si, e portanto:

$$\mathcal{P}(D|S, B) = \prod_{j=1}^{m} \mathrm{P}(AF(j) \mid S, B) \qquad (3.3)$$

$\mathcal{P}(AF(j) \mid S, B)$ pode ser obtido pela seguinte aproximação:

$$\mathcal{P}(AF(j) \mid \mathrm{S}, \mathrm{B}) \approx \frac{\overline{F}(R_j \mid < \overline{p}_j, \overline{t}_j >, < \overline{p}_{h_j}, \overline{t}_{h_j} >, \Delta_{j,h_j})}{\sum_{k=1\ldots m, k \neq j, r \in \Re} \overline{F}(r \mid < \overline{p}_j, t_j >, < \overline{p}_k, \overline{t}_k >, \Delta_{j,k})} \qquad (3.4)$$

onde R é o domínio das relações, $\overline{F}(r \mid < a,b >, < c,d >, \Delta_{ia,ic})$ é uma aproximação da probabilidade do item lexical <*a*,*b*> modificar o item lexical <*c*,*d*>, com uma relação *r* e medida de distância entre eles igual a $\Delta_{ia,ic}$. (*a* e *c* são palavras do dicionário, *b* e *d* são etiquetas de CMS). Essa aproximação pode ser obtida através dos dados de treinamento, dividindo a quantidade de vezes que <*a*,*b*> modifica o item lexical <*c*,*d*> com uma relação *r* a uma distância $\Delta_{ia,ic}$, pela quantidade de vezes que <*a*,*b*> modifica o item lexical <*c*,*d*>, a uma distância $\Delta_{ia,ic}$.

O denominador da equação (3.4) é um fator de normalização e é constante para todo $j \in [1,m]$, com *B* e *S* fixos. Assim, maximizar $\mathcal{P}(D|S, B)$ é equivalente a maximizar o produto dos numeradores N_j da equação (3.4). Isso simplifica muito o processo de análise sintática.

A medida de distância é útil para detectar preferências léxicas, pois refina o contexto da dependência. Apesar de intuitivo, usar como medida a quantidade de palavras entre o modificador e o núcleo na sentença original não fornece bons resultados. Ao contrário disso, a medida de distância tenta armazenar informações relevantes sintaticamente. O exemplo mais simples e que fornece bons resultados é usar como medida dois valores binários:

- o primeiro indica se a sentença superficial entre o modificador e o núcleo é vazia (valor igual a 1 se for, 0 caso contrário). Essa sentença superficial é composta pelas palavras geradas pelos símbolos não-terminais entre o símbolo não-terminal do modificador e o símbolo não-terminal do núcleo na aplicação da regra gramatical que associa os dois.
- o segundo indica se existe um verbo nesta sentença superficial definida acima. (1 se existir, 0 caso contrário).

Como exemplo, para uma mesma dependência < *pela* → *aquisição* >, temos as seguintes medidas de distância nos três casos diferentes na tabela a seguir:

Sentença	Sentença superficial	distância
sua aquisição pela IBM	-	*10*
sua aquisição ontem pela IBM	ontem	*00*
sua aquisição anunciada ontem pela IBM	Anunciada ontem	*01*

Facilmente percebe-se que a medida de distância *11* é inviável.

O processo de definição dos *NPs* básicos pode ser visto como assimilação de etiquetas nas lacunas entre as palavras da sentença. As etiquetas seriam: **S**(começo de um *NP* básico), **C**(continuação), **E**(término), **B**(entre dois *NPs* básicos), **N**(nenhum *NP* básico ao lado). Assim por exemplo, a sentença :

[João Paulo], [o dono] da [DSC], tirou [suas férias] [ontem]

ficaria: João **C** Paulo **B** o **C** dono **E** da **S** DSC **E** tirou **S** suas **C** férias **B** ontem

A seqüência de etiquetas define univocamente os *NPs* básicos presentes na sentença. O modelo considera as palavras à direita e à esquerda de cada lacuna e se existe vírgula entre elas. A probabilidade de uma seqüência de *NP* em uma sentença não reduzida pode ser estimada como sendo:

$$\prod_{i=2...n} P(G_i \mid p_{i-1}, t_{i-1}, p_i, t_i, c_i) \qquad (3.5)$$

onde: $G_i \in \{\mathbf{S}, \mathbf{C}, \mathbf{E}, \mathbf{B}, \mathbf{N}\}$

p_i é a i-ésima palavra da sentença não reduzida

t_i é a etiqueta de CMS da i-ésima palavra da sentença não reduzida

$c_i = 1$ se existe uma vírgula entre a (i-1)-ésima e a i-ésima palavras da sentença não reduzida, ou 0 caso contrário

O principal problema que se percebe para o modelo é o domínio grande e esparso da aproximação $\overline{F}(r \mid < a,b >, < c,d >, \Delta_{ia,ic})$. A quantidade de contextos possíveis é igual a $T^2 \text{x} P^2 \text{x} M$, onde T é o número de etiquetas de CMS existentes, P é o número de palavras do dicionário, e M é o número de medidas de distância possíveis. Assim, para resolver o problema de estimativa de ocorrências não encontradas no treinamento, usa-se um modelo de ajustagem, no qual se reduz gradativamente o tamanho do contexto até se ter uma contagem considerável para se fazer uma estimativa segura. Essa estimativa pode ser uma média ponderada das estimativas nos diferentes níveis de contextos percorridos. O mesmo comentário é válido para a estimativa dos parâmetros $P(G_i \mid p_{i-1}, t_{i-1}, p_i, t_i, c_i)$.

3.2.2 Função de Mapeamento Entre Árvores Sintáticas e Grupos de Dependências

Representar as árvores sintáticas como um conjunto de dependências é melhor que usar uma representação por regras de GLCP simples já que naquela representação tem-se diretamente o uso da informação léxica. E isso é o mais relevante como critério de distinção segundo a argumentação apresentada na seção 3.1 deste capítulo.

A função de mapeamento entre árvores sintáticas e grupos de dependências pode ser dividida em duas etapas: a primeira faz a lexicalização da árvore e a segunda parte deriva as (*n*-1) dependências para cada regra com *n* filhos. O primeiro passo já foi descrito na seção 2.3 deste trabalho, mas de forma resumida ele consiste em modificar cada não-terminal da árvore para incluir uma palavra núcleo (uma palavra da sentença) em si.

O segundo passo consiste em identificar as dependências presentes na árvore lexicalizada. Uma dependência é uma relação entre duas palavras, uma o modificador e a outra a modificada, que então será denotada por *modificador* → *núcleo*. As dependências são derivadas da seguinte forma:

- Para cada regra $X \rightarrow Y_1 .. .Y_n$ tal que: 1) $Y_1 .. .Y_n$ são não-terminais; 2) $n \geq 2$; 3) h = núcleo $(X \rightarrow Y_1 .. .Y_n)$. Cada regra gera $(n\text{-}1)$ dependências da forma $núcleo(Y_i) \rightarrow núcleo(Y_h)$ para cada $1 \leq i \leq n, i \neq h$
- Se X é o não-terminal raiz da árvore e x é seu núcleo então $x \rightarrow$ COMEÇO é a dependência

Veja na figura 3.1 uma árvore sintática lexicalizada e o conjunto completo de dependências presentes nela. Por exemplo, a regra *VP(empilharam)* → *VBD(empilharam) NP(sacos) PP(dentro)* é responsável por duas dependências: <*sacos* → *empilharam*> e < *dentro*→ *empilharam*>.

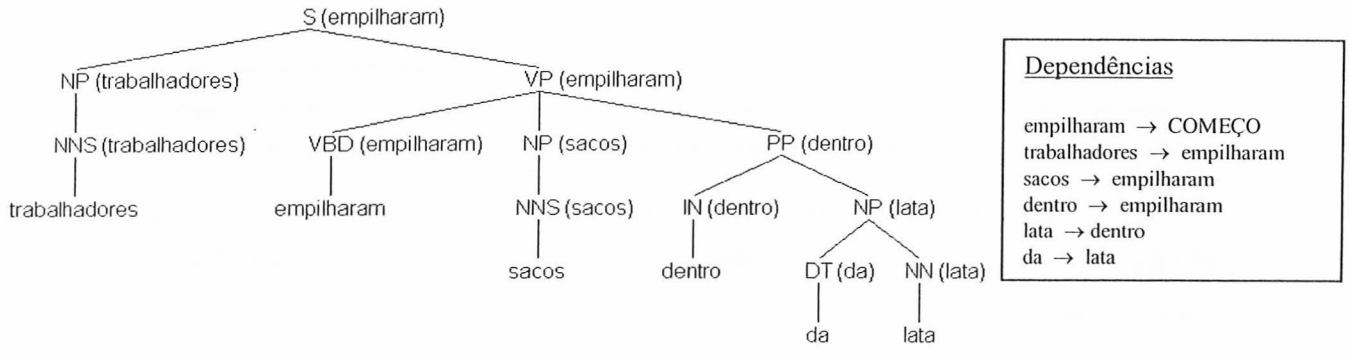


Figura 3.1: Exemplo de árvore sintática lexicalizada com seu respectivo conjunto de dependências [Collins 99]

(S[empilharam](NP[trabalhadores](NNS [trabalhadores] (trabalhadores))) (VP [empilharam](VBD [empilharam] (empilharam)) (NP [sacos] (NNS [sacos] (sacos))) (PP [dentro] (IN [dentro] (dentro)) (NP [lata] (DT [da] (da)) (NN [lata] (lata))))

3.2.3 Algoritmo de Aplicação: Adaptação do Algoritmo de Carta de Análise

O algoritmo de aplicação de análise sintática utilizado é uma adaptação do algoritmo simples de carta de análise de baixo para cima (ascendente). Não existem regras gramaticais, mas na prática a geração de dependências se limita as tuplas <modificador, pai, núcleo> vistas no treinamento. Assim, o espaço de procura se limita ao de todas as árvores cujos trios de não-terminais vistos no treinamento, já que as demais terão probabilidade zero segundo o modelo. As probabilidades dos *NPs* básicos na carta de análise são calculados usando-se a fórmula 3.5 enquanto as probabilidades dos outros constituintes são derivados pelas dependências e *NPs* básicos presentes no constituinte.

O algoritmo também descarta os constituintes que geram um mesmo grupo de palavras com as mesmas etiquetas, núcleos e medidas de distância entre o núcleo e os constituintes extremos da direita e da esquerda que outro já existente com nota maior. A figura abaixo exemplifica como um novo constituinte é gerado. Basicamente a nota do constituinte final é igual ao produto das notas dos constituintes que se unem, vezes a nota da dependência que é gerada pela junção e vezes a nota das etiquetas de *NPs* básicos geradas.

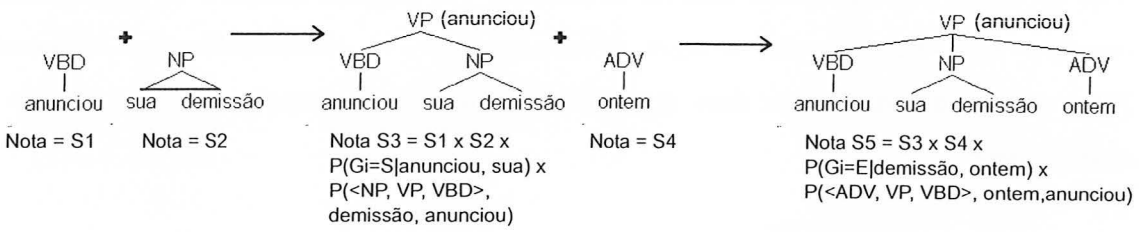


Figura 3.2: Visualização de como dois constituintes se unem para formar um novo constituinte. Cada operação gera dois novos termos de probabilidade, um para a etiqueta de lacuna NB básica entre os dois constituintes e uma outra probabilidade para a dependência entre os itens lexicais principais dos dois constituintes.

Duas técnicas são implementadas também no algoritmo para melhorar a performance de procura. A primeira define limiares de descartes decrescentes e a outra define uma estratégia se busca por raio.

O limiar de descarte consiste em se descartar à medida que a carta de análise é construída os constituintes cujas probabilidades são inferiores a um certo limiar parametrizado. Se uma árvore sintática é encontrada, ela deve ser a que tem a maior nota já que todas as que foram descartadas possuíam constituintes com notas inferiores ao limiar e portanto com nota inferior a da árvore encontrada. Se nenhuma árvore sintática é recuperada, o limiar de descarte é diminuído e a busca começa novamente. O processo continua até se encontrar a primeira árvore sintática.

A busca por raio limita a probabilidade de constituintes de acordo com a maior probabilidade P_h que gera a mesma seqüência de palavras. Todos constituintes que geram as

mesmas palavras (ignorando todo o resto, etiquetas, núcleos e regras) devem ter probabilidade maior que P_h/β para uma constante de raio β. Constituintes abaixo deste raio são descartados. Esta estratégia aumenta muito a eficiência do algoritmo de procura, mas introduz erros de procura dependentes do valor de β. Collins tabela então o relacionamento entre diversos valores de β, o desempenho obtido (sentenças analisadas/segundo) e a acurácia resultante.

3.3 Três Modelos Estatísticos de Análise Sintática Gerativos

O modelo anterior possui algumas deficiências que prejudicam sua precisão. A primeira delas é que ele não define uma probabilidade conjunta exatamente, isto é, $\sum P(A,S) = 1$. Além disso, não existe tratamento bem definido para as regras unárias da gramática. Por fim, o contexto é limitado a características da sentença de superfície apenas, sem levar em consideração qualquer subestrutura gerada anteriormente na árvore.

Pensando nisso, em [Collins 99], desenvolveu-se um conjunto de três modelos estatísticos baseados em história, que resolvessem tais deficiências. Tais modelos também têm a vantagem de usar uma medida de distância mais aprimorada e o fato da etiquetagem em categoria morfossintática ser acoplada ao processo de análise sintática.

O primeiro modelo parte da teoria da GLCP. Ou seja, dado que a geração de uma árvore sintática é feita via aplicação em seqüência de n regras gramaticais da forma $LHS_i \rightarrow RHS_i$, a probabilidade $\mathcal{P}(A,S)$ é dada por:

$$\prod_{i=1...n} P(RHS_i \mid LHS_i) \qquad (3.6)$$

O problema então passa a ser como estimar $\mathcal{P}(LHS_i \rightarrow RHS_i)$. No caso do primeiro modelo as regras gramaticais aplicadas são estendidas para conter além dos símbolos não-terminais, os itens lexicais correspondentes. Ou seja, o modelo estima probabilidades para regras do tipo: *S*(comprou, *VBD*) → *NP*(semana, *NN*) *NP*(IBM, *NNP*) *VP*(comprou, *VBD*). Estimar a probabilidade deste tipo de regra estendida não pode ser feito via contagem direta sobre ocorrência nos dados de treinamento, pois o domínio do contexto é muito amplo. Assim, para uma regra gramatical da forma:

$$E(h) \rightarrow L_n(l_n) \ldots L_1(l_1)\ H(h)\ R_1(r_1) \ldots R_m(r_m) \qquad (3.7)$$

(onde h é o item lexical do núcleo do lado direito, $l_1 \ldots l_n$ e $r_1 \ldots r_m$ são os itens lexicais dos demais símbolos não-terminais do lado direito da regra), a sua probabilidade é estimada como sendo:

$$\mathcal{P}(L_n(l_n) \ldots L_1(l_1)\, H(h)\, R_1(r_1) \ldots R_m(r_m) \mid E, h) = \mathcal{P}(H \mid E,h) \times \prod_{i=1\ldots n+1} P(L_i(l_i) \mid E, h, H, dist_l(i-1)) \times \prod_{i=1\ldots m+1} P(R_i(r_i) \mid E, h, H, dist_r(i-1)) \quad (3.8)$$

Para que o modelo tenha comportamento de uma distribuição conjunta, são acrescentados os símbolos não-terminais L_{n+1} e R_{m+1}, ambos definidos como símbolo especial *STOP*, indicando que terminam a seqüência de modificadores a esquerda e a direita do núcleo, respectivamente. As medidas de distância $dist_r$ e $dist_l$ são semelhantes à aquela utilizada pelo modelo anterior. Por exemplo, para a regra *S*(comprou, *VBD*) → *NP*(semana, *NN*) *NP*(IBM, *NNP*) *VP*(comprou, *VBD*) , a probabilidade seria estimada como:

$\mathcal{P}$(*S*(*comprou*, *VBD*) → *NP*(*semana*, *NN*) *NP*(*IBM*, *NNP*) *VP*(*comprou*, *VBD*)) = $\mathcal{P}_h$(*VP* | *S*, (*comprou*, *VBD*)) x $\mathcal{P}_l$(*NP*(IBM, *NNP*) | *S*, (*comprou*, *VBD*)) x $\mathcal{P}_l$(*NP*(*semana*, *NN*) | *S*, (*comprou*, *VBD*)) x $\mathcal{P}_l$(*STOP* | *S*, (*comprou*, *VBD*)) x $\mathcal{P}_r$(*STOP* | *S*, (*comprou*, *VBD*))

Esse primeiro modelo obteve resultados ligeiramente superiores ao do modelo anterior (melhora em torno de 1,7% na precisão). Os dois modelos derivados deste resolvem problemas particulares de não-gramaticalidade encontrados neste primeiro. A melhora final obtida é de cerca de 2.4% em média em relação ao modelo da seção 3.2.

O segundo modelo diferencia todos os símbolos não-terminais que são complementos de seus respectivos que são adjuntos. Isso praticamente dobra a quantidade de símbolos não-terminais presentes na gramática. A vantagem obtida com tal modificação é que estruturas incorretas gramaticalmente, como por exemplo, dois complementos nominais modificando um verbo com relação sujeito-predicado, obteriam probabilidades muito baixas e então seriam corretamente descartadas. Para que isso seja possível, o segundo modelo também gera parâmetros para subcategorias à esquerda e à direita do núcleo da regra gramatical. Esse tipo de parâmetro consegue avaliar relações entre os modificadores de um mesmo núcleo, ou seja, termina com a total independência de geração dos modificadores (hipótese simplificadora e com falhas).

O terceiro modelo trata dos problemas acarretados por movimentos-wh na estrutura das árvores sintáticas. Tal tarefa é complexa o bastante para merecer um tratamento estatístico, junto com a geração da árvore. Acrescentando parâmetros ao modelo que reflitam tais movimentos, a precisão final deve melhorar. Para se resolver tal problema, o terceiro modelo

cria índices *LACUNA*[20] para cada *WHNP*, presente na árvore sintática. As regras gramaticais são alteradas para terem um valor binário de lacuna (positivo ou negativo), que serve para indicar a necessidade de geração de uma *LACUNA* na subestrutura gerada abaixo. As modificações propostas tanto no segundo como no terceiro modelo obrigam em mudanças de anotação nas árvores sintáticas usadas para treinamento e teste.

3.4 Vantagens do Modelo 3.3 Sobre o Modelo 3.2

Os modelos gerativos orientados ao núcleo-léxico desenvolvidos no segundo trabalho de Michael Collins têm algumas vantagens em relação ao primeiro, de dependências léxicas mais simples, entre elas:

- Os modelos não são matematicamente deficientes (isto é, $\sum P(A \mid S) = 1$) e as regras unárias são consideradas de um modo natural previsto pela modelagem.
- A medida de distância é um pouco diferente e aperfeiçoada. A variável de adjacência corresponde diretamente a estruturas de ramificação à direita.
- Os três modelos apresentam uma melhora de acurácia em relação ao do trabalho anterior de 1,5% a 2,2% em precisão e de 1,9% a 2,4% em cobertura,
- As restrições do modelo anterior eram executadas sobre a sentença superficial apenas e não em qualquer subestrutura anteriormente gerada, conforme é possível nestes modelos mais recentes.
- As etiquetas morfossintáticas estão naturalmente incorporadas nos modelos recentes.
- Os modelos recentes definem uma probabilidade conjunta *P(A,S)* e podem portanto serem usados como modelo de linguagem para sistemas de reconhecimento de fala ou tradução automática de textos. Significa também que podem ser treinados de uma forma não supervisionada, usando o algoritmo *EM*, fato não possível no modelo anterior que cria um distribuição de probabilidade condicional *P(A|S)*.
- O modelo gerativo mais completo apresenta ainda os seguintes elementos lingüísticos úteis para alguns sistemas mais avançados de processamento de linguagem natural: informações sobre movimentos-wh, subcategorias e distinção entre complementos e adjuntos.

Apesar destas vantagens, o modelo orientado ao núcleo-léxico implementado nesta dissertação foi o primeiro, baseado em dependências léxicas. Esta escolha pode ser explicada pela

[20] Em inglês *TRACE*.

simplicidade do modelo e pelos seus relativos bons resultados, apesar de inferiores aos dos modelos análogos mais complexos. A próxima seção faz uma pequena comparação de acurácia entre diversos métodos e traça alguns comentários para embasar a escolha de implementação.

3.5 Conclusão

Em [Charniak 97] é feito uma comparação de resultados para a análise sintática de textos em Inglês entre vários analisadores sintáticos citados no Capítulo 2, incluindo o primeiro modelo baseado em dependências léxicas de Collins. Os resultados estão listados abaixo na tabela 3.1

	LR	LP	CB	0 CB	2 CB
<= 40 palavras (2245 frases)					
GLCP	71,7	75,8	2,03	39,5	68,1
Magerman	84,6	84,9	1,26	56,6	81,4
Collins	85,8	86,3	1,14	59,9	83,6
Charniak	87,5	87,4	1,0	62,1	86,1
<= 100 palavras (2416 frases)					
GLCP	70,6	74,8	2,37	37,2	64,5
Magerman	84,0	84,3	1,46	54,0	78,8
Collins	85,3	85,7	1,32	57,2	80,8
Charniak	86,7	86,6	1,20	59,5	83,2

Tabela 3.1: Resultados obtidos de vários modelos de análise sintática

(**LR**: Cobertura, **LP**: Precisão, **CB**: índice médio de parênteses atravessantes por sentença,

0CB: percentual de sentenças com zero parênteses atravessantes,

2CB: percentual de sentenças com até 2 parênteses atravessantes)

O trabalho ainda compara resultados de outras otimizações aplicadas sobre o método de Charniak baseado em gramática livre de contexto e estatísticas sobre palavras (objeto do artigo em si). Mas as observações mais interessantes são a respeito da comparação com os trabalhos de [Collins 96] e [Magerman 95], cujos sistemas foram aplicados no mesmo domínio de testes, o banco de árvores do *Penn Wall Street Journal*. Segundo Charniak, os três sistemas são muito parecidos entre si. Todos possuem pouco conhecimento das características do Inglês e calculam as estatísticas que precisam a partir de um córpus de treinamento. Em todos também existe a noção de um núcleo-léxico para cada constituinte da árvore e usam informações sobre este núcleo em suas estatísticas. Além disso, todos os três sistemas restringem a informações sobre o item lexical a até dois níveis na estrutura da árvore, isto é, usam estatísticas que levam em consideração o núcleo do constituinte e o núcleo de seu pai, mas não o de seu avô. E por fim,

todos retornam como árvore correta aquela com a maior probabilidade na distribuição de probabilidade suavizada que definem.

Pode-se ver pelos resultados da tabela 3.1 que os três sistemas possuem resultados de precisão bastante parecidos entre si, onde o melhor possui uma redução de até 18% no erro em relação ao pior. Apesar de terem muitas características em comum, eles divergem entre si em vários aspectos, como por exemplo, o modelo do Charniak é o único que cria uma distribuição conjunta $\hat{P}(A, S)$ enquanto os outros dois geram uma distribuição $\hat{P}(A|S)$. Estas diferenças criam vantagens entre um método e outro, geralmente beneficiando o mais recente, mas não chegam a alterar enormemente a precisão dos resultados finais. Isto leva a crer que os três sistemas são variações de uma mesma idéia: a de se usar estatísticas sobre as palavras como critério de distinção entre árvores sintáticas. E que portanto, possuem resultados parecidos entre si, diferenciando-se apenas por escolhas de parametrização mais exatas, modelos mais matematicamente exatos e técnicas heurísticas de melhora de desempenho (geralmente baseado em tratamento dos casos de erros, tentativa e erro e algum conhecimento da língua como regras de pontuação). Assim, o grande ganho de performance obtido em relação a GLCP se deve a idéia principal de se usar informações léxicas como critério de decisão.

O trabalho implementado nesta dissertação utiliza o modelo mais simples de dependências léxicas e tenta chegar a algumas conclusões aplicadas ao Português. Para isso, também foi implementado um sistema GLCP padrão.

Capítulo 4
O Córpus Utilizado para o Português: Tycho Brahe

4.1 Apresentação

O córpus Tycho Brahe está sendo desenvolvido como parte de um projeto que pretende estudar a evolução do Português Clássico (PEC) para o Português Europeu Moderno (PEM) [AlvFinger 99]. Os textos utilizados neste projeto são de autores nascidos no período desde o século XVI até XIX. Com a construção dos textos sintaticamente anotados, será possível, por exemplo, estabelecer como a mudança prosódica dos padrões rítmicos do Português levou a mudanças sintáticas, em especial na colocação dos pronomes clíticos.

Esse córpus histórico compreende atualmente os seguintes tipos de textos:

1. Textos originais, ortograficamente transcritos com informações sobre as edições, autores, datas e comentários dos editores.
2. Os mesmos textos com cada palavra marcada morfossintaticamente, utilizados para desenvolvimento nos projetos anteriores de etiquetagem automática de categoria morfossintática aplicada ao Português (exemplos em [AlvFinger 99] e [Archias 03]).
3. Os mesmos textos anotados com a estrutura de árvore sintática representada através de agrupamento por parênteses e etiquetas não-terminais da árvore.

O terceiro grupo de textos representa o material de base para treinamento e testes do trabalho desta dissertação. Hoje ele compreende textos de vários autores, mas apenas uma pequena parte está revisada quanto à validade da anotação sintática adotada. Para minimizar o efeito de falsos resultados gerados por erros de anotação apenas esta parcela de textos anotados e revisados foi utilizada no trabalho. Isso representa 4317 sentenças aprovadas, das quais 3715 foram utilizadas para treinamento, 432 para testes e 170 foram descartadas por erro ou terem tamanho inferior a 5 itens lexicais. A base toda ficou concentrada em um arquivo chamado *'AIRES_02.mjs'* que são as sentenças presentes em 'AIRES_02.mjc'[21], convertidas da forma anotada 4.3 para a 4.4.

[21] Texto anotado do original de AIRES, Matias. Reflexões sobre a Vaidade dos Homens e Carta sobre a Fortuna (seleção, prefácio e notas por Jacinto do Prado Coelho e Violeta Crespo Figueiredo). Lisboa, Imprensa Nacional - Casa da Moeda. 1980.

Vale notar que estamos construindo um analisador sintático com o objetivo de auxiliar a construção do córpus anotado sintaticamente. Isso diferencia muito o nosso trabalho em relação à maioria dos trabalhos desenvolvidos para o Inglês, principalmente em relação ao tamanho da base de treinamento, ao acúmulo de dados estatísticos e à base de documentação e conhecimento sobre as características dos textos utilizados.

4.2 Características do Córpus Anotado Sintaticamente

4.2.1 Características dos Textos

O córpus Tycho Brahe compreende textos históricos de autores de épocas, convenções ortográficas e estilos distintos. Isto o torna bastante diferente de outros córpus utilizados para o Inglês, como o de estilo jornalístico do *Penn Treebank Wall Street Journal Corpus* (PTWSJC) utilizado em [Collins 99] ou mesmo os de manuais técnicos da IBM já utilizados em outros trabalhos.

Na nossa língua, existe um número maior de flexões que as palavras podem assumir, como de gênero, número e tempo no caso dos verbos. Algumas palavras, como o pronome *'você'*, aparecem no córpus com mais de uma grafia, o que torna ainda mais difícil a coleta de estatísticas precisas sobre o córpus. Em outras palavras, sem o uso de um pré-processador que aglomere as mesmas palavras escritas de forma distinta, torna-se necessário um córpus ainda maior para ser '*estatisticamente relevante*' para o treinamento do analisador.

Em particular, o Português do século XVII tem uma ordem de palavras muito mais livre que o Inglês ou Português moderno, e isto também depende do estilo literário ao qual pertence o texto. Esta característica enfraquece uma hipótese utilizada nos trabalhos orientados ao núcleo-léxico para o Inglês, na qual a ordem entre a palavra modificada e a modificadora numa dependência era estatisticamente relevante para a viabilidade sintática da estrutura gerada.

Acrescenta-se também o fato do Português ser uma língua morfologicamente muito mais rica que o Inglês, o que torna o número de etiquetas de anotação sintáticas (apresentadas na seção a seguir) muitas vezes maior que o número de etiquetas utilizadas nas árvores sintáticas para o Inglês. Existem também fortes diferenças nas expressões idiomáticas utilizadas em cada língua, como por exemplo, o verbo '*ter*' é largamente utilizado no Português, aparecendo em várias expressões com diversos papéis idiomáticos. O correspondente '*to have*' tem um uso mais restrito no Inglês. O fenômeno inverso ocorre entre o verbo '*to be*' do Inglês e o nosso verbo '*ser*'. Exemplos destas diferenças podem ser vistas em [Schütz 03].

4.2.2 Etiquetas Utilizadas

No momento, o córpus utiliza basicamente 380 etiquetas sintáticas diferentes, das quais 281 foram vistas na fase de treinamento. Cada etiqueta é composta por um tipo principal e por um conjunto de complementos que acrescentam informações sintáticas adicionais (indicados por conectivos '+' ou '-'). Ao todo, existem 19 tipos principais de etiquetas. Um exemplo de etiqueta sintática completa é o item lexical '*padres*', um nome (*N*) com a indicação *P* de plural:

(N-P padres) (4.1)

Outras informações como contrações, por exemplo: '*de*' + '*o*' = '*do*'/ P+D presente em (P d-) (DP (D –o) (NP (NPR Padre)), também são codificadas, assim como, deslocamentos, traços e lacunas. A descrição de todas as etiquetas, com seus significados e explicação de uso pode ser vista no manual de anotação sintática do córpus Tycho Brahe ([Britto 02]).

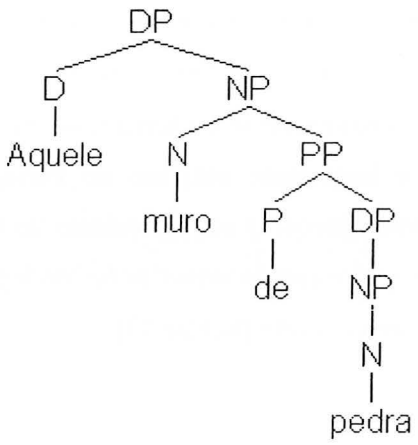
4.2.3 Características do Estilo da Anotação e Representação

O sistema de anotação sintática é aplicado em sentenças já marcadas com a categoria morfossintática de cada palavra, como em 4.2, sob o formato 4.3 que é equivalente a 4.4. Numa representação gráfica tradicional igual a 4.5, propriedades básicas das estruturas, como dependências, são devidamente codificadas em 4.3 e 4.4.

Aquele/D muro/N de/P pedra/N (4.2)

```
(DP      (D Aquele)
         (NP    (N muro)
                (PP    (P de)
                       (DP    (NP    (N pedra))))))
```
(4.3)

(DP (D Aquele) (NP (N muro) (PP (P de) (DP (NP (N pedra)))))) (4.4)



(4.5)

Não existe qualquer informação de dicionário vinculada aos itens lexicais da sentença, apenas a sua categoria morfossintática. Também não existe qualquer estrutura de agrupamento (*clustering*), ou código binário (*binary string*) de propriedades, nem mesmo informação de radical ou indicação de possíveis flexões presentes (como um desconjugador de verbos).

Por motivos de simplificação de custo computacional, o sistema não codifica projeções intermediárias, o que reduz a quantidade de nós gerados (exemplos podem ser vistos na seção 1.3 de [Britto 02]). Mesmo assim, existe uma quantidade considerável de regras unárias (um pai para um único filho) presente nas estruturas das árvores geradas. Essas projeções representam 32,6% das regras gramaticais coletadas no treinamento. Isso representa um complicador na ordem de busca do algoritmo sintático, já que para cada entrada da carta de análise pode-se, com uma boa probabilidade, gerar uma outra entrada, que não agrega nenhuma palavra à estrutura, mas que tem uma etiqueta diferente da anterior. Aplicando-se isso a cada entrada da carta, ocorre uma explosão na quantidade de permutações possíveis para o varredura e geração de novas entradas na carta de análise.

4.3 Adaptações Necessárias Para o Trabalho

4.3.1 Adaptações nas Etiquetas

As etiquetas utilizadas na anotação do córpus Tycho Brahe apresentam informações sintáticas detalhadas de acordo com a estrutura dominada pela etiqueta e com o tipo principal da etiqueta. Como o córpus anotado e revisado ainda é relativamente pequeno para um treinamento de dependências e regras gramaticais contendo 281 tipos de etiquetas sintáticas diferentes gerando 7349 regras gramaticais diferentes, tivemos que diminuir o problema. Basicamente, foram retiradas todas as informações complementares presentes nas etiquetas e mantivemos apenas seu tipo principal. Isso reduziu para 19 tipos de etiquetas sintáticas, que é menor, porém mais próximo dos 26 tipos de etiquetas que foram utilizados em [Collins 99] para o Inglês. Mesmo assim existe uma grande diferença no tamanho do córpus que Collins utilizou, cerca de 10 vezes maior que o utilizado em nossos experimentos.

Esta modificação representa na verdade uma escolha em se reduzir a quantidade de informação contextual sintática para aumentar o número de amostragens dos eventos vistos, reduzindo o problema de sub-treinamento do sistema.

Outra modificação foi eliminar as últimas derivações das árvores sintáticas, isto é, as folhas não são mais os itens lexicais, mas sim a categoria morfossintática de cada palavra, acrescentado como núcleo o respectivo item lexical removido. Isso foi aplicado tanto na GLCP como no método orientado ao núcleo-léxico, já que estas derivações finais não são captadas pelos modelos. De fato a GLCP não usa qualquer informação léxica para montar a árvore, partindo apenas da seqüência de categorias morfossintáticas da sentença para então montar a árvore associada. No caso do método orientado ao núcleo-léxico, estas relações unárias finais não acrescentam qualquer dependência na árvore e são tratadas de forma igual em todas as árvores candidatas geradas, não contando como critério de distinção entre elas. Podem assim, serem ignoradas a título de simplificação e ganho de performance.

4.3.2 Lexicalização das Árvores: Uso das Regras de Núcleos

A partir do conjunto de etiquetas reduzidas, todos os demais processos presentes no sistema de geração do analisador sintático foram executados de forma mais simples. O processo de lexicalização a princípio consiste em se definir um nó do lado direito para cada regra de derivação como sendo o nó principal da regra, de acordo com os critérios definidos em 3.1.

No caso de regras unárias, aquelas que só tem um constituinte no lado direito (por exemplo: $< NP \rightarrow IP >$) a lexicalização do nó pai fica fácil já que não há nenhuma decisão a se fazer, mas nos demais casos de regras com mais de um nó na derivação do lado direito uma escolha precisa ser feita. O problema é que geralmente a quantidade de regras a serem lexicalizadas é muito grande, tornando em uma tarefa humanamente exaustiva a solução de lexicalizar uma a uma. Nos testes executados neste trabalho o córpus de treinamento da GLCP gerou 45195[22] regras gramaticais de 3701 tipos diferentes, das quais 2413 (65.20%) foram amostradas apenas uma vez de um treinamento sobre 3715 sentenças anotadas contendo ao todo 58524 itens lexicais. Existem 84 tipos de regras unárias (2.27% do total de tipos) que foram aplicadas 14716 vezes (32.56% do total). A solução adotada para lexicalizar todas as regras gramaticais obtidas foi o uso de lista ordenada por prioridade para cada etiqueta que necessite de decisão para o nó principal. A tabela 4.1 apresenta todas as listas de prioridades utilizadas por etiqueta.

[22] No caso da GLCP, os nós folhas das árvores sintáticas são as etiquetas morfossintáticas e não as palavras da sentença, já que a informação do item lexical não é utilizada para a geração da árvore. Assim, não foram contadas nenhuma das regras do tipo <ETIQUETA → PALAVRA> o que reduz a quantidade de regras geradas.

ETIQUETA A ESQUERDA	LISTA DE PRIORIDADE
ADVP	ADV CP PP Q ADJ CONJ SENAO NEG ADJP FP NP CONJS ADVP CONJP , .
ADJP	ADJ CONJP ADV PP VB ADJP CP CONJ ADVP Q NP NEG IP WADJP C PRO$ FP NUMP SENAO P NPR , .
CP	IP CONJP WNP CONJ CP C CONJS PP WADVP WPP ADV NP WADJP ADVP INTJ NEG WQ WPRO WD ADJP N FP WQP WADV VB SENAO , .
CONJP	CONJ NP IP PP CONJP VB ADV CONJS NEG CP OUTRO WPP N ADVP ADJ WNP ADJX NX ADJP D , .
FRAG	ADJP CP PP NP IP ADVP NEG ADV , .
IP	NP VB PP CONJ CP SR IP ADV NEG ADVP ADJP TR HV FP CONJP ET VB+SE Q VB+CL FRAG ADJ P N SENAO DEM QP CL HV+SE LATIN SR+CL CONJS C , .
NP	N D CP CONJP ADJ PP NP Q NPR ADV ADJP PRO$ VB OUTRO CONJ FP NEG IP NUM DEM P PRO ADVP SENAO NUMP SR WPRO$ WPRO , .
NUMP	NUM CONJ
PP	P CONJP PP NP IP CP ADV NEG CONJ FP ADVP SENAO ADJP N D ADJ PRO NPR WPRO Q VB , .
VB	VB CONJ TR ,
WADJP	ADJ WADV WD ADJP
WADVP	WADV WD ADV PP ADVP P Q
WNP	N WD WPRO ADJ WPRO$ PP ADJP CONJP D ADV Q ADVP CP NPR VB OUTRO WNP FP ,
WPP	P N WD WPRO$ NP WPRO CP WADV ,

Tabela 4.1: Listas de prioridades para lexicalização. O nó pai é o não-terminal à esquerda da regra gramatical.

Como exemplo de utilização da tabela, seja a regra $< X \rightarrow Y_1 \ldots Y_n >$ onde X é um *VB*, o algoritmo procura a partir da esquerda da seqüência $< Y_1 \ldots Y_n >$ pelo primeiro Y_i do tipo *VB*; se nenhum *VB* foi encontrado, então o algoritmo procura pelo primeiro Y_i do tipo *CONJ*; se nenhum *CONJ* foi encontrado, então o algoritmo procura pelo primeiro Y_i do tipo *TR* e assim por diante. Se o algoritmo não encontrar nenhum item da sua lista de prioridade no lado direito da regra, escolhe-se o filho mais à esquerda (Y_1) como principal da regra.

No caso dos testes executados neste trabalho, as listas de prioridades para lexicalização montadas a partir do treinamento foram capazes de lexicalizar todas as árvores do córpus de teste sem o uso da regra final do nó mais à esquerda (Y_1).

Esta solução de lexicalização apesar de não tratar caso a caso as regras gramaticais, apresenta um resultado final satisfatório para o problema, desde que as listas de prioridades estejam ordenadas da melhor forma possível. Esta solução é uma simplificação do algoritmo utilizado em [Magerman 95].

4.3.3 Pontuação

A última modificação deve-se à forma como foi implementado o método orientado ao núcleo-léxico. As pontuações das sentenças (vírgula, ponto, dois pontos, ponto e vírgula e outros) são descartadas na geração das árvores sintáticas e coleta de parâmetros do modelo orientado ao núcleo-léxico, mas são consideradas no modelo GLCP. Assim, decidimos não considerá-los nos cálculos de precisão, cobertura e colchetes atravessantes dos resultados de nenhum dos dois analisadores sintáticos desenvolvidos, por questão de justiça na comparação de resultados.

Isso não significa que a pontuação da sentença não é utilizada nos contextos dos parâmetros coletados pelo método orientado ao núcleo-léxico. Ela é avaliada na medida de distância utilizada para cada dependência, conforme está descrito em 4.4.6. De fato, o que muda é que os nós etiquetados com ',' ou '*CLOSE*', ou '.' são ignorados na avaliação de performance dos algoritmos.

4.4 Aspectos Relevantes do Córpus para a Aplicabilidade do Modelo Núcleo-Léxico

Nos trabalhos anteriores, os analisadores orientados ao núcleo-léxico aplicados ao Inglês baseiam-se em algumas hipóteses a respeito das estruturas lingüísticas presentes na língua inglesa. Estas hipóteses são relevantes para o bom desempenho do método e devem ser consideradas com cuidado para a análise da aplicabilidade para o Português e avaliação de seus resultados.

Este tipo de análise pode ser fundamentada no conhecimento e estudo das características específicas da língua Portuguesa, o que normalmente é feito por um especialista da área de lingüística. Portanto, esta seção está longe de ser um estudo conclusivo dos aspectos citados abaixo para o Português. Seu objetivo é em levantar tais aspectos que influenciam nos resultados apresentados no Capítulo 6 e servir como guia para estudos e análises mais profundos

em trabalhos futuros relacionados a este método. Por exemplo, temas como preferência por anexação ao termo mais próximo (citado em 4.4.6) e avaliação do método de suavização da estimativa de parâmetros (4.4.7) foram alvos de trabalhos específicos para coleta de estatísticas no Inglês. De qualquer forma, além da apresentação do aspecto e citação de possíveis trabalhos relacionados, na medida do possível serão apresentados alguns dados coletados neste trabalho para o Português.

4.4.1 Preferências Estruturais (Modificador à Direita ou à Esquerda)

No caso da língua inglesa, a ordem relativa das palavras numa dependência do tipo *item lexical modificador* para o *principal* é probabilisticamente tendenciosa, isto é, existem fortes preferências pela direção em que as dependências ocorrem. Por exemplo: um sujeito é quase sempre visto antes (à esquerda na seqüência de palavras de uma frase) do item lexical principal (verbo) que ele modifica. Um caso interessante está em [Charniak 96], onde se desenvolveu um analisador sintático GLCP no qual foi avaliado o efeito do uso de diferentes índices de correção para estímulo à ramificação à direita da árvore. As árvores do estudo em questão tinham a propriedade de criar ramos para a direita, como em *(The (cat (licked (several pans))))* apresentado na figura 4.1 abaixo.

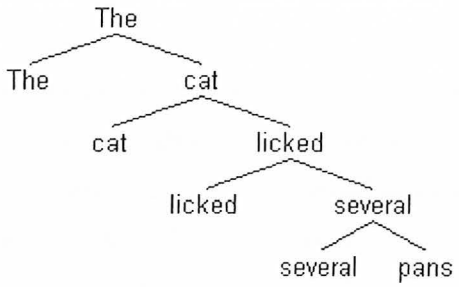


Figura 4.1: Exemplo de árvore sintática cujas derivações formam ramificações à direita [Collins 99]

Ao se experimentar diversos índices de correção que aumentavam as notas das derivações criadoras de ramificações à direita, percebeu-se uma diminuição de até 10,5% no erro da medida de cobertura. A justificativa é que este índice acrescenta a informação de preferência no sentido do relacionamento sintático entre as palavras, contexto que é perdido pela implementação padrão da GLCP.

Esta propriedade também é vista em muitos relacionamentos sintáticos presentes no Português. Nossos adjetivos normalmente são aplicados à direita do objeto modificado, como em 'menino bonito', 'mesa grande', 'bola murcha'. O conhecimento deste tipo de propriedade

pode ser aplicado nos métodos utilizados para análise sintática e portanto urge em se levantar dados correlatos a tais tendências.

No caso da GLCP desenvolvida neste trabalho para o Português, este efeito aparece em algumas sentenças, como a apresentada na forma 4.5. Futuramente, algumas estatísticas podem ser levantadas quanto a tal aspecto, conforme foi feito em [Charniak 96].

No caso do analisador sintático orientado ao núcleo-léxico desenvolvido neste trabalho (Capítulo 5), este tipo de informação é coletado na medida de distância entre os itens lexicais participantes da dependência. É o primeiro indicador da medida, que apresenta se o item lexical principal aparece depois ou não que o modificador na sentença. No córpus de treinamento, cerca de 60% dos modificadores estão depois do item lexical modificado (ou principal da dependência).

4.4.2 Propriedade da Localidade

Muitos dos trabalhos em lingüística para o Inglês focam a condição da localidade estrutural, isto é, insistem na localidade da influência dos núcleos-léxicos dentro da árvore sintática. Cada palavra da sentença afeta um domínio limitado dentro da árvore, conforme já argumentamos na seção 3.1 deste trabalho.

Este aspecto em particular é de extrema importância para o sucesso do método orientado ao núcleo-léxico, já que a própria escolha de decomposição da árvore assume que o item lexical principal deve ser gerado antes de toda a estrutura que depende dele (conforme citado em 2.9.4.1). Estudos específicos deste aspecto para o Português podem formar um importante arcabouço para novas decomposições e representações em futuras opções de analisadores sintáticos automáticos. Como referência de estudos deste aspecto efetuados para o Inglês existe [Joshi 87] que apresenta a localidade da influência do item lexical como uma lista de restrições associadas a este, e [Frank 92] no qual tais restrições são discutidas extensivamente.

4.4.3 O Nível de Barra do Constituinte

Uma palavra com uma determinada etiqueta morfossintática pode ser o núcleo de diversos tipos de constituintes, por exemplo, um verbo pode ser o núcleo de um constituinte *VB* de tamanho 1, ou de um *VP* completo, ou até mesmo de um *IP* (sentença inteira). Um nome de etiqueta morfossintática *N* pode ser o núcleo de um nome simples ou de um *NP* completo, como (*N*) 'homem' e (*NP*) 'o homem no parque'.

A variedade de não-terminais associados a cada par palavra/etiqueta morfossintática determina a complexidade da análise sintática automática para a língua em questão. Isso depende não só do estilo de anotação como da quantidade de etiquetas empregadas. Este tipo de análise é importante para avaliar os tipos de contextos relevantes para se distinguir os diferentes níveis de barra que são vistos nos diversos meios sintáticos presentes numa mesma árvore sintática. Um analisador sintático automático deve ser capaz de captar tais distinções de níveis para eliminar construções sintaticamente inválidas em sua busca. Coleta de estatísticas sobre o córpus a esse respeito também é útil como base de explicação para resultados obtidos neste e em outros trabalhos futuros para o Português.

4.4.4 Escolha da Forma de Lexicalização da Árvore Sintática

Conforme já apresentado na seção 4.3.2, a forma de lexicalização das árvores sintáticas anotadas influencia diretamente na qualidade dos resultados. A idéia de fazer os constituintes principais subirem na estrutura da árvore reflete na idéia capital da localidade da influência das palavras dentro de uma mesma sentença. A lexicalização mal feita acarreta na má definição destes domínios de localidade das palavras da sentença, o que gera dependências sintaticamente sem valor.

Trabalhos focados exclusivamente na tarefa de se definir as listas de prioridades para a lexicalização de cada nó podem então ser desenvolvidos objetivando a melhora dos resultados finais. A análise dos casos de erros também oferece bastante conteúdo para o aperfeiçoamento do processo de lexicalização. O caso extremo pode vir a se tornar um sistema especialista que leve em consideração não só a etiqueta do não-terminal pai, mas também outros contextos da árvore.

No caso do Inglês, tanto [Magerman 95] como [Collins 99] usaram listas de prioridades com a definição de duas exceções: regras especiais para os *NPs* e frases com termos coordenados (relação de coordenação entre dois ou mais nomes). Além disso, o percorrimento dos nós filhos do lado direito de uma derivação para procura da etiqueta principal pode ser tanto no sentido para direita como para a esquerda. Assim, além da lista de prioridade, cada etiqueta não-terminal tem definido seu sentido de percorrimento e de procura para seus nós filhos. No caso deste trabalho foram apenas utilizadas as listas de prioridades, montadas a partir da análise das anotações completas das árvores do córpus Tycho Brahe não lexicalizado. O sentido de percorrimento é fixo, sempre da esquerda para direita dos nós filhos, o que de certa forma simplifica mas enfraquece a exatidão da técnica. Tais listas poderão ser ordenadas de forma melhor com o devido embasamento lingüístico e o processo pode ser complementado com regras mais complexas para os casos de exceções não analisados aqui.

4.4.5 A Relevância das Subcategorias

Os métodos finais do trabalho de [Collins 99] são motivados por um tratamento matemático mais preciso em seus modelos e pelo o uso de conjuntos de subcategorias sintáticas, no qual se distingue cada constituinte como sendo complemento ou adjunto em sua própria etiqueta. Isso na verdade duplicou o número de etiquetas não-terminais existentes no córpus, através da inserção do sufixo *–C* a cada constituinte que fosse complemento sintático na sentença. Pode-se então treinar os conjuntos de subcategorias de complementos esperados à esquerda e à direita de cada tipo de constituinte. Desta forma, codificam-se informações sintáticas importantes como o fato de um verbo esperar apenas um sujeito à esquerda e não dois ou mais, ou por exemplo, verbos transitivos diretos esperarem apenas um complemento objeto à direita (normalmente). Assim, a estrutura sintaticamente incorreta da figura 4.2 seria descartada por ter uma subcategoria (NP-C, NP-C) (dois complementos nominais) à esquerda cuja nota de probabilidade de existência baseada no treinamento do córpus seria bastante baixa. Veja que, no entanto, todas as dependências presentes em tal estrutura são comuns e deveriam ter notas de probabilidades relativamente altas.

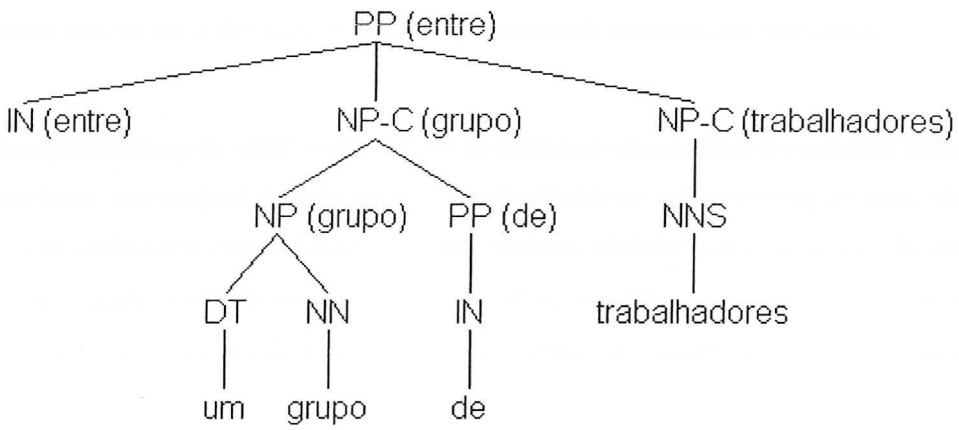


Figura 4.2: Esta árvore deveria ter uma probabilidade baixa devido ao conjunto de subcategorias improvável da direita de *IN(entre)*. Entretanto, as dependências presentes nesta estrutura são todas de boa probabilidade e portanto sua representação baseada em dependências sem informação de subcategorias falhará em reprovar tal estrutura [Collins 99].

O trabalho desenvolvido no Capítulo 5 orientado ao núcleo-léxico não faz uso de subcategorias devido à impossibilidade de se inserir naturalmente tal estrutura no mecanismo do algoritmo. Tais informações sintáticas de complementos são parcialmente captadas por aproximação na medida de distância utilizada e que está apresentada a seguir. A medida de distância coleta informações sobre o contexto da dependência existente e que podem ser úteis para captar parcialmente relações de complementos. Trabalhos com uso direto de subcategorias aplicados para o Português podem ser testados e comparados com o desenvolvido nesta dissertação.

4.4.6 Medida de Distância Como Aproximação das Subcategorias

O método orientado ao núcleo-léxico apresentado no Capítulo 5 utilizou uma tupla de 6 valores como medida de distância entre o modificador e o núcleo principal de cada dependência coletada. A definição desta medida foi aproveitada do trabalho desenvolvido por [Collins 99]. Os 6 valores concatenados formam uma palavra que representa a medida de distância presente na dependência e seus valores são respectivamente calculados da seguinte forma:

1°) O núcleo principal da dependência aparece depois do modificador na sentença?

Valores: 1 se sim, 0 caso contrário.

A língua Portuguesa tem forte ordenação entre as palavras de suas sentenças, assim como o Inglês. Assim, tal informação é relevante nas estatísticas das dependências existentes. Mas no caso do córpus Tycho Brahe, conforme foi apresentado na seção 4.2.1, os textos do século XVII possuem uma convenção gramatical mais livre quanto à ordem das palavras. Isso de certa forma enfraquece a relevância sintática desta informação na medida de distância. No treinamento executado sobre o córpus Tycho Brahe, em cerca de 65% das 37585 dependências coletadas, o núcleo-léxico principal (modificado) aparece antes do modificador, confirmando a preferência descrita em 4.4.1.

2°) São os dois termos (o modificador e o núcleo principal) da dependência adjacentes entre si?

Valores: 1 se sim, 0 caso contrário.

No Inglês, é relativamente comum as dependências ocorrerem entre itens lexicais adjacentes devido ao fato das estruturas serem de ramificações à direita. No treinamento executado sobre o esquema de anotação do córpus Tycho Brahe, cerca de 44% das 37585 dependências coletadas eram entre termos adjacentes.

3°) Existe um ou mais verbos entre os dois termos da dependência?

Valores: 1 se sim, 0 caso contrário.

Foi calculado para o córpus Tycho Brahe que 83% das dependências não têm nenhum verbo entre o modificador e o modificado. No Inglês foi reportada uma preferência ainda mais forte que esta, cerca de 94% das dependências não passavam por verbos.

4°) Existem 0, 1, 2, ou mais 'vírgulas' (significa qualquer tipo de pontuação) entre os dois termos da dependência?

Valores: 0, 1, 2, 3 para respectivamente nenhuma, uma, duas, e três ou mais 'vírgulas'.

5°) Existe uma 'vírgula' imediatamente após o primeiro dos dois termos da dependência?

Valores: 1 se sim, 0 caso contrário.

6°) Existe uma 'vírgula' imediatamente antes do segundo dos dois termos da dependência?

Valores: 1 se sim, 0 caso contrário.

As perguntas 4, 5 e 6 tentam codificar informações relevantes que a pontuação das sentenças podem passar a respeito da estrutura sintática existente. Como as pontuações são simplesmente retiradas na análise sintática do modelo orientado ao núcleo-léxico, sua influência se limita apenas à medida de distância. Portanto não é de se estranhar que metade do tamanho da medida de distância trate apenas do contexto da pontuação entre os termos da dependência. Esta codificação da informação de pontuação é idêntica à utilizada para o Inglês e isto pode ser um ponto aperfeiçoado para o modelo em Português, já que existem muitas diferenças nas regras de pontuação entre as duas línguas. Exceto a pergunta 4, todas as demais cinco perguntas têm respostas binárias (sim ou não) e o conjunto dos seis valores formam a medida de distância de cada dependência coletada.

Para exemplificar a utilidade da medida de distância, vamos usar a medida de tamanho 2, mais simples, definida na seção 3.2.1 e repetida aqui:

- o primeiro bit indica se a sentença superficial entre o modificador e o núcleo é vazia (valor igual a 1 se for, 0 caso contrário). Essa sentença superficial é composta pelas palavras geradas pelos símbolos não-terminais entre o símbolo não-terminal do modificador e o símbolo não-terminal do núcleo na aplicação da regra gramatical que associa os dois.
- o segundo indica se existe um verbo nesta sentença superficial definida acima. (1 se existir, 0 caso contrário).

No caso da figura 4.3 vemos como esta medida é útil. Nos dois casos as duas árvores candidatas diferenciam-se apenas por uma dependência e suas notas de probabilidades tendem a serem próximas entre si se analisarmos apenas o contexto da dependência sem a medida de distância. Mas a adição desta medida diferencia bem uma dependência da outra e permite a assimilar uma probabilidade mais elevada as dependências com anexação mais próximas.

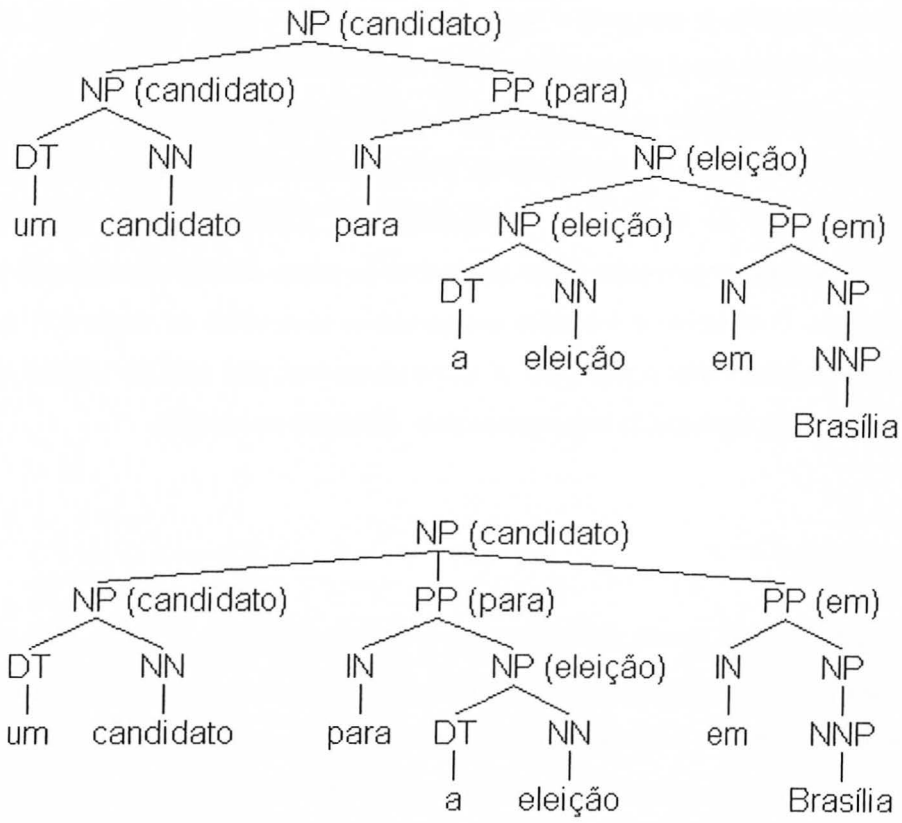


Figura 4.3: Duas árvores candidatas que se diferenciam apenas por uma dependência, *<em → eleição, dist: 10>* e *<em → candidato, dist: 00>*, respectivamente em cada árvore. Sem a medida de distância as duas dependências teriam notas de probabilidades próximas entre si e seria difícil distinguir. Com a medida de distância, o modelo pode aprender que existe uma preferência por anexação mais próxima, o que aumenta a nota da primeira candidata.

De fato, a medida de distância é apenas uma aproximação para a informação de subcategorias de complementos dos constituintes de uma árvore. Não é difícil encontrar exemplos em que tal medida falha em captar algumas informações que seriam habilmente codificadas com o uso de conjunto de subcategorias. De qualquer forma, o aprimoramento desta medida, com a inserção de novos dados de contexto e a modificação daqueles que não são tão úteis para o Português, pode ser pesquisado em detalhe nos analisadores sintáticos futuros.

4.4.7 Método de Suavização da Estimativa de Parâmetros

O Capítulo 5 explica o método de suavização de estimativas utilizado nos parâmetros coletados para o analisador sintático orientado ao núcleo-léxico desenvolvido para o Português. Basicamente o método foi idêntico ao utilizado para o Inglês por [Collins 99], mas provavelmente, pela diferença do tamanho de córpus utilizado, seria mais interessante usar outras formas de suavização. Particularmente a medida de distância nunca é enfraquecida no

contexto da suavização, o que para o caso do nosso córpus, cerca de 10 vezes menor que o utilizado para o Inglês em [Collins 99], poderia ser interessante. Outra suavização que poderia ser pensada seria o uso de (*Contagem(Regra unária| etiqueta CMS) / Contagem (etiqueta CMS)*) como mais um nível de suavização da estimativa da probabilidade das regras unárias no algoritmo de cálculo de estimativas, apresentado em 5.4.3.2, Capítulo 5. Estas e outras suavizações poderiam ser pensadas como alternativa ao nosso limitado domínio de treinamento para o Português. O objetivo é o melhor compromisso pretendido na seção 2.7: um contexto nem tão completo para evitar o problema de sub-treinamento, mas nem tão restrito para evitar a perda de informações contextuais importantes para distinção no modelo.

4.5 Conclusão

O córpus Tycho Brahe serviu de base para o desenvolvimento e testes dos analisadores sintáticos desenvolvidos neste trabalho e descritos no Capítulo 5. Como este córpus ainda está em fase de crescimento e anotação manual das estruturas sintáticas, o tamanho da base de treinamento foi relativamente pequeno em comparação com outros trabalhos desenvolvidos para o Inglês. No nosso caso, utilizamos 3715 sentenças para treinamento e 432 para testes, com um total de 37585 dependências coletadas na fase de treinamento. No caso do trabalho de [Collins 99], utilizou-se o *Penn Treebank Wall Street Journal Corpus* (PTWSJC), com 40.000 sentenças anotadas para treinamento, mais de 10 vezes maior que o nosso caso.

Estas condições exigiram uma redução forte no tamanho do problema, o que significou em termos práticos a redução da quantidade de tipos de etiquetas de 181 para 19 não-terminais, mais as 38 etiquetas morfossintáticas existentes. Tal redução foi efetuada através da remoção de todas as informações complementares, mantendo apenas o núcleo de cada etiqueta. Apesar de reduzir o tamanho do problema, esta medida também significou uma grande perda de informação e conhecimento sintático, cujo valor para distinção de estruturas sintáticas pode ser relevante.

De qualquer forma, o trabalho desenvolvido conseguiu resultados a primeira vista animadores para uma pesquisa mais profunda na área, conforme apresentado no Capítulo 6 e 7. Assim, espera-se ajudar o processo de anotação manual do córpus através da aplicação inicial dos analisadores sintáticos construídos neste trabalho que retornam uma estrutura de árvore inicial como sugestão para a anotação. Esta sugestão inicial pode então ser alterada, corrigida e ter suas etiquetas refinadas manualmente até estar de acordo com as regras de anotação do córpus. A conseqüência imediata seria um aumento na taxa de crescimento do córpus anotado, o que permitiria novos treinamentos e coletas de parâmetros mais exatos para os modelos implementados. Isso melhoraria a performance dos analisadores sintáticos automáticos, o que

resultaria em melhores sugestões de árvores iniciais para a anotação. Além disso, tarefas como o próprio refinamento, poderiam ser automáticas e no caso extremo, só restariam as tarefas manuais de verificar, revisar e aprovar as árvores geradas automaticamente.

Um córpus para o Português mais maduro possibilitará então o desenvolvimento de novos analisadores sintáticos e de outras ferramentas para o processamento lingüístico (extração de informação, reconhecimento de fala, por exemplo). A pesquisa por fenômenos lingüísticos para o Português receberia mais uma ferramenta de análise, como no caso atual em que se procura estabelecer como a mudança prosódica dos padrões rítmicos do Português levou a mudanças sintáticas, em particular na colocação dos pronomes clíticos.

Capítulo 5
Implementação

5.1 Visão Geral dos Analisadores Sintáticos Implementados: GLCP e o Orientado ao Núcleo-Léxico (ONL)[23]

Ambos os analisadores sintáticos implementados podem ser divididos em duas tarefas básicas de implementação: a primeira visa à coleta dos parâmetros estatísticos necessários para a execução da segunda tarefa, que é a execução da busca e geração da árvore sintática mais provável para cada sentença de entrada, com estatísticas de precisão de resultados. Para executar cada uma destas tarefas são necessários alguns módulos auxiliares que de uma forma geral são aproveitados para os dois métodos implementados. O sistema como um todo pode ser visto conforme a figura 5.1 abaixo.

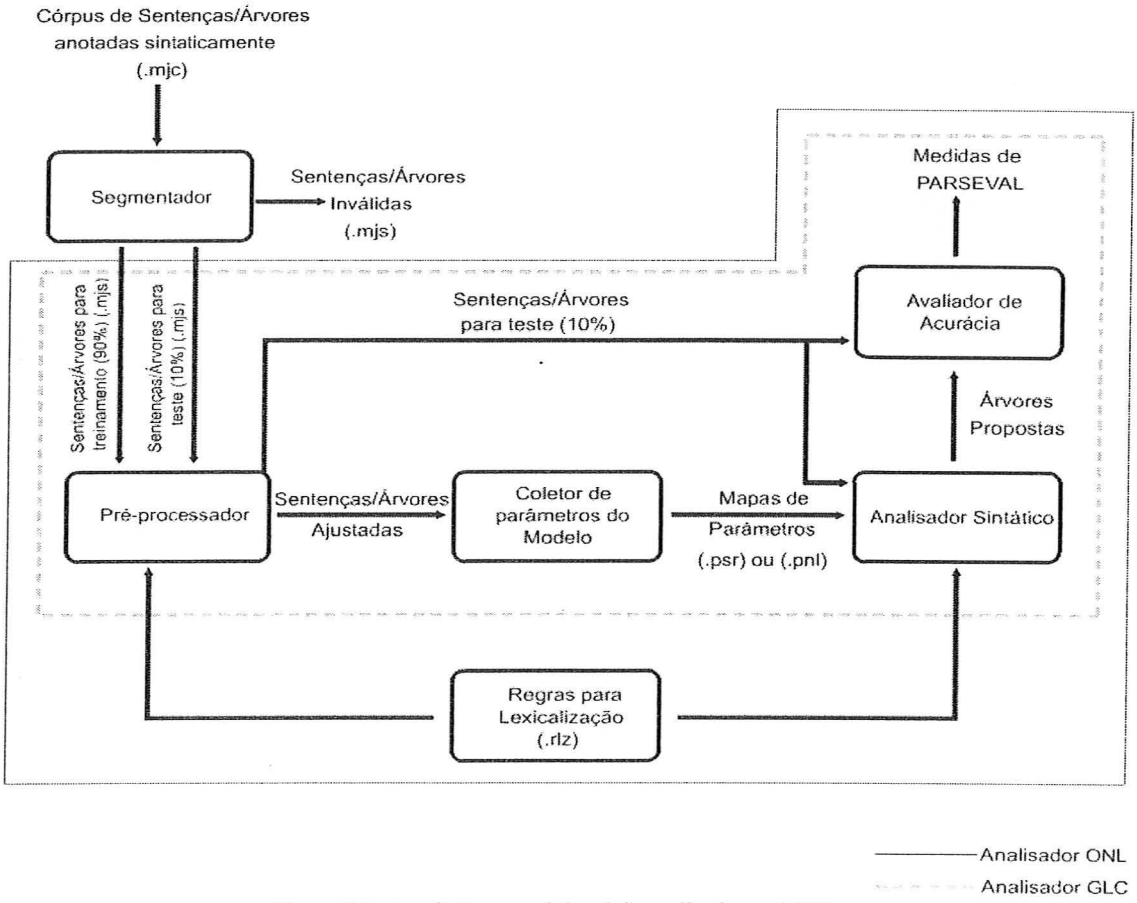


Figura 5.1: Arquitetura geral dos dois analisadores sintáticos

[23] Para abreviar o texto, deste capítulo em diante é comum o uso da sigla ONL para representar o método orientado ao núcleo-léxico baseado em dependências léxicas implementado neste trabalho.

Basicamente o que muda de um analisador sintático para o outro são os módulos de pré-processamento, o 'Coletor de Parâmetros do Modelo' e o módulo 'Analisador Sintático'. A seção 5.2 deste capítulo explica cada um dos módulos comuns à implementação dos dois modelos. A seção 5.3 trata das particularidades de implementação para o analisador GLCP e o 5.4 para o método orientado ao núcleo-léxico (ONL).

Todos os módulos descritos neste capítulo estão agregados em uma única aplicação de janela *SDI*[24] para *Windows9x/NT/2000®*. A chamada dos módulos pode ser efetuada separadamente por item de menu da aplicação. As entradas e saídas de cada módulo são selecionadas via escolha de arquivo, respectivamente para leitura e escrita. Todos os arquivos são em modo texto, para facilitar a interface com outros projetos futuros. A aplicação foi implementada em C++, utilizando as bibliotecas de classe *STL* e a *MFC* da *Microsoft®*. A biblioteca *MFC* é utilizada tanto para a codificação da interface gráfica da aplicação como em estruturas de dados básicas dos algoritmos implementados.

5.2 Os Módulos Auxiliares Comuns

5.2.1 O Segmentador Entre Grupo de Treinamento e Grupo de Testes

Este módulo é responsável por eliminar toda sentença/árvore sintática com erros de anotação e por montar os grupos de sentenças/árvores para treinamento e testes dos analisadores.

A entrada deste módulo é um arquivo texto de extensão .mjs contendo as anotação das árvores sintáticas do córpus linha a linha, conforme notação 4.4. A saída são três arquivos .mjs, subconjuntos sem interseção entre si, mas cuja união é igual ao conjunto das árvores de entrada. O primeiro arquivo contém as árvores que possuem algum tipo de erro de anotação, o segundo representa o conjunto de árvores para treinamento dos analisadores sintáticos e o terceiro representa o conjunto de teste para avaliação de performance deste analisador.

Mesmo tendo limitado o processamento ao grupo de sentenças revisadas pelos lingüistas, ainda existia uma minoria de sentenças/árvores que continham alguns erros de anotação, como etiquetas em minúsculas (fora do padrão) e parênteses faltando ou sobrando ou que tinham tamanho inferior a 5 itens lexicais. Tais sentenças foram então separadas e descartadas do uso no sistema. O que sobrou (cerca de 96%), foi então dividido em dois grupos (dois arquivos de texto): um de sentenças para treinamento e outro para teste. A proporção

[24] *Single Document Interface* – padrão de aplicação na qual só se exibe um documento por vez. Deste modo só se pode ter o resultado de um módulo por vez na aplicação, já que só existe uma janela de saída de processamento.

utilizada na literatura varia muito, mas como referência utilizamos cerca de 90% das sentenças para treinamento e o restante para testes, o que resultou respectivamente em 3715 e 432. Numa primeira tentativa, com o objetivo de diminuir o problema de sub-treinamento, selecionamos as 10% menores sentenças do córpus como de teste. Os resultados finais obtidos indicaram que isto podia ser visto como uma escolha tendenciosa a maquiar o desempenho dos sistemas, já que a maior sentença testada tinha apenas 9 palavras, o que é bastante raro para este tipo de tarefa.

Alteramos então o segmentador para retornar aleatoriamente 10% das sentenças como sendo de teste, restringindo estas ao tamanho mínimo de 5 e máximo de 30 palavras. Este tipo de restrição é relativamente comum principalmente entre os primeiros analisadores sintáticos desenvolvidos para o Inglês. Os resultados obtidos modificaram bastante, conforme comentado no Capítulo 6. Por fim, o córpus de treinamento conteve 58524 palavras e o de teste 8378 palavras. Vale notar que apesar do segmentador escolher aleatoriamente as sentenças, os arquivos de treinamento e teste utilizados foram exatamente os mesmos tanto para o analisador GLCP como para o orientado ao núcleo-léxico.

5.2.2 O Avaliador de Acurácia

O módulo para avaliação de acurácia executa a tarefa de comparar as duas árvores sintáticas, a anotada manualmente no córpus, considerada como a correta, e a retornada pelo analisador sintático automático. Esta comparação basicamente coletava as seguintes medidas:

- **Precisão não-etiquetada**: percentual de constituintes gerados pelo analisador que existem também na árvore correta. Para dois constituintes serem considerados iguais basta eles gerarem o mesmo conjunto de palavras, isto é, compreenderem o mesmo início e fim na seqüência de palavras da sentença, independente das etiquetas dos constituintes serem iguais ou não.
- **Cobertura não-etiquetada**: percentual de constituintes da árvore correta que existem também na árvore gerada pelo analisador. A comparação é a mesma da anterior, desconsiderando as etiquetas dos constituintes.
- **Precisão etiquetada**: percentual de constituintes gerados pelo analisador que existem também na árvore correta, mas que além de terem o mesmo início e fim na seqüência de palavras da sentença devem ter a mesma etiqueta associada ao constituinte.
- **Cobertura etiquetada**: percentual de constituintes da árvore correta que existem também na árvore gerada pelo analisador. A comparação é a mesma da anterior, isto é, considerando as etiquetas dos constituintes.

- **Percentual de Colchetes Atravessantes**: percentual de constituintes produzidos que não estão compatíveis com a estrutura da árvore correta, isto é, os colchetes do constituinte na representação do estilo 2.11 que possuem parte incompleta de colchetes da árvore correta.

Para se calcular estas medidas, o módulo precisa apenas da lista de colchetes de cada árvore, que é uma lista contendo os índices na sentença da palavra de início, da de fim e da etiqueta associada a cada constituinte presente. Os elementos de uma lista são comparados um a um com os da outra lista, de acordo com o tipo de medida que está sendo avaliada. Este tipo de entrada pode ser obtida para os dois analisadores sintáticos implementados. Portanto, este módulo foi aproveitado para análise dos resultados dos dois sistemas. Existe a vantagem também de se poder comparar a maior subestrutura encontrada no processamento de uma sentença, caso não se tenha encontrado uma árvore completa que gere todas as palavras da frase. Isso possibilitou a coleta de dados de precisão para os casos em que nenhuma árvore completa foi gerada, fato que ocorreu com certa freqüência no analisador orientado ao núcleo-léxico conforme apresentado no Capítulo 6. Este tipo de falha é captado principalmente pela medida de cobertura, que fica bastante prejudicada pela falta de constituintes gerados.

O módulo retorna as cinco medidas para o par de listas de colchetes fornecido e também os acumula em contadores para fornecer as cinco medidas para todo o conjunto de árvores testadas, ou seja, o resultado final e completo para o córpus.

5.2.3 Visualizador de Árvores Sintáticas

Apesar de não ser utilizado diretamente em nenhuma das tarefas presentes nos dois analisadores sintáticos, o desenvolvimento do projeto tornou interessante a implementação de um módulo que desenha graficamente uma árvore sintática expressa da forma 4.4 na mais fácil de se entender 4.5. Isto facilitou a tarefa de depuração do código ajudando a entender as estruturas que resultavam em falhas na aplicação e também ajudou na escrita da dissertação (todas as figuras de representação gráfica de árvores sintáticas presentes neste trabalho foram inicialmente traçadas por este módulo).

5.3 O Analisador Sintático GLCP

5.3.1 O Pré-Processamento

O analisador sintático GLCP desconsidera totalmente os itens lexicais para a formação da árvore sintática. De fato, como a sentença de entrada já está com as respectivas categorias

morfossintáticas para cada item lexical, a palavra em si não está presente em nenhuma derivação gerada de regra gramatical da árvore. Pode-se dizer que as regras gramaticais aplicadas na parte inferior da árvore sintática gerada são todas da forma etiqueta *não-terminal* → *categoria morfossintática (*ou meramente ***cms)***. As regras posteriores do tipo *categoria morfossintática* → *item lexical* vão estar presentes em todas as árvores candidatas, e portanto não diferenciará em nada uma árvore candidata de outra. Assim, tais regras podem ser descartadas sem prejudicar a escolha do algoritmo.

Em poucas palavras, significa que o algoritmo GLCP faz a geração da árvore sintática a partir apenas da seqüência das categorias morfossintáticas presente na sentença. Assim, duas sentenças com palavras diferentes terão exatamente a mesma árvore sintática gerada pelo GLCP caso tenham a mesma seqüência de categorias morfossintáticas entre si.

Deste modo, o pré-processamento é responsável por eliminar as folhas de item lexical das árvores analisadas do córpus, conforme figura 5.2 abaixo, deixando como folhas das árvores apenas as categorias morfossintáticas.

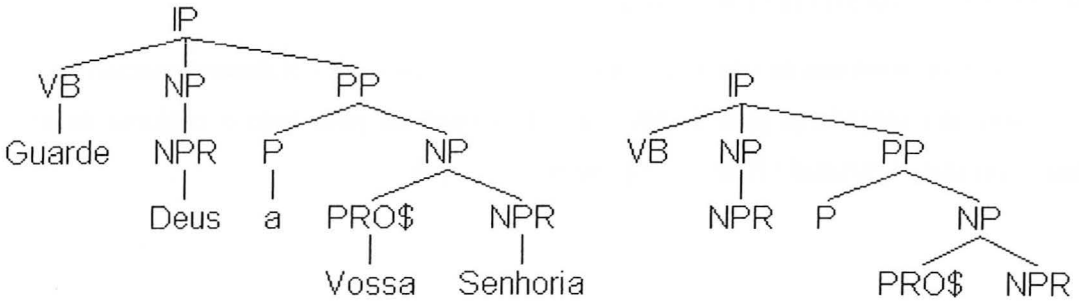


Figura 5.2: Exemplo de entrada e saída do pré-processador aplicado no sistema GLCP

Este tipo de pré-processamento é aplicado tanto na fase de treinamento (coleta de parâmetros) como na de testes da GLCP. Assim, se reduz a quantidade de parâmetros coletados pelo modelo para o conjunto realmente útil, o que também diminui o custo de aplicação e busca do algoritmo.

5.3.2 O Coletor de Parâmetros

O Coletor de Parâmetros é responsável por coletar as probabilidades de cada uma das regras de derivação presentes nas árvores do córpus de treinamento. A entrada é o arquivo de treinamento de árvores sintáticas anotadas e revisadas (arquivo .mjs) e a saída é um arquivo texto de extensão .psr onde cada linha tem o seguinte formato:

```
#Quantidade de ocorrências da regra <TAB> Probabilidade da regra <TAB>
Regra gramatical <FIM DE LINHA>
```

Um exemplo de parâmetro coletado é o seguinte:

```
46      0.007748      PP: ADV P NP                    (5.1)
```

Que significa que a regra gramatical $PP \rightarrow ADV\ P\ NP$ foi vista 46 vezes no córpus de treinamento e a probabilidade de um não-terminal *PP* derivar esta regra é de 0,7748%. O cálculo da probabilidade de uma regra r é feito conforme descrito abaixo (onde $LEFT(r)$ significa o lado esquerdo da regra, ou nó pai da derivação):

$$P(r) = \frac{Contagem(r)}{\sum_{r' \in \{r' | LEFT(r') = LEFT(r)\}} Contagem(r')} \qquad (5.2)$$

A coleta demorou menos de 1 minuto para o córpus de treinamento com 58524 palavras num Pentium III 866Mhz. Foram coletados 3701 tipos de regras gramaticais diferentes num total de 45195 instâncias. Conforme descrito na seção anterior, não foram coletados parâmetros de regras finais do tipo *categoria morfossintática* → *item lexical*. Maiores informações sobre o treinamento obtido estão presentes no Capítulo 6 de análise de resultados.

5.3.3 O Analisador Sintático

A implementação do analisador sintático foi de acordo com o padrão do algoritmo de Carta de Análise resumido na seção 2.10 e bem descrito em [Charniak 93], inclusive em suas estruturas de dados.

Basicamente o algoritmo recebe a seqüência de categorias morfossintáticas da sentença, monta a carta de análise e retorna uma lista ordenada por probabilidade decrescente de árvores geradas com as aplicações das regras gramaticais vistas no treinamento. A entrada é basicamente o arquivo do córpus de teste (anotado sintaticamente, extensão .mjs) e o arquivo de parâmetros coletados do treinamento (extensão .psr, contendo as probabilidades de todas as regras gramaticais vistas). Da lista ordenada de saída é retirada apenas a primeira árvore, a que tem maior probabilidade dentre as que foram geradas, e esta é passada automaticamente para o módulo Avaliador de Acurácia (descrito em 5.2.2). Este apresenta o resultado para cada par árvore correta/árvore gerada e por fim, o resultado geral para o córpus de teste.

O algoritmo tem a opção de limitar a quantidade de estruturas que são geradas por cada entrada na carta. Assim, quando uma nova entrada é gerada por um arco de regra gramatical completado, deveria-se teoricamente associar a esta entrada todas as estruturas possíveis geradas pela permutação entre os conjuntos de subestruturas de cada um dos filhos da entrada. Imagine,

por exemplo, que uma entrada da carta de análise foi gerada por se ter completado o seguinte arco $PP \rightarrow ADV\ P\ NP$. A entrada tem então etiqueta *PP* e se, por exemplo, a entrada filha *ADV* tem 4 estruturas associadas, a *P* tem 2 e a *NP* tem 5, o total de estruturas geradas para *PP* seria 40 (= 4x2x5). Isso faz a atualização de estruturas ser extremamente custosa à medida que tais entradas vão se associando e a quantidade de permutações cada vez mais aumentando. Ao se limitar a quantidade de estruturas geradas, diminui-se tal problema sem acrescentar erros ao algoritmo. Este parâmetro, no caso dos nossos testes, ficou fixo em 20 árvores por entrada criada. O algoritmo, neste caso, toma o cuidado de manter as 20 estruturas de maior nota e apagar as demais.

Foi adicionada também uma lista de regras unárias consecutivas. Basicamente esta lista foi utilizada para evitar ciclos entre entradas na carta de análise de mesmo tamanho, início e fim, mas com etiquetas diferentes. Estes ciclos surgem por causa de regras unárias coletadas no córpus de treinamento. Por exemplo, a seguinte seqüência forma um ciclo: $NP \rightarrow CP \rightarrow IP \rightarrow NP \rightarrow \ldots$. Veja que alguns processos da carta de análise fazem varredura entre os nós das estruturas geradas até o momento, varrendo todas as entradas geradoras de uma entrada. Estes ciclos fazem o algoritmo entrar em iteração de varredura infinita, o que trava o andamento do processo. O uso da lista limitou as regras unárias em dois aspectos:

1. Uma seqüência de regras unárias consecutivas presente na carta de análise nunca pode ter uma mesma entrada repetida duas vezes. Em outras palavras, as etiquetas de uma seqüência de regras unárias consecutivas são todas diferentes entre si. Assim, é impossível se ter seqüências do tipo $NP \rightarrow IP \rightarrow NP$, por exemplo, pois *NP* não pode voltar ao mesmo *NP*.
2. O tamanho máximo da seqüência de regras unárias consecutivas é três. Assim não se pode ter seqüências do tipo $NP \rightarrow CP \rightarrow IP \rightarrow DP$. Este limite foi verificado experimentalmente no córpus de treinamento utilizado, nenhuma seqüência de regras unárias consecutivas nas árvores de treinamento tinha 4 ou mais regras gramaticais. Esta restrição diminuiu a geração de estruturas pouco prováveis.

De resto, o algoritmo utilizou as técnicas descritas no método de carta de análise padrão: uma pilha chamada agenda, ordenada por nota de probabilidade decrescente, uma estrutura bidimensional chamada carta de análise e listas de arcos de derivação incompletos. Existe uma estrutura auxiliar para permitir a recuperação das árvores associadas a um constituinte raiz através de listas de entradas geradoras ordenadas por probabilidade a cada entrada da carta. Outra lista indica para cada entrada da carta quais são as entradas que ele ajudou a gerar na carta de análise. Assim, quando ocorre alguma atualização da probabilidade

máxima do constituinte pode se atualizar as probabilidades dos constituintes gerados de forma rápida.

O algoritmo só retorna árvores encontradas que gerem a sentença inteira. Caso não seja encontrada nenhuma árvore, o retorno é uma lista vazia. Não foi implementado nenhum método para retornar a maior subestrutura encontrada ou algo parecido. Na prática isto também não foi necessário, pois como será visto no Capítulo 6, todas as sentenças de entradas foram processadas com sucesso, isto é, tiveram pelo menos uma árvore completa gerada.

O tempo de execução do analisador sintático num Pentium III 866Mhz (1GByte de memória RAM) para 432 sentenças com 8378 palavras ao todo foi de 13 horas. Este tempo excessivo se justifica pelo fato de não ter sido implementado nenhuma técnica de limiar de corte para reduzir o número de constituintes gerados e também por ter-se usado muitas estruturas de listas, que poderiam ser modificadas para mapas (*hash-tables*), o que aceleraria o tempo de busca de algumas operações iterativas.

5.4 O Analisador Sintático Orientado ao Núcleo-Léxico

5.4.1 O Pré-Processamento

O analisador sintático orientado ao núcleo-léxico descreve a árvore sintática através das dependências entre itens lexicais presentes nela, conforme visto em 3.2. Além disso, o modelo implementado neste trabalho reduz a sentença de entrada a seus *NPs* básicos para melhorar o valor sintático das medidas de distância presentes em cada dependência. Assim, o pré-processamento é responsável por três tarefas distintas na etapa de treinamento:

1. Lexicalizar a árvore de entrada, conforme descrito em 4.3.2.
2. Reduzir os *NPs* básicos para apenas os arcos do núcleo principal de seu *NP*. Além de reduzir a árvore sintática, o pré-processador armazena dados que permitam a coleta de probabilidades das etiquetas G_is nas lacunas entre palavras da sentença, denotada por $P(G_i \mid p_{i-1}, t_{i-1}, p_i, t_i, c_i)$, conforme a equação 3.5 apresentada no capitulo 3.
3. Retirar todas as pontuações presentes na sentença, atualizando uma estrutura auxiliar que é utilizada para o cálculo da medida de distância das dependências e de parte do contexto para os parâmetros do modelo *NPs* básicos.

A figura 5.3 exemplifica uma árvore de entrada para o pré-processador e a sua respectiva saída.

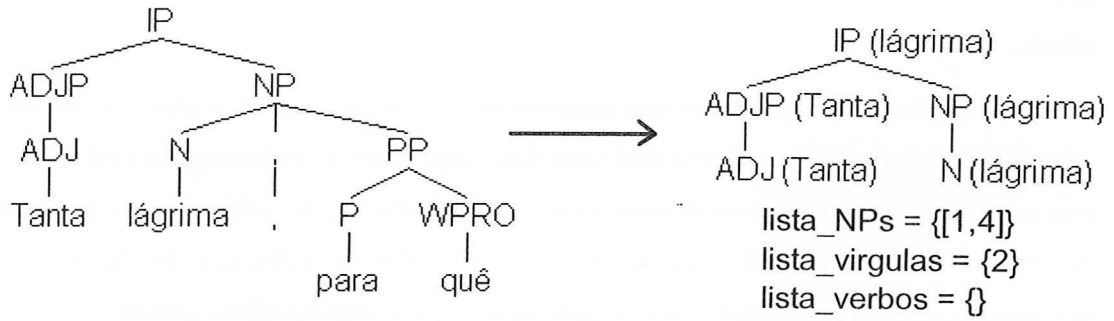


Figura 5.3: Exemplo de entrada e saída do pré-processador aplicado no sistema ONL

Perceba que a árvore de saída e as estruturas de listas retornadas serão utilizadas pelo Coletor de Parâmetros tanto para estimar as probabilidades do modelo de *NPs* básicos como o de dependências, conforme apresentado pela fórmula 3.1 repetida abaixo:

$$\mathcal{P}(A|S) = \mathcal{P}(B, D|S) = \mathcal{P}(B|S) \times \mathcal{P}(D|S, B) \tag{5.3}$$

Onde B é o conjunto de *NPs* básicos e D é o conjunto de dependências presente na árvore A que gera a sentença S.

Na fase de testes, o pré-processamento só realiza a tarefa da lexicalização das árvores e remoção da pontuação, já que a seleção dos *NPs* básicos da sentença analisada é feita automaticamente pelo analisador sintático descrito na seção 5.4.3 . O processo de lexicalização usa um arquivo externo de extensão .rlz com as definições das listas de prioridades para cada etiqueta não-terminal presente no córpus, conforme explicado em 4.3.2 .

5.4.2 O Coletor de Parâmetros

O Coletor de Parâmetros para o método orientado ao núcleo-léxico foi responsável por fazer as contagens de eventos do córpus de treinamento para os 3 processos distintos do analisador sintático descrito em 5.4.3:

1. Estimativa das probabilidades das etiquetas das lacunas de definição dos *NPs* básicos
2. Estimativa das probabilidades das dependências léxicas
3. Estimativa da geração de regras unárias, segundo um critério ad-hoc, já que tais eventos não se encaixam diretamente no modelo de dependências por não criarem dependência alguma.

No total são coletados 18 mapas de probabilidades a partir do córpus anotado de treinamento. Cada mapa é uma *hash-table*, uma estrutura de dados otimizada para procura, onde se associa a cada evento presente no mapa um número inteiro que indica sua quantidade de ocorrências verificada no córpus de treinamento. O arquivo de entrada tem extensão .mjs e o de saída tem extensão .pnl, onde cada linha tem o seguinte formato:

```
#tipo do mapa <TAB> Evento <TAB> Contagem de ocorrência do
                    evento <FIM DA LINHA>                    (5.4)
```

Um exemplo de parâmetro gravado no arquivo é o seguinte:

```
0      Reconhecendo|VB|Margarida|NPR|0      1              (5.5)
```

significa que o evento do tipo palavra *'Reconhecendo'* de categoria morfossintática *VB* antes da palavra *'Margarida'* de categoria morfossintática *NPR,* sem vírgula ou qualquer outra pontuação entre eles (indicado pelo zero no fim do evento), ocorreu uma única vez e que pertence ao mapa de índice 0, o primeiro dos 18 mapas, que está descrito na seção a seguir.

Os mapas armazenam apenas contagem de eventos a partir dos quais serão calculadas dinamicamente as estimativas de probabilidade úteis para o analisador sintático em 5.4.3. Esta forma de implementação poupa esforço, na medida em que só se calcula uma estimativa de probabilidade quando esta é necessária. Calcular antecipadamente tais estimativas seria um absurdo dado o tamanho enorme do domínio de possibilidades de parâmetros existentes (no pior caso, todas as possíveis permutações de duas palavras/categoria morfossintáticas presentes no córpus de treinamento, do qual a maioria nunca seria vista no córpus de teste).

O tempo de coleta dos parâmetros foi de 42 minutos em um Pentium III 866Mhz para o córpus de treinamento de 3715 sentenças contendo ao todo 58524 palavras.

A seguir são apresentados cada um dos mapas coletados para os processos presentes no analisador sintático.

5.4.2.1 Mapas de Estimativas das Probabilidades de Etiquetas de Definição de *NPs* Básicos

A estimativa das probabilidades para as etiquetas das lacunas de definição dos *NPs* básicos é feita com a utilização de 8 mapas de parâmetros coletados do córpus de treinamento.

O modelo consiste em etiquetar $G_i \in \{\mathbf{S}, \mathbf{C}, \mathbf{E}, \mathbf{B}, \mathbf{N}\}$[25] em cada lacuna entre palavras da sentença, considerando-se para isso apenas a palavra à esquerda e à direita de cada uma das

[25] O significado de cada etiqueta e o entendimento da idéia geral da etiquetagem pode ser visto em 3.2.1 .

lacunas e se existe ou não uma vírgula entre as duas palavras ($c_i = 1$ se existe uma 'vírgula', ou $c_i = 0$ caso contrário). Usa-se a sentença não reduzida para isso, desconsiderando-se apenas as lacunas entre pontuação.

A estimativa necessária a se fazer é da forma $P(G_i \mid p_{i-1}, t_{i-1}, p_i, t_i, c_i)$, onde $G_i \in$ {**S**, **C**, **E**, **B**, **N**}, p_i é a i-ésima palavra da sentença não reduzida, t_i é a etiqueta de *CMS* da i-ésima palavra da sentença não reduzida, e c_i é da forma explicada anteriormente. A estimativa passa por 3 estágios de interpolação, ou ajustagem de parâmetros, a fim de suavizar as medidas e reduzir o problema de sub-treinamento. O contexto completo tem um domínio grande, com eventos esparsos para os dados de treinamento utilizados, o que acarretaria em muitos eventos na fase de teste não vistos e então não estimados. Para se fazer esta suavização das estimativas são coletados 8 mapas, conforme a seguir:

MAPA[0]: Contagem de Evento $w_{i-1}, t_{i-1}, w_i, t_i, c_i$ no córpus.

SINTAXE DO EVENTO: $w_{i-1} \mid t_{i-1} \mid w_i \mid t_i \mid c_i$

EXEMPLO: `0 <TAB> loucura|N|própria|ADJ|0 <TAB> 1`

MAPA[1]: Contagem de Evento $w_{i-1}, t_{i-1}, t_i, c_i$ no córpus.

SINTAXE DO EVENTO: $w_{i-1} \mid t_{i-1} \mid t_i \mid c_i$

EXEMPLO: `1 <TAB> loucura|N|ADJ|0 <TAB> 2`

MAPA[2]: Contagem de Evento t_{i-1}, w_i, t_i, c_i no córpus.

SINTAXE DO EVENTO: $t_{i-1} \mid w_i \mid t_i \mid c_i$

EXEMPLO: `2 <TAB> N|própria|ADJ|0 <TAB> 3`

MAPA[3]: Contagem de Evento t_{i-1}, t_i, c_i no córpus.

SINTAXE DO EVENTO: $t_{i-1} \mid t_i \mid c_i$

EXEMPLO: `3 <TAB> N|ADJ|0 <TAB> 475`

MAPA[4]: Contagem de Evento $G_i, w_{i-1}, t_{i-1}, w_i, t_i, c_i$ no córpus.

SINTAXE DO EVENTO: $G_i \mid w_{i-1} \mid t_{i-1} \mid w_i \mid t_i \mid c_i$

EXEMPLO: `4 <TAB> N|loucura|N|própria|ADJ|0 <TAB> 1`

MAPA[5]: Contagem de Evento $G_i, w_{i-1}, t_{i-1}, t_i, c_i$ no córpus.

SINTAXE DO EVENTO: $G_i \mid w_{i-1} \mid t_{i-1} \mid t_i \mid c_i$

EXEMPLO: `5 <TAB> N|loucura|N|ADJ|0 <TAB> 1`

MAPA[6]: Contagem de Evento $G_i \mid t_{i-1}, w_i, t_i, c_i$ no córpus.

SINTAXE DO EVENTO: $G_i \mid t_{i-1} \mid w_i \mid t_i \mid c_i$

EXEMPLO: `6 <TAB> N|N|própria|ADJ|0 <TAB> 2`

MAPA[7]: Contagem de Evento G_i, t_{i-1}, t_i, c_i no córpus.

SINTAXE DO EVENTO: $G_i \mid t_{i-1} \mid t_i \mid c_i$

EXEMPLO: `7 <TAB> N|N|ADJ|0 <TAB> 157`

5.4.2.2 Mapas para a Estimativa das Probabilidades das Dependências Léxicas

A estimativa das probabilidades das dependências léxicas é feita com a utilização de 8 mapas de parâmetros coletados do córpus de treinamento.

O modelo consiste em avaliar uma probabilidade para cada uma das *i-1* dependências presentes numa sentença reduzida a *NPs* básicos sem pontuação com *i* itens lexicais. Uma dependência é definida pelo item lexical modificador, o item lexical modificado ou principal da dependência e pela relação existente entre eles, conforme apresentado em 3.2.1. A relação é dada pelo conjunto dos três não-terminais envolvidos na dependência, o não-terminal do modificador, o não-terminal do pai gerado pela dependência e por fim o não-terminal do principal da dependência. Exemplos podem ser vistos em 3.2.1 também.

Para a estimativa da probabilidade da dependência também é levada em consideração a medida de distância existente entre os dois termos (modificado e o modificador) na sentença reduzida, denotada por $\Delta_{i,hi}$. A medida de distância é calculada pela forma exposta em 4.4.6.

Da mesma maneira que as estimativas do modelo de *NPs* básicos, existem 3 estágios análogos de interpolação, ou ajustagem de parâmetros, a fim de suavizar as medidas e reduzir o problema de sub-treinamento. O contexto completo tem um domínio grande, o que levaria a eventos não avaliados durante a fase de teste, isto é, sem estimativa calculada. Para se fazer esta suavização são coletados os 8 mapas a seguir:

MAPA[8]: Contagem de Evento $w_i, t_i, w_{hi}, t_{hi}, \Delta_{i,hi}$ no córpus.

SINTAXE DO EVENTO: $w_i \mid t_i \mid w_{hi} \mid t_{hi} \mid \Delta_{i,hi}$

EXEMPLO: `8 <TAB> primeira|ADJ|a|D|010000 <TAB> 2`

Onde w_i é o i-ésimo item lexical da sentença, t_i é a respectiva etiqueta morfossintática, w_{hi} é o núcleo-léxico principal que é alterado pela dependência vinda de w_i e t_{hi} é sua respectiva etiqueta morfossintática. $\Delta_{i,hi}$ é a medida de distância existente entre w_i e w_{hi}, segundo a definição em 4.4.6.

MAPA[9]: Contagem de Evento $w_i, t_i, t_{hi}, \Delta_{i,hi}$ no córpus.

SINTAXE DO EVENTO: $w_i \mid t_i \mid t_{hi} \mid \Delta_{i,hi}$

EXEMPLO: `9 <TAB> primeira|ADJ|D|010000 <TAB> 5`

MAPA[10]: Contagem de Evento $t_i, w_{hi}, t_{hi}, \Delta_{i,hi}$ no córpus.

SINTAXE DO EVENTO: $t_i \mid w_{hi} \mid t_{hi} \mid \Delta_{i,hi}$

EXEMPLO: 10 <TAB> ADJ|a|D|010000 <TAB> 31

MAPA[11]: Contagem de Evento $t_i, t_{hi}, \Delta_{i,hi}$ no córpus.

SINTAXE DO EVENTO: $t_i \mid t_{hi} \mid \Delta_{i,hi}$

EXEMPLO: `11 <TAB> ADJ|D|010000 <TAB> 138`

MAPA[12]: Contagem de Evento $R_i, w_i, t_i, w_{hi}, t_{hi}, \Delta_{i,hi}$ no córpus.

SINTAXE DO EVENTO: $R_i \mid w_i \mid t_i \mid w_{hi} \mid t_{hi} \mid \Delta_{i,hi}$

EXEMPLO: `12 <TAB> ADJ_NP_D|primeira|ADJ|a|D|010000 <TAB> 4`

Onde R_i é a relação existente entre w_i e w_{hi}, formada pela seqüência do não-terminal do modificador, o não-terminal do pai gerado pela dependência e por fim o não-terminal do principal da dependência.

MAPA[13]: Contagem de Evento $R_i, w_i, t_i, t_{hi}, \Delta_{i,hi}$ no córpus.

SINTAXE DO EVENTO: $R_i \mid w_i \mid t_i \mid t_{hi} \mid \Delta_{i,hi}$

EXEMPLO: `13 <TAB> ADJ_NP_D|primeira|ADJ|D|010000 <TAB> 5`

MAPA[14]: Contagem de Evento $R_i, t_i, w_{hi}, t_{hi}, \Delta_{i,hi}$ no córpus.

SINTAXE DO EVENTO: $R_i \mid t_i \mid w_{hi} \mid t_{hi} \mid \Delta_{i,hi}$

EXEMPLO: `14 <TAB> ADJ_NP_D|ADJ|a|D|010000 <TAB> 3`

MAPA[15]: Contagem de Evento $R_i, t_i, t_{hi}, \Delta_{i,hi}$ no córpus.

SINTAXE DO EVENTO: $R_i \mid t_i \mid t_{hi} \mid \Delta_{i,hi}$

EXEMPLO: `15 <TAB> ADJ_NP_D|ADJ|D|010000 <TAB> 130`

5.4.2.3 Mapas para a Estimativa da Geração de Regras Unárias

A geração de regras unárias é vista como um processo à parte do modelo orientado ao núcleo-léxico, e a definição de sua estimativa apresentada em 5.4.2.3 é matematicamente fraco, já que não estava previsto no modelo de dependências léxicas. As regras unárias são as únicas estruturas que não são captadas diretamente no modelo de dependências. Pela forma como foi definida, sua estimativa exige os seguintes dois mapas:

MAPA[16]: Contagem de Eventos de Regra unária aplicada sobre Palavra/CMS no córpus

SINTAXE DO EVENTO: *epai_efilho* | w_i | t_i , onde o não-terminal *epai* deriva a etiqueta *efilho* a partir da regra unária *epai* → *efilho*.

EXEMPLO: `16 <TAB> NP_N|necessidade|N <TAB> 3`

Este mapa conta a quantidade de ocorrências que a regra unária *epai* → *efilho* é vista aplicada para o par w_i / t_i no córpus.

MAPA[17]: Contagem de Eventos Palavra/CMS no córpus

SINTAXE DO EVENTO: *palavra|CMS*

EXEMPLO: `17 <TAB> necessidade|N <TAB> 6`

Este mapa conta a quantidade de ocorrências de cada par *palavra/CMS* vista no córpus.

5.4.3 O Analisador Sintático

O algoritmo do analisador sintático processa as entradas da carta de análise de baixo para cima (ascendente) e da esquerda para a direita. Não existe gramática apesar de na prática toda dependência com relação de três não-terminais não vista no treinamento terá probabilidade zero.

Ou seja, o algoritmo varre a carta de análise percorrendo o espaço de todas as árvores cujos trios de não-terminais foram vistos nos dados de treinamento. As probabilidades dos *NPs* básicos são calculadas à medida que são geradas novas entradas na carta de análise. Se dois constituintes geram o mesmo conjunto de palavras, têm a mesma etiqueta, mesmo núcleo-léxico e mesmas medidas de distância do núcleo para o extremo da esquerda e para o da direita do constituinte, então o de menor probabilidade pode ser descartado sem prejudicar a precisão do algoritmo. Veja na figura 3.2 do Capítulo 3 como dois constituintes na carta de análise combinam para formar um novo constituinte.

O algoritmo implementado varre todo o espaço de árvores possíveis e termina quando a agenda de processamento fica vazia. Não foi necessário implementar nenhuma das técnicas utilizadas por [Collins 99] para otimizar a performance de tempo de execução do algoritmo, como a estratégia de procura por raio nem limiares de descarte variáveis. Estimativas calculadas no decorrer do processo não são armazenadas para uso futuro, por se considerar pouco provável a repetição da mesma estimativa dentro do córpus de teste. Uma estrutura auxiliar é utilizada para indicar quais pares de entradas da carta de análise já foram processadas entre si e evitar repetição de processamento.

As regras unárias são acrescentadas nas entradas da carta de análise, mas as mesmas restrições implementadas no analisador *GLCP* foram utilizadas neste analisador. Basicamente evita-se ciclos entre entradas na carta de análise de mesmo tamanho, início e fim. Além disso, o tamanho máximo da seqüência de regras unárias consecutivas é três. Esta restrição diminuiu a geração de estruturas pouco prováveis.

Caso não seja encontrada nenhuma árvore, retorna-se a estrutura de árvore de maior probabilidade e tamanho em quantidade de palavras geradas da sentença de entrada. Na prática isto foi necessário, pois como será visto no Capítulo 6, um percentual considerável das sentenças de entradas não foram completamente processadas com sucesso. Por outro lado, quando se encontra pelo menos uma árvore completa que gere a sentença inteira, retorna-se a de maior nota de probabilidade.

O tempo de execução do analisador sintático num Pentium III 866Mhz para 432 sentenças com 8378 palavras ao todo foi de 42 minutos.

5.4.3.1 Estimativa das Probabilidades das Etiquetas de Definição do *NPs* Básicos

O cálculo da estimativa das probabilidades das etiquetas de definição dos *NPs* básicos geradas durante o processo de geração da árvore pelo analisador sintático segue o seguinte algoritmo de suavização de parâmetros:

$$\text{Dado: } E_1 = \frac{\eta_1}{\delta_1} \qquad E_2 = \frac{\eta_2}{\delta_2} \qquad E_3 = \frac{\eta_3}{\delta_3} \qquad E_4 = \frac{\eta_4}{\delta_4} \qquad E_{23} = \frac{\eta_2 + \eta_3}{\delta_2 + \delta_3} \tag{5.6}$$

Onde:

$$\delta_1 = Contagem(w_{i-1}, t_{i-1}, w_i, t_i, c_i) \text{ , obtida pelo MAPA[0]} \tag{5.7}$$

$$\delta_2 = Contagem(w_{i-1}, t_{i-1}, t_i, c_i) \text{ , obtida pelo MAPA[1]} \tag{5.8}$$

$$\delta_3 = Contagem(t_{i-1}, w_i, t_i, c_i) \text{ , obtida pelo MAPA[2]} \tag{5.9}$$

$$\delta_4 = Contagem(t_{i-1}, t_i, c_i) \text{ , obtida pelo MAPA[3]} \tag{5.10}$$

$$\eta_1 = Contagem(G_i, w_{i-1}, t_{i-1}, w_i, t_i, c_i) \text{ , obtida pelo MAPA[4]} \tag{5.11}$$

$$\eta_2 = Contagem(G_i, w_{i-1}, t_{i-1}, t_i, c_i) \text{ , obtida pelo MAPA[5]} \tag{5.12}$$

$$\eta_3 = Contagem(G_i, t_{i-1}, w_i, t_i, c_i) \text{ , obtida pelo MAPA[6]} \tag{5.13}$$

$$\eta_4 = Contagem(G_i, t_{i-1}, t_i, c_i) \text{ , obtida pelo MAPA[7]} \tag{5.14}$$

Algoritmo para o cálculo de $\widetilde{P}(G_i \mid w_{i-1}, t_{i-1}, w_i, t_i, c_i)$ (estimativa da probabilidade):

Se E_1 existe, isto é $\delta_1 > 0$, então

$$\widetilde{P}(G_i \mid w_{i-1}, t_{i-1}, w_i, t_i, c_i) = \lambda_1 \times E_1 + (1 - \lambda_1) \times E_{23} \tag{5.15}$$

Senão se E_{23} existe, isto é $\delta_2 + \delta_3 > 0$, então

$$\widetilde{P}(G_i \mid w_{i-1}, t_{i-1}, w_i, t_i, c_i) = \lambda_2 \times E_{23} + (1 - \lambda_2) \times E_4 \tag{5.16}$$

Senão

$$\widetilde{P}(G_i \mid w_{i-1}, t_{i-1}, w_i, t_i, c_i) = E_4 \tag{5.17}$$

Onde: $\lambda_1 = \frac{\delta_1}{\delta_1 + 1}$ $\lambda_2 = \frac{\delta_2 + \delta_3}{\delta_2 + \delta_3 + 1}$ (5.18)

5.4.3.2 Estimativa das Probabilidades das Dependências Léxicas

O cálculo da estimativa das probabilidades das dependências léxicas geradas durante o processo de geração da árvore pelo analisador sintático segue o mesmo algoritmo de suavização de parâmetros descritos na seção anterior, só que as variáveis são definidas da seguinte forma:

Dado: $E_1 = \frac{\eta_1}{\delta_1}$ $E_2 = \frac{\eta_2}{\delta_2}$ $E_3 = \frac{\eta_3}{\delta_3}$ $E_4 = \frac{\eta_4}{\delta_4}$ $E_{23} = \frac{\eta_2 + \eta_3}{\delta_2 + \delta_3}$ (5.19)

Onde:

$\delta_1 = Contagem(w_i, t_i, w_{hi}, t_{hi}, \Delta_{i,hi})$, obtida pelo MAPA[8] (5.20)

$\delta_2 = Contagem(w_i, t_i, t_{hi}, \Delta_{i,hi})$, obtida pelo MAPA[9] (5.21)

$\delta_3 = Contagem(t_i, w_{hi}, t_{hi}, \Delta_{i,hi})$, obtida pelo MAPA[10] (5.22)

$\delta_4 = Contagem(t_i, t_{hi}, \Delta_{i,hi})$, obtida pelo MAPA[11] (5.23)

$\eta_1 = Contagem(R_i, w_i, t_i, w_{hi}, t_{hi}, \Delta_{i,hi})$, obtida pelo MAPA[12] (5.24)

$\eta_2 = Contagem(R_i, w_i, t_i, t_{hi}, \Delta_{i,hi})$, obtida pelo MAPA[13] (5.25)

$\eta_3 = Contagem(R_i, t_i, w_{hi}, t_{hi}, \Delta_{i,hi})$, obtida pelo MAPA[14] (5.26)

$\eta_4 = Contagem(R_i, t_i, t_{hi}, \Delta_{i,hi})$, obtida pelo MAPA[15] (5.27)

5.4.3.3 Estimativa de Probabilidade da Geração de Regras Unárias

A probabilidade da aplicação de uma regra unária é simplesmente multiplicada com a nota da estrutura sintática processada ao se inserir a regra nela. Isto não tem nenhuma base matemática no modelo de dependência criado e pode ser considerado como uma falha grave do modelo. A estimativa da probabilidade de uma regra unária do tipo *R*: *epai* → *efilho* é dada pela razão da

quantidade de vezes que a regra unária R é vista aplicada no córpus sobre a palavra w com etiqueta morfossintática t pela quantidade de vezes que w com etiqueta t é vista no córpus de treinamento. Em forma matemática:

$$P(R \mid w,t) = \frac{Contagem(< w,t >, R)}{Contagem(< w,t >)} \tag{5.28}$$

O numerador da fração é obtido a partir do MAPA[16] e o denominador a partir do MAPA[17].

5.5 Conclusão

As implementações descritas neste capítulo permitiram a execução de testes sobre o córpus Tycho Brahe revisado e a coleta de seus primeiros resultados de precisão para o Português. Além de facilitar o trabalho de anotação manual, os sistemas desenvolvidos podem ser usados para comparação segundo diversos parâmetros de córpus, conforme apresentado nos Capítulo 6 e 7 a seguir.

Capítulo 6
Resultados e Comentários

6.1 Apresentação

O objetivo deste capítulo é avaliar o desempenho dos sistemas desenvolvidos e apresentados no capítulo anterior, para a língua Portuguesa. Esta avaliação é baseada na medida de acurácia dos resultados obtidos por cada sistema na aplicação sobre o córpus Tycho Brahe. O ideal seria restringir a análise apenas para a parte de modelo e implementação de cada sistema, mas isto é impossível. Gostaríamos de responder 'Qual dos dois é melhor?', mas como veremos a seguir, a resposta que podemos dar é 'Qual dos dois é melhor nas condições testadas?'.

Avaliar a acurácia de um sistema de análise sintática automático é uma tarefa que exige cuidado. É preciso ter em mente que existem diversos pequenos fatores nas condições de teste e desenvolvimento que por si só podem alterar consideravelmente a medida de acurácia. A própria forma de se medir a acurácia é ponto de divergências na literatura. As poucas medidas usadas como padrão apresentam limitações, como falta de imparcialidade em generalização do córpus, pois beneficiam um tipo específico de anotação de córpus em relação aos demais (veja mais detalhes em [MS 00]).

Abaixo estão listados alguns pontos relevantes para a definição da acurácia dos resultados, que independem do modelo ou da implementação utilizada:

1. Tamanho do córpus de teste e treinamento (em quantidade de sentenças ou palavras).
2. Estilo lingüístico dos textos do córpus (se é um estilo gramaticalmente mais livre como poesia ou mais técnico como o conjunto de manuais e especificações técnicas).
3. Estilo de anotação de árvore utilizada (plana ou com muitas ramificações ou com muitas derivações binárias).
4. Quantidade de etiquetas de não-terminais utilizadas.
5. Forma dos itens lexicais das sentenças (só a palavra, palavra mais categoria morfossintática, uso de dicionário ou qualquer outra técnica de classificação).

6. Quantidade média de palavras por sentença no córpus de testes (os trabalhos para Inglês no começo da década de 90 eram aplicados apenas para sentenças de até 20 palavras. A partir de [Magerman 95] começou a se aplicar a sentenças até 100 palavras).

7. A medida de acurácia utilizada (usamos a mais comum, medida de PARSEVAL definida em 2.11, sem enfraquecimento algum, mas existem outras, como por exemplo a '*exact-match*', definida em [MS 00]).

8. Cuidados na separação do desenvolvimento, treinamento e testes do sistema (relacionamento entre o material anotado de cada fase).

As condições acima citadas influenciam a acurácia obtida. Mudando-se uma das características listadas, a qualidade dos resultados também muda. Portanto, deve-se ser bastante criterioso ao se fazer qualquer tipo de comparação de acurácia entre sistemas diferentes, para não incorrer no erro de se comparar medidas de tipos diferentes ou sob contextos diferentes. No nosso caso, os itens de 1 a 7 já foram apresentados nos capítulos anteriores e participarão da análise de resultados da seção 6.4. O item 8 será abordado na próxima seção.

Apesar de não sermos capazes de fazer uma análise absoluta dos modelos e das simplificações utilizadas nas implementações, podemos compará-los aplicados ao nosso contexto, apresentado no Capítulo 4 e de forma resumida a seguir:

- 3715 sentenças para treinamento com 58524 palavras ao todo.

- 432 sentenças para teste com 8378 palavras ao todo, restritas de 5 a 30 palavras por sentença. Por motivos de performance dos métodos implementados, os resultados coletados só consideraram as sentenças com até 20 palavras ao todo, o que reduziu para 228 sentenças testadas. Veja em 6.3 mais comentários a este respeito.

- 19 etiquetas de não-terminais derivadas das 281 originais aplicadas no córpus.

- 38 etiquetas de categorias morfossintáticas utilizadas nos córpus anotados.

- Utilização da medida de PARSEVAL sem enfraquecimento. Contabilizamos as medidas sobre apenas os constituintes gerados, ou seja, os constituintes da base das árvores diretamente gerados pelas etiquetas morfossintáticas/itens lexicais não são contabilizados em nenhuma das medidas de acurácia. Assim, não estamos avaliando

a árvore inteira, mas sim somente a parte que efetivamente foi gerada pelo sistema, que é o que nos interessa.

- Estilo de anotação definido em [Britto 02]
- Estilo de texto de Português Clássico, retirado de "AIRES, Matias. Reflexões sobre a Vaidade dos Homens e Carta sobre a Fortuna (seleção, prefácio e notas por Jacinto do Prado Coelho e Violeta Crespo Figueiredo). Lisboa, Imprensa Nacional - Casa da Moeda. 1980".

Assim, acreditamos estar contribuindo para o objetivo final desta dissertação, que é auxiliar no desenvolvimento e crescimento da anotação sintática do córpus Tycho Brahe.

6.2 Cuidados no Material de Desenvolvimento x Treinamento x Testes

Um erro comum em trabalhos que envolve aprendizado computacional é a falta de cuidado na separação entre o material utilizado para desenvolvimento da implementação, do utilizado para treinamento e do utilizado para testes e avaliação. Os materiais devem ser diferentes entre si, de preferência sem interseção comum.

Se usarmos para teste um subconjunto do material utilizado para treinamento, teremos resultados tendenciosos, que falsamente apresentam melhora nas medidas de acurácia. Neste caso, não estamos testando o desempenho do modelo/implementação para a tarefa de processamento de linguagem natural desejada, mas sim, estamos testando a capacidade do nosso sistema em aprender os exemplos específicos do córpus de treinamento, e não do córpus geral ao qual ele pertence. No caso extremo o sistema se vicia ao córpus treinado, isto é, funciona bem com aquele córpus de teste, mas se for inserido um outro pertencente à mesma categoria do anterior e não contido no de treinamento, seus resultados serão inferiores. Neste trabalho, o Segmentador descrito em 5.2.1 se responsabiliza por fazer a devida separação entre córpus de teste e de treinamento.

Da mesma forma, deve-se tomar o cuidado com o material utilizado para desenvolvimento e depuração da implementação do sistema. Se ele também não for à parte em relação ao material de treinamento e de teste, corre-se o risco de se fazer implementações não previstas pelo modelo, mas que '*ajudam*' muito nos resultados dos testes. O caso extremo seria um sistema especialista (fantasioso, pelo esforço necessário) aplicado aos dados de teste.

Neste trabalho utilizamos para desenvolvimento dos algoritmos cerca de 2000 sentenças anotadas e parcialmente revisadas presentes no arquivo chamado '*cavaleiro.mjc*', passado pelo grupo de Sintaxe Histórico do IEL/UNICAMP. Com estes dados para desenvolvimento, simulamos tanto a fase de treinamento como a de teste, segmentando o arquivo em 90% das sentenças para treinamento e 10% para teste. Os resultados obtidos foram cerca de 10% superiores aos apresentados na próxima seção. Além dos fatores citados acima, o motivo da diferença também está no fato das sentenças de teste terem sido limitadas na faixa de 5 a 9 palavras, bem menores que as sentenças utilizadas nos testes finais. Tais resultados não foram mais documentados para não poluir o capítulo com dados tendenciosos.

6.3 Resultados dos Testes Sobre o Córpus Revisado

Os resultados de acurácia obtidos em nossos testes sobre o contexto citado no Capítulo 4 estão apresentados na tabela 6.1 abaixo:

Método	LR	LP	NLR	NLP	%CB	%SUC
GLCP	54,8%	59,2%	59,9%	64,4%	20%	100%
ONL	50,1%	52,2%	56,6%	57,7%	21%	80%

Tabela 6.1: Resultados de acurácia de PARSEVAL e outras informações sobre os testes efetuados nos dois sistemas de análise sintática: GLCP e o orientado ao núcleo-léxico (ONL)

Os códigos das colunas significam o seguinte:

- **LR** indica a medida de cobertura de PARSEVAL.
- **LP** indica medida de precisão de PARSEVEL.
- **NLR** indica a medida de cobertura baseada na de PARSEVAL, mas sem considerar as etiquetas dos não-terminais comparados.
- **NLP** indica a medida de precisão baseada na de PARSEVAL, mas sem considerar as etiquetas dos não-terminais comparados.
- **%CB** indica o percentual dos constituintes gerados pelo analisador sintático com colchetes atravessantes em relação aos constituintes da árvore correta, ou seja, cuja

definição de localidade é incompatível com a estrutura da árvore correta. Quanto menor este índice, melhor o resultado obtido.

- **%SUC** indica o percentual de sentenças de teste que foram completamente processadas sintaticamente. Isto é, o percentual de sentenças que o analisador conseguiu encontrar pelo menos uma árvore sintática que gera a sentença completa.

As definições das medidas anteriores podem ser vistas em 2.11.

Na prática dos testes constatou-se uma grande dificuldade para o modelo GLCP realizar a análise sintática em sentenças com mais de 20 palavras. O maior problema, como será explicado em 6.4.1.1, está na forma como foi implementado o algoritmo de procura. Há uma explosão rápida da quantidade de constituintes gerados na carta de análise. Isto acarreta num aumento exponencial do consumo de memória física da máquina exigido pelo algoritmo, o que faz a tarefa ser impraticável para sentenças maiores de 20 palavras mesmo com 1GByte de RAM disponível (este limite foi avaliado empiricamente para os dados utilizados de teste). Por questão de justiça na comparação de resultados, limitamos todos os testes às sentenças de 5 a 20 palavras (228 das 432 originais de teste), mesmo com o ONL conseguindo executar a análise sintática para sentenças até 30 palavras dentro do nosso ambiente de teste.

Os resultados tabelados são a primeira vista bem inferiores aos reportados em trabalhos similares para a literatura inglesa, como os LR = 71,7%; LP=75,8% da GLCP do [Charniak 96] e os LR = 85,8%; LP=86,3% de [Collins 96] para seu método ONL baseado em dependências léxicas. Isto será analisado na seção 6.4.2 adiante. Também pode-se estranhar o fato da GLCP apresentar resultados melhores que o ONL nos nossos testes, quando esta possui um modelo teoricamente mais sensível para a distinção de árvores do que aquela. Uma análise a este respeito está presente na seção 6.4.1.

6.4 Análise dos Resultados

A documentação da análise foi dividida em duas partes para simplificação de raciocínio. A primeira tenta comparar os resultados entre os diferentes sistemas, levantando possíveis justificativas. A segunda foca na comparação dos resultados obtidos para o Português no Tycho Brahe em relação aos reportados em trabalhos anteriores aplicados ao Inglês.

Para auxiliar nas duas análises, algumas estatísticas a respeito de treinamento e de teste estão nas tabelas 6.2 para o GLCP e 6.3 para a ONL. As variáveis presentes na coluna 6 da

tabela 6.3 estão definidas em 5.4.2, dado o seguinte mapeamento: Gi = G_i , ti = t_i , whi = w_{hi} , thi = t_{hi} , wii = w_{i-1} , tii = t_{i-1} , di = Δ_i ,Ri = R_i , Runitaria = *epai_efilho* .

#	Descrição	Valor
C1	Quantidade de regras coletadas	45195
C2	Quantidade de tipos de regras identificadas	3701
C3	Quantidade de regras amostradas uma única vez	2413 (65.20% de **C2**)
C4	Quantidade de tipos de regras unárias	84 (2.27% de **C2**)
C5	Quantidade de regras unárias amostradas	14716 (32.56% de **C1**)
C6	Quantidade de itens lexicais processados	58524

Tabela 6.2: Estatísticas de amostragens das regras gramaticais colhidas na fase de treinamento da GLCP

# Mapa	Total de Tipos de Eventos Coletados (T1)	Total de Eventos Coletados Vistos uma Única Vez (T2)	Proporção (T2/T1)	Média de contagem por Evento	Sintaxe do Evento
0	26600	21551	81,02%	1,72	<wii, tii, wi, ti, ci>
1	12829	9101	70,94%	3,57	<wii, tii, ti, ci>
2	13179	8807	66,83%	3,47	<tii, wi, ti, ci>
3	892	200	22,42%	51,28	<tii, ti, ci>
4	28762	23888	83,05%	1,59	<Gi, wii, tii, wi, ti, ci>
5	14763	10871	73,64%	3,10	<Gi, wii, tii, ti, ci>
6	15631	10983	70,26%	2,93	<Gi, tii, wi, ti, ci>
7	1585	477	30,09%	28,86	<Gi, tii, ti, ci>
8	27814	24317	87,43%	1,35	<wi, ti, whi, thi, di>
9	15467	11562	74,75%	2,43	<wi, ti, thi, di>
10	16939	12992	76,70%	2,22	<ti, whi, thi, di>
11	2738	1131	41,31%	13,73	<ti, thi, di>
12	28347	25012	88,24%	1,33	<Ri, wi, ti, whi,thi, di>
13	16832	12944	76,90%	2,23	<Ri, wi, ti, thi, di>
14	18374	14550	79,19%	2,05	<Ri, ti, whi, thi, di>
15	4536	2444	53,88%	8,29	<Ri, ti, thi, di>
16	3979	2530	63,58%	3,92	<Runitaria, wi, ti>
17	5540	2996	54,08%	7,42	<wi, ti>

Tabela 6.3: Informações de amostragens dos eventos coletados para cada mapa pertencente ao ONL

6.4.1 Análise dos Resultados da GLCP e do ONL no Tycho Brahe e Comparação

Os dados mencionados na análise desta seção estão todos presentes nas tabelas 6.1, 6.2 e 6.3 apresentadas anteriormente. O primeiro fato que chama atenção é o fato do desempenho da GLCP ter sido em média 5,9% superior ao do orientado ao núcleo-léxico (ONL). Mas antes de tentar encontrar explicações para isso, vamos tentar analisar os motivos das falhas de cada sistema.

6.4.1.1 Análise dos Resultados para a GLCP

Analisando a tabela 6.2 percebemos que a grande maioria (65,20%) dos 3701 tipos de regras gramaticais coletados pelo treinamento só foi amostrada uma única vez. Isso indica que nas árvores do córpus de teste deve existir um bom percentual de regras gramaticais que não foram amostradas nenhuma vez no córpus de treinamento. Assim, como o algoritmo de aplicação só é capaz de gerar estruturas que foram vistas na fase de treinamento, fica impossível de se obter a estrutura exatamente igual à correta em um percentual elevado das sentenças testadas.

Além disso, percebe-se que os poucos tipos de regras unárias (um pai deriva um único filho, cerca de 2,27% das 3701 existentes), foram amostradas 14716 vezes, 32,56% do total de aplicações existentes nas árvores de treinamento. Isto indica que a projeção de uma etiqueta não-terminal para outra é um fato relativamente comum a todas as sentenças do córpus.

Porém, pelo modelo GLCP, a aplicação de uma regra unária nunca irá aumentar a nota da estrutura a qual ela entra, sempre diminuirá (todas as regras unárias têm probabilidade menor que 1,0 e esta probabilidade é simplesmente multiplicada à nota da estrutura original). Assim, o modelo naturalmente tem preferência pelas estruturas com o menor número possível de nós. Isso significa que a aplicação de regras unárias muitas vezes penaliza a estrutura e o modelo terá a tendência de descartá-las em preferência a estruturas mais planas, sem a projeção. Não existe qualquer parâmetro, além da própria distribuição de probabilidade das aplicações de regras gramaticais, que seja contrário a esta tendência do modelo.

Esta limitação fica bastante evidente pela medida *LR* e *LR2*. Ambas são inferiores às suas respectivas LPs, em média 4,5%. Pela definição destas medidas de acurácia, percebe-se que o motivo desta diferença é justamente o fato das árvores geradas em média terem menos nós que as corretas anotadas no córpus.

De qualquer forma, um aspecto interessante do modelo é o fato dele ter encontrado uma árvore sintática de resposta para todas as sentenças treinadas. Uma análise mais detalhada do algoritmo mostra que o método da carta de análise tende a gerar muitas entradas na carta com muita facilidade. As próprias regras unárias criam novas entradas sem aumentar em nada a

cobertura da estrutura. Esta explosão de entradas na carta cria um efeito de bola de neve: quanto mais entradas surgem, mais entradas poderão ser geradas com a aplicação de arcos completos. O efeito prático é a lentidão do algoritmo de geração e de busca de árvore, levando até 10 minutos num Pentium III 866Mhz para processar uma sentença com mais de 20 palavras. Para sentenças com mais de 20 palavras tornou-se comum ocorrer o estouro de memória na nossa máquina de teste com 1GByte de RAM, o que então limitou a coleta de resultados para sentenças de 5 a 20 palavras do córpus de teste. Otimizações de código podem minimizar o problema, mas não mudam o fato da gramática ser super geradora. Felizmente, as estruturas incomuns normalmente recebem baixas notas e assim a árvore retornada não é tão ruim.

Os aspectos relativos à falta de sensibilidade do modelo, apresentado na 2.9.2 também colaboram para o mau desempenho do sistema. Técnicas auxiliares que acrescentem preferências às estruturas corretas, como por exemplo o fator de incentivo a ramificações à direita, também ajudariam a aumentar o desempenho do sistema.

6.4.1.2 Análise dos Resultados para a ONL

Os dados da tabela 6.3 demonstram o principal problema que o método ONL teve ao ser aplicado no córpus de teste: sub-treinamento.

O MAPA[12] armazena as contagens dos eventos de dependências completas, isto é, considerando a relação, os itens lexicais e as etiquetas *cms* do núcleo e do modificador da dependência. Percebe-se que 88% das dependências foram vistas apenas uma única vez. Esta baixa taxa de amostragem reflete na incerteza das estimativas, principalmente nas de ordem maiores. Isto já degrada bastante o resultado obtido.

O MAPA[15] armazena as contagens dos eventos de dependências de ordem menor, isto é, considerando o menor contexto coletado. Este mapa é usado no último nível do processo de ajustagem presente na estimativa de probabilidade das novas dependências. Estas estimativas são requisitadas ao se construir as novas estruturas de árvore na fase de procura do algoritmo. Pela tabela, percebemos que notavelmente 53% das dependências treinadas possuem este seu contexto sozinha, isto é, nenhuma das outras 37701 dependências treinadas tem o mesmo contexto mínimo que ela. Este índice é bastante alto! Aplicando ao córpus de teste indica que provavelmente quase todas as sentenças testadas têm dependências que não podem ser capturadas pelo algoritmo, já que este mapeia apenas o espaço das dependências treinadas. O resultado acima também sugere que o algoritmo aplicado ao córpus Tycho Brahe atual gera poucos novos constituintes, por ter poucas estimativas de parâmetros definidas num domínio muito grande. O tamanho deste domínio pode ser calculado no pior caso da seguinte forma:

Considere *TD*(X) como sendo a função que retorna o tamanho do domínio de X, onde

X = *relações* é o universo de todas as relações de dependências possíveis.

X = *cms* é o universo de todas as etiquetas de categorias morfossintáticas existentes.

X = *nt* é o universo de todas as etiquetas de não-terminais existentes.

X = *d* é o universo de todas as medidas de distâncias possíveis, de acordo com a definição em 4.4.6.

X = Q1 é o universo das possíveis respostas a questão 1 de *di* conforme 4.4.6.

X = Q2Q3, é o universo das possíveis respostas às questões 2 e 3 de *di* conforme 4.4.6.

X = Q4Q5Q6, universo das possíveis respostas às questões 4,5 e 6 de *di* conforme 4.4.6.

Assim, temos para os eventos do MAPA[15] da forma <*Ri, ti, thi, di*>, o seguinte:

$TD(\text{MAPA[15]}) = TD(relações) \times TD(cms) \times TD(cms) \times TD(di)$

Dado que $TD(relações) = (TD(nt + cms) \times TD(nt) \times TD(nt + cms))$

E $TD(\text{di}) = TD(\text{Q1}) \times TD(\text{Q2Q3}) \times TD(\text{Q4Q5Q6}) = 2 \times 3 \times 12 = 72$

$\Rightarrow TD(\text{MAPA[15]}) = (TD(nt + cms) \times TD(nt) \times TD(nt + cms)) \times TD(cms)^2 \times 72$

$\Rightarrow TD(\text{MAPA[15]}) = TD(nt) \times TD(nt + cms)^2 \times TD(cms)^2 \times 72$

Que no caso do córpus utilizado é:

$\Rightarrow TD(\text{MAPA[15]}) = 19 \times 57^2 \times 38^2 \times 72 = 6418048608$

Mais de 6 bilhões de possíveis contextos mínimos para as dependências!

Foram coletados apenas 4536 tipos deste contexto. De fato, o valor calculado acima está muito aquém do necessário para termos um córpus de tamanho '*estatisticamente relevante*' para o Português. Este tamanho é difícil de calcular teoricamente. Normalmente se utilizam treinamentos com tamanhos de córpus cada vez maiores para se identificar quando o sub-treinamento deixa de ser uma fonte de erro considerável para a aplicação do modelo testado.

Uma análise sobre os dados tabelados em 6.3 para o MAPA[4] e o MAPA[7] leva a uma conclusão análoga para o modelo de *NPs* básicos. E finalmente, analisando os mesmos dados para o MAPA[16], chega-se também à conclusão de que o sub-treinamento prejudica

muito a geração de regras unárias presentes no córpus de teste mas não vistas no de treinamento (neste caso o índice de regras unária que foram vistas uma única vez é ainda maior, 63%).

Assim, as três principais distribuições de probabilidades usadas para o processamento do modelo sofrem de estimativas pouco exatas e incompletas, quando aplicadas ao córpus Tycho Brahe utilizado. O resultado final fica então muito comprometido.

6.4.1.3 Comparação de Resultados

Os dois modelos sofrem do problema de sub-treinamento, mas seus efeitos são diferentes entre eles. No caso da GLCP, o sub-treinamento influencia mais na falta de exatidão nos parâmetros (estimativas de probabilidades) das regras gramaticais. Apesar disto, normalmente o modelo é capaz de encontrar árvores sintáticas com relativo sucesso para as sentenças de entrada. Já no caso da ONL, além de prejudicar a exatidão das estimativas, o sub-treinamento acarreta em falta de parâmetros necessários para a criação de estruturas mais completas na aplicação do algoritmo de geração de árvores. Em poucas palavras, além de se ter estimativas ruins, o método fica com pouco poder de busca de estruturas. Isto acarreta em um percentual elevado de sentenças não completamente analisadas (20% das testadas) e prejudica principalmente as medidas de PARSEVAL de cobertura.

Assim, espera-se que com o aumento do córpus de treinamento, esta vantagem dos resultados da GLCP diminua, e possivelmente até se reverta, isto é, a precisão da ONL fique maior do que a da GLCP. Claro que aprimoramentos e refinamentos adicionais nos sistemas deverão influenciar em ganho de performance no futuro.

Apesar das diferenças, ambos os modelos tendem a gerar menos nós que o correto, sendo que na GLCP este problema é mais grave. Ambos tiveram desempenhos próximos entre si na capacidade de montar uma estrutura de árvore consistente com a de entrada. Em média 20% dos constituintes das árvores geradas tem seus colchetes cruzados com os das respectivas árvores corretas.

6.4.2 Comparação Entre os Resultados da GLCP e do ONL no Tycho Brahe

Comparar os resultados obtidos neste trabalho com os reportados em trabalhos anteriores executados para o Inglês é uma tarefa perigosa. Como já foi dito na seção 6.1, corre-se o risco de se fazer conclusões erradas, devido ao fato de se estar comparando *maçãs com bananas* (conforme o dito popular). Os resultados obtidos ainda são muito incipientes para serem conclusivos quanto à existência de qualquer limitação ou vantagem das definições dos parâmetros dos modelos em relação às características particulares de cada língua. Algumas

hipóteses já foram levantadas no capítulo 4 quanto a particularidades do córpus Tycho Brahe, mas a confirmação destas exigiria um estudo mais avançado.

De qualquer forma, pode-se resumir nos seguintes pontos as possíveis justificativas para a diferença entre as línguas na performance dos modelos (está em ordem decrescente de importância):

- Tamanho do córpus de treinamento: Os córpus utilizados para os trabalhos em Inglês são em média 10 vezes maiores que o utilizado neste trabalho. Isto minimiza o efeito do erro de falta de treinamento por poucas amostragens, já que as estimativas das probabilidades dos parâmetros de cada modelo podem ser feitas com maior precisão (ou segurança).

- A gama de conhecimento coletado sobre o problema: Centenas de trabalhos já foram realizados para o Inglês neste e em outros problemas de processamento de linguagem natural correlatos. Assim, conta-se com um conjunto de estatísticas e de conhecimento de particularidades muito maior que para o caso do Português. Isso resulta em várias otimizações que foram aplicadas além do modelo estatístico em si para melhorar a performance dos analisadores para o Inglês. Em [Collins 99], usam-se técnicas específicas para tratar de problemas críticos já conhecidos para a análise sintática no Inglês. Entre elas estão: parâmetros auxiliares para 1) escolha de anexação nominal ou verbal, 2) definição de coordenação, 3) escolha de subcategorias, 4) refinamento sobre características de pontuação e 5) inserção de mais de uma saída do etiquetador morfossintático (*tagger*) na entrada do analisador sintático. Em [Charniak 96], usa-se por exemplo a multiplicação de uma probabilidade para estimular ramificações a direita nas árvores geradas. Todos estes refinamentos melhoram os resultados finais reportados.

- A qualidade do processo de lexicalização das árvores: Neste trabalho utilizamos uma técnica simplificada baseada em listas de prioridades para se realizar esta tarefa. Nos trabalhos em Inglês é comum o uso de técnicas mais refinadas inclusive com tratamento de exceções. Além disso, as listas de prioridades utilizadas neste trabalho foram construídas baseadas apenas em dados estatísticos coletados a partir do córpus de treinamento. Ordenamos a lista de acordo com a freqüência que cada tipo de nó filho era visto numa derivação no córpus de treinamento. Não houve nenhuma base lingüística para esta decisão e provavelmente um especialista da área consegue melhorar tais listas. Vale notar que a qualidade deste processo é primordial para o bom desempenho do modelo ONL já que sua argumentação é fundamentada no princípio da

localidade dos itens lexicais, que no nosso caso fica definida pela técnica de lexicalização aplicada às árvores sintáticas.

- Características das línguas: O Português é uma língua morfologicamente muito mais rica que o Inglês, o que torna o número de etiquetas sintáticas necessárias muitas vezes maior que o número de etiquetas utilizadas nas árvores sintáticas para o Inglês. Com a decisão de se reduzir as etiquetas do Tycho Brahe para apenas os seus respectivos núcleos perdem-se informações contextuais importantes para decisões na construção das estruturas das árvores sintáticas.
- Estilo de textos diferentes: Os textos em Inglês geralmente têm caráter jornalístico, às vezes até são técnicos, como de manuais de produtos e softwares. No caso do Tycho Brahe, os textos são clássicos e dos séculos XVI ao XIX, com convenções gramaticais mais livres que os textos dos estudos para o Inglês.
- Estilo de anotação diferente. O córpus do *Penn Wall Street Journal* tem um estilo de anotação em suas árvores sintáticas que facilita o melhor desempenho nas medidas de PARSEVAL, segundo [MS 00]. No Tycho Brahe existem muitas projeções sintáticas que geram regras unárias, sem dependências. Conforme visto na seção anterior, isto gera problemas tanto para o GLCP como para o ONL baseado em dependências léxicas implementado. As medidas de *LR* e *LP* reportadas para a CLCP de [Charniak 96] indicam que o córpus que ele utilizou também sofria desta mesma característica. A GLCP dele teve desempenho *LR* 4,1% inferior ao *LP*, o que significa que o sistema tinha a tendência de gerar menos nós que o correto anotado no córpus.

6.5 Conclusão

Os resultados apresentados neste capítulo são o retrato obtido desta primeira aproximação para o problema de análise sintática aplicado aos textos clássicos pertencentes ao córpus Tycho Brahe em Português. Aparentemente eles são desanimadores devido aos índices baixos de acurácia relativos aos de trabalhos similares para o Inglês. Mas por outro lado, baseado nas análises apresentadas em 6.4, deve-se estar otimista quanto ao futuro deste tipo de aplicação como foco de pesquisa e otimização em projetos. Muitos detalhes precisam e podem ser melhorados, através de mais testes estatísticos específicos e com orientação técnica proveniente de conhecimentos de caso dos lingüistas.

Mesmo com índices de acurácia baixos, os analisadores sintáticos implementados podem ser aproveitados no processo de anotação manual. Podem servir de base inicial para a

análise sintática de novas sentenças, e até mesmo como apontador de possíveis erros na anotação, no papel de conselheiro automático. Assim, espera-se ajudar não só no aumento da taxa de crescimento do córpus anotado e revisado, como também na melhora na qualidade sintática destes.

O próximo capítulo resume todo o trabalho realizado nesta dissertação, partindo do estado atual da pesquisa para focar no andamento futuro desta linha de estudo.

Capítulo 7
Conclusões Finais

7.1 O Estado Atual

O desenvolvimento do córpus Tycho Brahe do Português histórico anotado sintaticamente motivou a realização deste trabalho, com o objetivo específico de construir uma ferramenta automática para a geração das árvores sintáticas das sentenças de um texto. Esta ferramenta pode ser então aplicada em novos textos pertencentes ao material de estudo do projeto, gerando um conjunto inicialmente anotado, que pode ser revisado e refinado manualmente. A anotação sintática automática obtida ainda não é completa e tem uma performance de acurácia muito baixa frente a outros trabalhos desenvolvidos para a língua inglesa. Isso de certa forma restringe o sucesso da ferramenta implementada, mas ao mesmo tempo incentiva o seu desenvolvimento.

Assim como a própria definição das regras de anotação sintática usadas no córpus Tycho Brahe passou por uma fase de concepção e avaliação, o projeto do analisador sintático automático também tem que passar. Este trabalho foi um passo na direção de se obter os primeiros resultados úteis para auxiliar no crescimento automático deste córpus. A partir dele, pode-se partir para análises mais profundas e implementações adicionais que melhorem a acurácia final. A idéia de se iniciar com o desenvolvimento de dois modelos com parâmetros bastante distintos entre si, permite não só um crescimento fundamentado em comparações, mas também serve como base para um leque maior de possíveis trabalhos futuros.

A própria arquitetura dos sistemas implementados sugerem um crescimento do projeto em módulos, com interfaces bem definidas entre si, de tal forma a facilitar sua manutenção e desenvolvimento. O objetivo final, a construção de um analisador sintático completo e de boa acurácia para o córpus Tycho Brahe em Português, ainda exigirá muito esforço de pesquisa e desenvolvimento.

A experiência com projetos análogos para a língua inglesa sugere que o problema é muito complexo de se resolver. Os resultados superiores reportados para esta língua devem-se principalmente à acumulação de conhecimento proveniente do conjunto de diversos projetos separados, alguns inclusive com resultados pífios.

O caminho das pedras deverá ser percorrido para o Português também, mas aproveitando-se na medida do possível as experiências documentadas para o Inglês. Este

trabalho abriu apenas uma das possíveis trilhas. Com o objetivo de que ela cresça ainda mais, apresentamos a seguir algumas sugestões para trabalhos futuros correlatos.

7.2 Trabalhos Futuros: Continuando a Trilha

Abaixo estão listadas algumas tarefas que podem ser executadas em estudos futuros, com o objetivo final de melhorar a acurácia dos resultados obtidos. Todas elas deveriam aproveitar o trabalho já desenvolvido e documentado nesta dissertação.

- Aumentar o tamanho de córpus de treinamento até conseguir chegar em um tamanho '*estatisticamente relevante*'.
- Desenvolver técnicas heurísticas de correções de erros, como por exemplo, o uso do fator de estímulo às ramificações à direita na GLCP, ou o uso de regras melhores para a pontuação com maior valor de conhecimento para distinção no ONL.
- Aperfeiçoar o processo de lexicalização das árvores, com regras mais completas e listas de prioridades melhor ordenadas.
- Testar definições melhores de conjunto de etiquetas de não-terminais. O subconjunto utilizado neste trabalho foi criado da forma mais simples, retirando todos os complementos das etiquetas e mantendo apenas o tipo básico. Provavelmente existem outros subconjuntos de tamanho próximo ao utilizado, mas que façam o modelo ONL perder menos poder de discernimento.
- Coletar dados estatísticos precisos quanto às diversas hipóteses a respeito das características dos textos do córpus assumidas pelos modelos e apresentadas na seção 4.4
- Desenvolver um refinador para as etiquetas a fim de se passar do conjunto limitado em poucas dezenas de etiquetas que o desempenho dos modelos originais exigem, para potencialmente o conjunto completo de 281 etiquetas não-terminais.
- Experimentar integrar o sistema de análise sintática com o sistema de etiquetagem morfossintática a fim de se testar a geração automática da árvore sintática a partir de mais de um conjunto de etiquetas de categorias morfossintáticas por sentença.
- Experimentar métodos de estimativa de parâmetros mais precisos, possivelmente com mais níveis de ajustagem, ou até mesmo utilizando outras técnicas, como maximização de entropia. Em [Archias 03] utilizou-se a maximização da entropia para o problema de etiquetagem morfossintática automática em Português sobre o córpus Tycho Brahe.

- Desenvolver modelos matematicamente exatos e com acesso a mais objetos lingüísticos para distinção das árvores, como por exemplo, os modelos finais de [Collins 99] ou [Charniak 97].
- Desenvolver testes e coletar estatísticas sobre problemas menores, mas que podem auxiliar na tarefa como um todo. Por exemplo, o problema de decisão da anexação preposicional ao verbo ou ao nome, o problema da estimativa das subcategorias à esquerda e à direita do constituinte ou o problema da definição de coordenação.

A lista acima é bastante dinâmica com o desenvolvimento da pesquisa, isto é, a possibilidade de novos estudos aumenta à medida que novos projetos são executados e novos problemas ou soluções surgem. No caso extremo, a acurácia estará limitada apenas às deficiências do próprio modelo orientado ao núcleo-léxico. Mas antes de chegarmos a este extremo, esperamos ter contribuído para o desenvolvimento de um córpus para o Português '*estatisticamente relevante*'.

Anexo A: Arquivos do Projeto

Os arquivos do projeto estão disponíveis em:

http://www.ime.usp.br/~mfinger/alunos/fabiano/mestrado.zip

Manual

Todos os arquivos são de formato texto. De uma forma geral cada linha representa um evento do arquivo. A sintaxe das linhas é composta por elementos separados por <TAB>, de fácil utilização por outros sistemas. Basicamente, existem os seguintes arquivos:

- O arquivo executável chama-se *AnalisaSintatico.exe*. Trata-se de uma aplicação para Windows 9X/2000/NT®. As chamadas aos módulos são executadas por comando de menu, tanto do sistema GLCP como do ONL.
- O arquivo *Fab_Tese_rec2.doc* é o documento da dissertação no formato Microsoft Word 2000®.
- Os arquivos de extensão *.mjc* são os originais anotados sintaticamente, na forma 4.3, com as etiquetas de não-terminais completas.
- Os arquivos de extensão *.mjs* são os anotados sintaticamente, na forma 4.4, obtidos dos respectivos *.mjc*, com as etiquetas simplificadas apresentadas no Anexo B. Cada *.mjc* cria 3 *.mjs*, um para teste, outro para treinamento e outro com o total (os dois anteriores + erros).
- O arquivo *.rlz* define as listas de prioridades para a lexicalização das árvores.
- Os arquivos *.psr* armazenam os parâmetros coletados do treinamento do GLCP.
- Os arquivos *.onl* armazenam os parâmetros coletados do treinamento do ONL.
- Os arquivos *.txt* apresentam os resultados reportados nos processos executados em cada módulo do sistema.

Anexo B: Lista de Etiquetas Usadas

Aqui estão listadas todas as etiquetas simplificadas de não-terminais e etiquetas de categorias morfossintáticas utilizadas nos nossos testes. Seus significados estão em [Britto 02].

#	ETIQUETA
1	ADJP
2	ADJX
3	ADVP
4	CONJP
5	CP
6	FRAG
7	IP
8	LATIN
9	NP
10	NUMP
11	NX
12	PP
13	QP
14	VB
15	WADJP
16	WADVP
17	WPP
18	WQP
19	WNP

Tabela A1: Etiquetas dos não-terminais

#	ETIOUETA
1	.
2	.
3	ADJ
4	ADV
5	C
6	CL
7	CONJ
8	CONJS
8	D
9	DEM
11	ET
12	FP
13	FW
14	HV
15	HV+SE
16	INTJ
17	N
18	NEG
19	NPR
20	NUM
21	OUTRO
22	P
23	PRO$
24	PRO
25	Q
26	SE
27	SENAO
28	SR
29	SR+CL
30	TR
31	VB
32	VB+SE
33	VB+CL
34	WD
35	WADV
36	WPRO
37	WPRO$
38	WQ

Tabela A2: Etiquetas das categorias morfossintáticas

Referências Bibliográficas

[AlvFinger 99] Alves, Carlos Daniel Chacur e Finger, Marcelo, 1999. Etiquetagem do Português Clássico Baseada em Corpus in *IV Encontro para Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR99), Portugal.*

[Archias 03] Filho, Archias Alves de Almeida, 2003. *Maximização de Entropia em Lingüística Computacional para a Língua Portuguesa*, Dissertação de Mestrado, Departamento do Computação, IME – USP.

[BD 77] Bickel e Docksum, 1977. *Mathematical Statistics: Basic Ideas and Selected Topics*. Prentice Hall, Englewood Cliffs, New Jersey.

[Bikel et al 97] Bikel, D. M., Miller, S., Schwartz, R. e Weischedel, R., 1997. Nymble: a High-Performance Learning Name-finder in *Proceedings of the Fifth Conference on Applied Natural Language Processing*, páginas 194-201.

[Black et al 92] Black, E., Lafferty, J. e Roulos, S., 1992. Development and Evaluation of a Broad-Coverage Probabilistic Grammar of English-Language Computer Manuals in *Proceedings of the 30th Annual meeting of the Association for Computational Linguistics*, páginas 185-192.

[Bonfante 98] Bonfante, Andréia Gentil, 2003. *Parsing Probabilístico Para o Português do Brasil*, Tese de Doutorado, Instituto de Ciências Matemáticas e Computação, ICMC-USP.

[Britto 02] Britto, Helena, 2002. *Tycho Brahe Syntactic Annotation System, First version Revised*, Projeto *Rhythmic Patterns Parameter Setting & Language Change* url: http://www.ime.usp.br/~tycho/corpus/manual/syntax2.html - acessado em 16/09/2003.

[Cáccamo 98] Cáccamo, Mário José, 1998. *Um Ambiente para a Análise Superficial de Línguas Baseado em Autômatos Finitos*, Dissertação de Mestrado, Instituto de Computação, UNICAMP.

[Charniak 93] Charniak, E., 1993. *Statistical Language Learning*. The MIT Press.

[Charniak 96] Charniak, E., 1996. *Tree-bank Grammars*, Department of Computer, Brown University.

[Charniak 97] Charniak, E. , 1997. *Statistical Parsing with a Context-free Grammar and Word Statistics*, Department of Computer, Brown University.

[CG 96] Chen, S. e Godman, J., 1996. An Empirical Study of Smoothing Techniques for Language Modeling in *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics,* páginas 310-318.

[Chen 98] Chen, Kuang-hua, 1998. Digital Libraries, Metadata and Text Processing *in Proceedings of the First Asia Digital Library Workshop, Hong Kong,* páginas 123-135.

[Collins 96] Collins, Michael, 1996. A new Statistical Parser Based on Bigram Lexical Dependencies in *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics,* páginas 184-191.

[Collins 99] Collins, Michael, 1999. *Head Driven Statistical Models For Natural Language Parsing*, Tese de Doutorado, Department of Computer and Information Science, University of Pennsylvania.

[Frank 92] Frank, R., 1992. *Syntactic Locality and Tree Adjoining Grammar, Gammatical, Acquisition and Processing Perspectives*, Ph.D. Thesis, University of Pennsylvania.

[HU 79] Hopcroft, John e Ullman, Jeffrey, 1979. *Introduction to Automata Theory, Languages and Computation,* Series in Computer Science, Addison Wesley.

[Jelinek 90] Jelinek, F., 1990. Self-organized Language Modeling for Speech Recognition in *Readings in Speech Recognition.*

[Joshi 87] Joshi, A., 1987. An Introduction to Tree Adjoining Grammars in *A. Manaster-Ramis, Mathematics of Language*, John Benjamins, Amsterdam.

[Magerman 95] Magerman, David M., 1994. *Natural Language Parsing as Statistical Pattern Recognition,* Tese de Doutorado, Department of Computer Science, Stanford University.

[MS 00] Manning, Christopher D. e Shütze, Hinrich, 2000. *Foundations of Statistical Natural Language Processing*, capítulo 12, The MIT Press.

[RN 95] Russel, S. J. e Norving, P., 1995. *Artificial Intelligence: A Machine Approach.* Englewood Cliffs, N.J.: Prentice Hall.

[Schütz 03] Schütz, Ricardo, 2003. *Diferenças Idiomáticas Entre Português e Inglês*, Projeto *English Made in Brazil*
url: http://www.english.sk.com.br/sk-idiom.html - acessado em 16/09/2003.

[Winograd 83] Winograd, T., 1983. *Language as Cognitive Process*, Volume I: Sintax, Addison-Wesley, Reading, MA.