Parsing Probabilístico para o Português do Brasil

Andréia Gentil Bonfante[†]

Orientadora:

Dra. Maria das Graças Volpe Nunes

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Doutor em Ciência da Computação.

USP - São Carlos

[†]Este trabalho contou com apoio financeiro do CNPq.

Resumo

A análise sintática automática de sentenças (parsing) é uma tarefa vital para a maioria das aplicações que envolvem Processamento de Língua Natural (PLN). Para que os parsers tenham robustez e boa precisão, eles devem se basear em gramáticas abrangentes e, por isso, complexas. O processo manual de definição de uma gramática "ideal" envolve um alto grau de conhecimento lingüístico e também um alto risco de ambigüidade, dadas as variedades sintáticas envolvidas. Alguns métodos de aprendizado de máquina podem ser usados para a obtenção de gramáticas a partir de grandes conjuntos de sentenças analisadas (treebanks). Entre eles, destacam-se os estatísticos, cuja característica é atribuir probabilidades às análises encontradas e, com isso, conseguir classificá-las, a ponto de produzir as mais prováveis para uma determinada sentença. O objetivo desta tese foi investigar o comportamento de um desses métodos estatísticos quando usado para analisar sentenças da língua portuguesa do Brasil. O método implementado usa um modelo gerativo de destaque na literatura, cuja característica é considerar o núcleo como o elemento principal e direcionador de todo o processo de análise. Como produto, apresenta-se o ambiente PAPO, formado por vários módulos que executam três funções básicas: (1) o pré-processamento e a 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 parser propriamente dito, que usa um dos modelos gerados e produz as árvores sintáticas mais prováveis para uma sentença. Uma avaliação qualitativa dos casos de teste é apresentada.

Abstract

The syntactic analysis of sentences (parsing) is an essential task for the majority of Natural Language Processing (PLN) applications. For parsers to be robust and have good accuracy, they must be based on a comprehensive and, therefore, complex grammar. The manual specification of an ideal grammar deals with a big amount of linguistic knowledge and ambiguity, given language syntactic variety. Some machine learning (ML) techniques may be used to obtain grammars from large sets of syntactically analyzed sentences (treebanks). The main characteristic of statistical ML techniques is to associate probabilities to the possible syntactic analyses in order to classify them for producing the most probable ones for a given sentence. The main goal of this work was to investigate the behavior of a statistical method applied to the analysis of sentences in Brazilian Portuguese. The implemented method uses a well known generative model from the literature, whose characteristic is to consider the head of a sentence as main element, which drives all the steps of the analysis. As a product, it was built the PAPO system, which executes three basic tasks: (1) data pre-processing and preparation of the set of sentences to be used during the training phase, (2) the generation of two probabilistic models of analysis (PAPO-I and PAPO-II) and, finally, (3) a parser that uses one of the generated models and produces the most probable syntactic analysis for a sentence. A qualitative evaluation of the test set is also presented.

À minha família

Especialmente aos meus pais e meus avós

Agradecimentos

Em primeiro lugar, agradeço o apoio financeiro do CNPq, sem o qual este trabalho não poderia ter se realizado.

Em segundo lugar, quero deixar aqui registrado o afeto e o agradecimento às pessoas que, de alguma forma, contribuiram para minha formação. Dedico a todos os amigos do ICMC, que tanto enriquecem e suprem minhas limitações e carências.

Lembro-me de minha mãe profetizando que "Deus coloca as pessoas certas, mas somente na hora propícia, no caminho da gente". De certa forma, isso aconteceu comigo. Devo agradecer imensamente ao Jorge P., que foi a "luz" no momento mais desesperador de meu trabalho. Semanas que varamos, de domingo a domingo, tentando achar um formato para o PAPO, que se recusava em "nascer" com todas as suas forças. Taí um exemplo mais que perfeito de tal teoria. Se o PAPO tomou forma, e espero que ainda se torne muito robusto, é graças à sua crucial ajuda. Ficarei em eterna divida com você, Jorge. E também com você, Val, por todas as figuras que fez para incluir nesta tese.

Quero aqui também, expressar minha profunda admiração pela Graça, como orientadora e amiga. Os trabalhos no NILC, sob sua coordenação e dos igualmente queridos, Lúcia, Sandra, Chu e Bento, alavancam pesquisas nessa área tão esquecida e pouco representada no Brasil.

Aliás, quero também explicitar minha admiração por todos da equipe NILC. Apesar de muitas serem as provações e as privações dessa vida acadêmica, não deixamos de nos reunir para momentos de descontração. Quantas alegrias passamos juntos. Nunca esquecerei dos churrascos maravilhosos na casa da Graça, dos filmes que assistimos, regados a pizza e boa conversa e dos e-mails do Thiago, mobilizando os mais acomodados e os ratos de festas (Raquel, Ariani, Anselmo, Valéria, Lúcia S., Gawa, Graça, Helena e Juliana) para os *happy-hours* de quinta; e também altas discussões e profecias da Gi e do Ronaldo (duas figuras!!), e de todos os outros amigos que estão e que passaram por aqui.

Não posso também deixar de mencionar a importância da Crau, minha irmãzinha de coração, nesse meu percurso. O que mais me agrada de ter seguido esse caminho foi poder compartilhar contigo, Crau, uma amizade tão duradoura e verdadeira. Obrigada por todas as vezes que você me confortou e me mostrou o caminho correto a seguir. E por todos os momentos de alegrias também, que passamos juntas; dos almoços de domingo em casa, e na casa do Gustavo, e dos passeios confortantes com a Alice e a Sofia.

E por último, mas não menos importante, agradecer demais o apoio de meus pais, e de toda a minha família, sem o qual, acho que não conseguiria passar por todos essas armadilhas que a vida apronta.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Muito esforço tem sido empregado na construção de sistemas de processamento de língua natural (PLN)[1]. Durante os anos 70, os sistemas construídos demonstravam aspectos interessantes de interpretação de língua em domínios restritos, como o Mundo dos Blocos (Winograd 1970) ou sistemas de perguntas e respostas a bases de dados ((Woods 1977), (Waltz 1978)). Nos anos 80, as pesquisas se voltaram à construção de sistemas de PLN robustos (Allen 1995), com gramáticas mais elaboradas, ainda na forma de regras compiladas manualmente (*hand-coded*). No entanto, os sistemas continuaram de domínio restrito e, portanto, bastante limitados; a codificação do conhecimento necessário chegou a níveis de grande complexidade, e o aumento na quantidade de regras, juntamente com a conseqüente dificuldade de gerenciamento, fizeram com que as pesquisas não obtivessem avanços consideráveis durante esses anos. A partir da década de 90, o interesse passou a ser por novos paradigmas que superassem as dificuldades até então encontradas, como a necessidade de conhecimento de um especialista para a construção da gramática. A aquisição do conhecimento necessário passa, então, a ser automática, num processo de aprendizagem a partir de grandes bases de exemplos ((Church e Mercer 1993), (Charniak 1993), (Brill e Mooney 1997)), habilitando tais sistemas a fazer previsões sobre novos exemplos.

Nesse paradigma, chamado de empírico, são três as abordagens principais das áreas de aprendizado: o aprendizado de máquina simbólico, o aprendizado neural conexionista e o aprendizado estatístico. As abordagens de aprendizagem simbólica usam formas de representação como árvores de decisão (Hermjakob e Mooney 1997), regras de transfor-

[1] Optou-se, aqui, pelo uso do termo *língua*, ao invés de *linguagem*, para tradução de *language* na expressão *Natural Language Processing*.

mação ((Brill 1993), (Brill 1995)) e programação lógica indutiva ((Zelle e Mooney 1996), (Mooney 1996), (Wermter, Rillof, e Scheler 1996)). Os modelos conexionistas têm nas redes neurais recorrentes de Elman ((Elman 1990), (John e McClelland 1990), (Reilly e Eds. 1992), (Miikkulainen 1993), (Lawrence, Fong, e Giles 1996)) e algumas extensões, como a proposta de Shastri e Ajjanagadde (1993) e de Lane e Henderson (1998), conforme observado em (Bonfante e Nunes 1997) e (Bonfante e Nunes 1999), seus melhores representantes no aprendizado de línguas. Já os modelos estatísticos são os mais usados, conforme a literatura. Quando aplicados a problemas de processamento de língua, usam técnicas como n-gramas, Modelos Ocultos de Markov (*Hidden Markov Models*), Gramáticas Livres de Contexto Probabilísticas ((Charniak 1993), (Charniak 1997)) ou Árvores de Decisão Probabilísticas (Magerman 1995).

Sabe-se que um dos requisitos fundamentais para o bom funcionamento de sistemas de processamento de língua, especialmente quando é necessária a interpretação (como em extração de informação, tradução automática, reconhecimento de fala, etc.), é a eficiência na recuperação da estrutura sintática das sentenças. Essa análise sintática, ou parsing, é realizada pelo parser, que tem por função recuperá-la com a ajuda de uma gramática, ou, no caso da aprendizagem, utilizando um banco de sentenças, também chamado de *corpus* ou *treebank*. Uma treebank é uma coleção (grande) de sentenças autênticas anatodas com informações sintáticas e morfossintáticas. Exemplos desses corpora são o Penn Treebank (Marcus e et al. 1993), para o inglês, e o CETENFolha[2], para o português do Brasil.

O parsing pode também ser tratado como um problema de aprendizado de máquina. Uma função *sentença* $\rightarrow$ *árvore* é induzida a partir de um conjunto de treinamento, constituído de exemplos *sentença-árvore sintática*. A avaliação da precisão[3] do modelo é feita usando-se um conjunto de teste contendo sentenças não participantes do treinamento. No caso dos métodos estatísticos, o aprendizado se torna, então, uma tarefa de estimativa de valores de parâmetros a partir dos dados de treinamento. A combinação dos melhores parâmetros forma a árvore mais provável para uma dada sentença. Esta tese aborda o problema do parsing de sentenças para a língua portuguesa brasileira usando um desses métodos estatísticos.

Métodos Estatísticos

A grande motivação para a escolha dos métodos estatísticos para parsing é o problema

[2] Uma porção do corpus NILC (Núcleo Interinstitucional de Lingüística Computacional), consistindo de sentenças compiladas de textos jornalísticos, etiquetado com o parser "*Palavras*", de Bick (2000). Disponível no endereço: http://www.linguateca.pt

[3] Tradução do termo inglês *precision*, calculada como: $\frac{número\ de\ constituintes\ corretos\ na\ análise\ proposta}{número\ total\ de\ constituintes\ na\ anúlise\ proposta}$

da ambigüidade, que se manifesta na sintaxe em casos como:

- Ambigüidade de categoria sintática. Por exemplo, a palavra *banco* pode ser tanto um verbo quanto um substantivo.
- Ambigüidade na ligação de sintagmas preposicionais. A sentença *A menina viu o menino de binóculo* tem no mínimo duas árvores sintáticas: uma com o sintagma *de binóculo* modificando *menino* (ou seja, o menino é que estava de binóculo) e outra, na qual ele modifica *menina* (ou seja, a menina estava de binóculo e viu o menino).
- Ambigüidade em coordenação. Na sentença *um programa para promover segurança em caminhões e peruas*, a palavra *peruas* pode ser coordenada com *caminhões*, *segurança* ou *programa*, gerando várias árvores sintáticas possíveis.

Conforme observado por Church e Patil (1982), a ambigüidade sintática gera uma explosão exponencial de análises para uma sentença. Segundo Collins (1999), o problema com este e outros tipos de ambigüidade é ainda agravado quando se trata de sentenças longas, tornando necessária uma gramática bastante ampla. Tudo isso motivou as pesquisas por métodos estatísticos, que usam probabilidades observadas no conjunto de treinamento para ajudar a tomar decisões de ligações entre as palavras. Como já observado em Bonfante e Nunes (2000), as pesquisas iniciais investigavam o uso das gramáticas probabilísticas livres de contexto, mas os resultados não foram satisfatórios. Outros caminhos apontaram para modelos com sensibilidade estrutural aumentada, os modelos baseados na história da análise (*history-based*).

O elemento principal de um modelo estatístico é a sua parametrização. Os parâmetros são as informações que são utilizadas para estimar as probabilidades da *treebank* usada como treinamento. Assim, os métodos estatísticos na literatura diferem pelo modo como tratam esses parâmetros, ou seja, como determinam quais objetos lingüísticos são considerados para estimar os parâmetros de seu modelo. A opção ideal é associá-los a árvores inteiras, o que é inviável, pois seria necessário que todas as sentenças a serem analisadas já tivessem sido vistas pelo parser, o que não acontece. Sendo assim, a escolha de uma parametrização passa a ser a escolha de como "decompor" uma árvore sintática em eventos/decisões discretas, de forma que os parâmetros associados a esses eventos representem o máximo de informação possível. Segundo Collins (1999), dois critérios são importantes nessa escolha: (1) o poder discriminativo, que dita o quanto de informação contextual os parâmetros possuem para resolver questões de ambiguidade; e (2)

a compactação, que consiste na utilização da menor quantidade possível de parâmetros, mantendo seu poder discriminativo.

Nas gramáticas probabilísticas, os parâmetros são associados às regras. As árvores sintáticas são desmembradas em suas regras constituintes e a cada regra é associada a probabilidade de ocorrência para aquele conjunto de sentenças observado. Os primeiros trabalhos (Charniak 1997) apresentaram precisão em torno de 75% e não foram muito satisfatórios pelo fato de o modelo não ter sensibilidade quanto a informações lexicais e preferências estruturais.

Os modelos baseados na história da análise superam as dificuldades das gramáticas probabilísticas associando os paramêtros a eventos, em que cada evento contribui para a formação da árvore como um todo. Exemplos mais marcantes são os modelos do parser SPATTER (Magerman 1995) (uma extensão do trabalho descrito por Jelinek e et al. (1994)), do parser de Collins (1999) e do parser baseado em máxima entropia, proposto por Ratnaparkhi (1999).

O SPATTER apresentou um amadurecimento quanto ao uso de técnicas estatísticas em vários aspectos:

- Representou um maior avanço na escala de tarefas dos parsers estatísticos. Conseguiu analisar sentenças de um jornal americano, o *Wall Street Journal*, em um domínio muito menos restrito que os domínios anteriores, constituídos basicamente de manuais de computação.

- O parser foi treinado de uma forma completamente automática a partir da *treebank*, sem qualquer necessidade de uma gramática.

- Os resultados foram mais precisos que os das gramáticas probabilísticas: 84.5%/ 84% de precisão/cobertura[4] na seção 23 da *Penn Wall Street Journal Treebank*.

- O modelo tem parâmetros condicionados a informações lexicais, o que ajudou em performance, superior a das gramáticas probabilísticas.

Em 1999, Collins apontou falhas no modelo do SPATTER quanto à escolha dos critérios de decomposição da árvore, e propôs seu modelo baseado na teoria *X-barra* de Chomsky, na qual o núcleo do sintagma é projetado para seu pai num processo recursivo

[4]Do inglês *recall*, calculada como: $\frac{número\ de\ constituintes\ corretos\ na\ aánlise\ proposta}{número\ total\ de\ constituintes\ na\ treebank}$
Obs: No Apêndice B, é apresentado mais detalhes sobre as medidas precision e recall.

até que a raiz da árvore tenha seu núcleo preenchido. Isto é o que ele chama de *princípio da localidade*, cujo domínio é a região do espaço-tempo que ele pode afetar. Assim, de acordo com a localidade dos núcleos lexicais, cada palavra na sentença afeta um domínio específico na árvore. Isto leva a que cada núcleo seja gerado antes de toda a estrutura que lhe é dependente. As suposições de independência dos parâmetros e a escolha da decomposição da árvore também são influenciadas pela localidade do núcleo.

No modelo Collins, a árvore sintática é formada em vários estágios. O primeiro envolve a escolha do eixo principal da subárvore. Considere, por exemplo, a sentença *A menina comprou uma boneca nova*[5], apresentada na Figura 1.1. A primeira subderivação é a do eixo principal, conforme mostrado na Figura 1.2, do núcleo *comprou* da sentença. Neste ponto, são duas as decisões aí envolvidas. A probabilidade é dada para um parâmetro específico ligado ao núcleo: $P(Verbo|P, comprou)$ é a probabilidade de um nó P com *comprou* como núcleo e verbo como etiqueta morfossintática; e $P(S|P, comprou)$, a probabilidade de um eixo principal S tendo como núcleo *comprou* com etiqueta sintática P. Nos próximos estágios são tomadas as outras decisões de subcategorização: por exemplo, simplificadamente, como na Figura 1.3, as probabilidades estimadas são as que associam probabilidades ao sujeito (à esquerda do núcleo) e ao objeto direto daquela árvore (à direita do núcleo):

P(SUBJ(menina)|pai=S,nucleo=P,comprou,esquerda)
P(ACC(boneca)|pai=S,nucleo=P,comprou,direita)

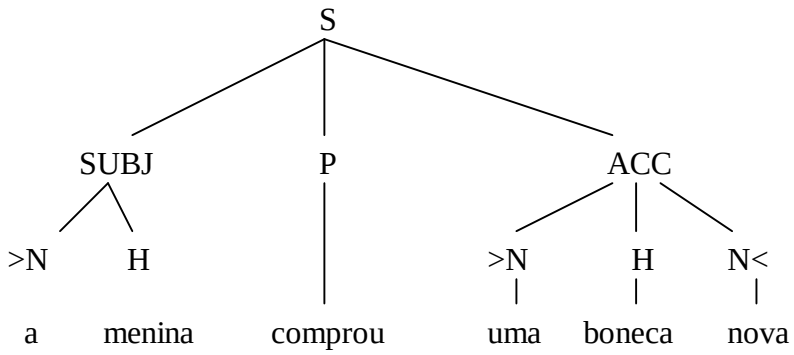

Figura 1.1: Árvore Sintática da sentença *A menina comprou uma boneca nova.*

[5] A anotação usada é a proposta por Bick (2000), na qual a sentença S é formada pelo sujeito (SUBJ), por um verbo (P) e por um complemento (ACC-objeto direto acusativo). O sujeito, por sua vez, é formado por um modificador pré-nominal (>N) e por um núcleo (H). E o objeto, composto por um modificador pré-nominal (>N), um núcleo (H) e um modificador pós-nominal (N<). Obs: as etiquetas morfossintáticas foram omitidas da figura, mas elas fazem parte da sentença. Assim, *a* é artigo, *menina* é nome, *comprou* é verbo, *uma*, artigo, *boneca*, nome e *nova*, adjetivos.

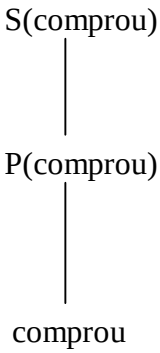

Figura 1.2: Eixo principal da sentença *A menina comprou uma boneca nova.*

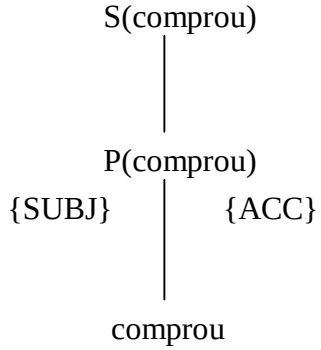

Figura 1.3: Complementos do eixo principal da sentença *A menina comprou uma boneca nova.*

Com essas informações, são montados os modelos que representam todas as relações de dependência existentes nas árvores observadas durante a fase de treinamento. Cada dependência é constituída basicamente por um par modificador-núcleo, a indicação da posição desse modificador (à esquerda ou à direita do núcleo) e o pai dessa subestrutura. A tarefa do parser é utilizar essas informações e, num processo de baixo para cima (*bottom-up*), que começa nas palavras de uma sentença dada, tentar recuperar a melhor árvore sintática combinando cada probabilidade de cada dependência, maximizando uma função $P(AS, S)$, que é a probabilidade de uma sentença S aparecer associada a uma árvore sintática AS.

Uma outra abordagem para a construção de parsers é usar modelos probabilísticos lexicalizados para ajudar no preenchimento das tabelas LALR, usadas pelos parsers de "empilhar-reduzir" (*shift-reduce*). Exemplo de tal modelo é o parser RADISP do projeto RASP[6] (Briscoe e Carrol 2002). Esse projeto consiste na construção de um ambiente

[6] http://www.cogs.susx.ac.uk/lab/nlp/rasp/

modular de análise no qual a busca pela análise de um texto é alcançada em várias etapas: tokenização, etiquetação morfossintática, análise morfológica e lematização dos tokens etiquetados e, finalmente, a análise sintática, feita usando-se uma gramática de domínio geral, manualmente construída. O passo final da análise é a seleção das n melhores análises da floresta sintática obtida, usando um modelo de seleção probabilístico condicionado ao contexto estrutural da análise e informações lexicais, quando aparecem.

Objetivo

O objetivo desta tese inclui investigar, entender, implementar e analisar o comportamento e a viabilidade de uso de um método estatístico para analisar sentenças da língua portuguesa do Brasil, reconhecidamente com estruturas mais complexas que as da língua inglesa. O modelo e o mecanismo de busca usados seguiram os moldes propostos por Collins (1999). Trata-se da primeira iniciativa em se investigar métodos estatísticos nesse domínio. De fato, desconhecem-se trabalhos análogos também para o português de Portugal.

Devidamente consolidados na área, com propostas como as apresentadas anteriormente e outras abordagens, como a híbrida do projeto RASP, os parsers estatísticos para a língua inglesa já atingem o patamar próximo a 90% de acerto. Os autores dispõem de várias *treebanks* anotadas manualmente para realizarem experimentos de aprendizagem e teste. Uma *treebank*-referência (*benchmark*) bastante usada é a *Penn Treebank* (Marcus e et al. 1993), anotada manualmente a partir de sentenças do *Wall Street Journal*.

Para a língua portuguesa do Brasil, a única *treebank* de domínio não restrito disponível é a CETENFolha, subproduto do corpus NILC[7] (Pinheiro e Aluísio 2003), anotada automaticamente com o parser "Palavras" [8] (Bick 2000)[9]. O esquema de anotação também difere da usada na *Penn Treebank*. A anotação segue o paradigma das gramáticas de restrição, com etiquetas funcionais, como mostrado no exemplo da Figura 1.1, com noções de função como: sujeito, objeto direto, modificador de nome, adjunto, argumento de preposição, etc., contendo um número maior de etiquetas (ver Apêndice A para uma descrição detalhada das etiquetas).

Como produto desta tese, foi criado o sistema PAPO, composto por alguns submódulos responsáveis pelo pré-processamento e preparação dos dados e construção dos

[7] http://www.nilc.icmc.sc.usp.br/tools.html

[8] Segundo Bick, a precisão do parser é de cerca de 98%.

[9] Está em andamento o projeto de construção do parser *Curupira* (Martins, Hasegawa, e Nunes 2002), um parser simbólico que deverá ser responsável pela anotação sintática de uma *treebank* do NILC.

modelos estatísticos, e o parser propriamente dito, de mesmo nome, que utiliza todas estas informações recolhidas. Previa-se, desde o início, que as dificuldades desse trabalho seriam muitas. As tarefas de implementação foram complexas e longas e os resultados alcançados, embora ricos em informação, não garantem ainda uma análise definitiva do problema. A avaliação parcial foi feita sobre uma amostra de sentenças autênticas e mostra que os problemas que merecem investigação futura se concentram, entre outros, em: (1) incluir novos exemplos na *treebank* de treinamento, bem como balanceá-los; (2) identificar a fonte dos ruídos observados quando as sentenças de treinamento são transformadas em estruturas árboreas (a partir da anotação "plana" de Bick); (3) implementar possíveis extensões ao modelo de forma a lidar com o problema da não-aprendibilidade e (4) investigar mecanismos de busca heurística que lidem melhor com a explosão tanto do uso de memória quanto de tempo.

Se, por um lado, as dificuldades surgidas durante o trabalho frustaram as expectativas de se construir um parser robusto, eficaz e eficiente, por outro lado, fizeram surgir um conjunto rico de desafios que certamente gerarão relevantes trabalhos de pesquisa, alavancando a área de processamento de língua portuguesa.

Esta tese está organizada conforme o que segue. No Capítulo 2 é apresentada uma revisão bibliográfica sobre o estado da arte do aprendizado estatístico de gramáticas de língua natural que servem de base para sistemas de parsing. Em seguida, no Capítulo 3, é apresentada uma descrição detalhada do modelo de aprendizado probabilístico proposto por Collins, que é um modelo gerativo baseado na noção *head-centering*, em que o núcleo é o elemento principal de todo o processo de geração. No Capítulo 4 é descrita a arquitetura do sistema PAPO, apresentando o módulo gerador de modelos para PAPO-I e PAPO-II, e ainda, o mecanismo de busca (chart-parser) implementado. Em seguida, no Capítulo 5, são discutidos alguns resultados do comportamento do PAPO. Finalmente, no Capítulo 6, são feitas as conclusões e as considerações finais sobre o trabalho. Nos Apêndices A, B e C, são apresentados, respectivamente, o conjunto de etiquetas que anotaram o CETENFolha, uma revisão bibliográfica sobre avaliação de parsers e uma descrição sobre os fundamentos do chart parser.

Capítulo 2

Aprendizado Estatístico de Gramáticas de Língua Natural

2.1 Introdução

O aprendizado estatístico se insere num contexto cuja linha de pesquisa é chamada de empírica, uma vez que se baseia em exemplos já prontos e aprende como lidar com aqueles ainda não vistos. De acordo com Manning e Schütze (1999), a linha empiricista, que entre as décadas de 60 e 80 ficou nas sombras de crenças racionalistas encabeçadas por Chomsky (1965), cujo reflexo dentro da Inteligência Artificial caracterizava-se pela criação de sistemas inteligentes com grande quantidade de conhecimento inicial codificado à mão, ressurgiu na década de 90, com a idéia de que o conhecimento pode ser induzido a partir de algumas operações básicas de associação e generalização (Mitchell 1980). Assim, segundo o empiricismo, uma máquina poderia aprender a estrutura de uma linguagem apenas observando uma grande quantidade de exemplos, usando procedimentos estatísticos gerais e métodos de associação e generalização indutiva, como aprendizado indutivo de regras ((Brill 1993), (Hermjakob e Mooney 1997)) .

Segundo essa linha, a análise sintática automática (parsing) caracteriza-se por um processo de indução de uma função $f : S \rightarrow AS$, que mapeia um conjunto de possíveis entradas (sentenças S) num conjunto de possíveis saídas (suas respectivas árvores sintáticas AS). Essa função é induzida a partir de um conjunto de sentenças já analisadas que servem como o treinamento do modelo. O conjunto de dados (*treebank*) é formado por n pares *entrada* (S_i) - *saída* (AS_i), em que cada árvore AS_i é obtida através da aplicação da função sobre a sentença S_i, ou:

$$AS_i = f(S_i) \tag{2.1}$$

Ambigüidade

Um dos maiores obstáculos encontrados pelos sistemas automáticos de parsing (parsers) surge quando estes se deparam com sentenças que possuem algum tipo de ambigüidade sintática, ou seja, sentenças com duas ou mais árvores possíveis. Por exemplo, na sentença *A menina viu o menino de binóculo*, a atribuição dos sintagmas sintáticos pode mudar a interpretação que se faz da frase. Na Figura 2.1 são apresentadas duas árvores possíveis para a mesma sentença, nas quais, de acordo com a estrutura da primeira, a menina é que estava de binóculo e observou o menino, e na segunda, a menina observou o menino que estava de binóculo. Nesses casos, é necessário que o parser opte por uma delas, e que, de preferência, seja a que se esteja buscando.

Assim, para que o parser pudesse fazer a atribuição procurada, ele precisaria de uma certa interpretação de significado que o ajudasse a fazer a escolha correta. No entanto, tais sistemas são totalmente desprovidos de quaisquer informações nesse sentido. Muitos concordam que essa não é uma função do parser, delegando tal responsabilidade a uma unidade especial de desambiguação.

Os parsers estatísticos utilizam medidas de probabilidades observadas em sentenças previamente analisadas como critério de desempate em prováveis ações de desambiguação. Portanto, para que funcione, é necessário que se tenha um conjunto bastante representativo de sentenças com suas respectivas árvores sintáticas. Desse modo, o parser atribuirá probabilidades às possíveis análises de uma sentença, apresentando como resposta aquela de maior probabilidade, sendo isso feito em três passos: (1) encontra todas as possíveis análises; (2) atribui-lhes probabilidades, e (3) seleciona a de mais alta probabilidade.

Parametrização no aprendizado estatístico

O processo de parsing estatístico é separado em dois componentes: o *modelo*, que é a função que define o espaço de eventos e os parâmetros a serem associados a cada evento; e o *parser* propriamente dito, que é um algoritmo que implementa uma busca da melhor árvore sintática (AS_{best}) para qualquer sentença S de entrada.

Nesse cenário, se o espaço de eventos possíveis for associado aos mapeamentos entre as sentenças (S) e suas respectivas árvores sintáticas (AS), o aprendizado passa a ser a indução de uma função probabilística

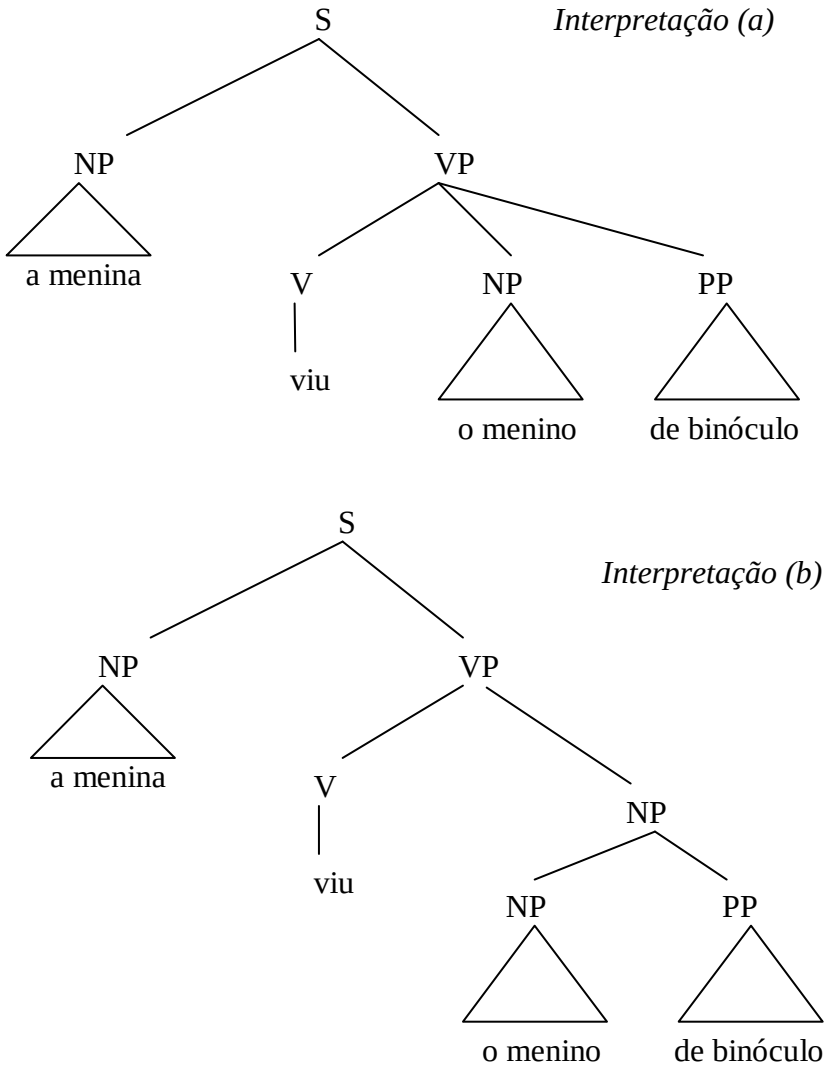

Figura 2.1: Exemplo de ambigüidade na sentença *A menina viu o menino de binóculo*

$$Score : AS \times S \rightarrow [0, 1] \tag{2.2}$$

em que $Score(AS, S)$ pode ser tanto uma probabilidade conjunta, em que o evento S aparece conjuntamente com o evento AS, representado por $P(AS, S)$, quanto uma probabilidade condicional $P(AS|S)$, em que a ocorrência do evento AS é condicionada à ocorrência do evento S. $f(S)$ pode ser definida como o membro mais provável de AS sob uma probabilidade, chamada de AS_{best}, de distribuição:

$$AS_{best}(S) = argmax < S \rightarrow Score(AS, S) > \tag{2.3}$$

Tendo em vista que, no parsing, o conjunto de sentenças vistas no teste quase nunca é visto no treinamento, é impossível estimar parâmetros baseados em árvores completas. Para que o modelo tenha um número tratável de parâmetros, estes devem ser associados a subestruturas. A parametrização consiste, então, na quebra do par (AS, S) em um conjunto de "eventos" $<Evento_1...Evento_n>$. O $Score$ para uma estrutura inteira passa a ser calculado como o produto de probabilidades, uma para cada subestrutura:

$$Score(AS, S) = \prod_{i=1..n} Score(Evento_i) \tag{2.4}$$

A escolha dessa parametrização é central para o sucesso do modelo de parsing. Uma boa parametrização deve representar bem os dados, e ainda concisamente, de modo que siga os seguintes critérios, segundo Collins (1999):

1. Poder discriminativo - deve incluir informação contextual requerida para decisões de desambiguação. Os modelos mais simples falham nesse sentido por serem insensíveis a informações lexicais e preferências estruturais (Por exemplo, as preferências estruturais de ligações entre os sintagmas dependem de informações lexicais, como as palavras que estão sendo ligadas, para serem realizadas.);

2. Poder de compactação - o modelo deve ter o menor número de parâmetros possível. O número de parâmetros no modelo determina a quantidade de dados de treinamento requeridos para treiná-lo. Isso significa que o poder de compactação do modelo determinará o quão perto ele chega do seu máximo, dada uma quantidade limitada de dados para treinamento. Como ilustração, considere o modelo que atribui probabilidades a árvores inteiras, cada árvore representada como um evento

único. Esse modelo tem um enorme poder discriminativo, sendo capaz de modelar propriedades arbitrárias das árvores. Mas, falha pelo critério de compactação, sendo quase improvável que se tenha dados suficientes para treinar um modelo sob esses parâmetros.

O modelo parametrizável conta com um vetor de parâmetros Θ como um argumento adicional para a função $Score$, escrita então como, $Score(AS, S|\Theta)$. Nesse contexto, a tarefa de modelagem é dividida em 3 etapas:

- Definição do modelo de estrutura: a função $Score(AS, S|\Theta)$.
- Definição de um método para estimar os parâmetros, isto é, uma função de exemplos de treinamento, $<S_1, AS_1>...<S_n, AS_n>$, para estimar os parâmetros Θ.
- Definição de um método de busca: um algorimo que, para cada entrada S, encontrará a melhor saída $AS_{best}(S) = argmax < S \rightarrow Score(AS, S|\Theta) >$.

Num modelo simples, a probabilidade $Score(AS, S|\Theta)$ é igual à probabilidade de AS tendo S e Θ, $P(AS|S, \Theta)$, na qual o parâmetro Θ lista uma probabilidade para cada membro do conjunto $S \times AS$. Assim, Θ consiste de um vetor com $|AS| \times |S|$ elementos. Os parâmetros do modelo são estimados usando a estimativa ML (*Maximum Likelihood*) (Manning e Schütze 1999), que dá a estimativa de Θ tal que:

$$Score(AS, S|\hat{\Theta}) = P(AS|S, \hat{\Theta}) = \frac{Cont(AS, S)}{Cont(S)} \tag{2.5}$$

em que $Cont(S)$ é a freqüência com a qual a sentença S é vista no treinamento, e $Cont(AS, S)$ é a freqüência com a qual a sentença S é vista no treinamento associada à árvore AS.

Nesse modelo, encontrar o AS_{best} para uma entrada S consiste em procurar sua árvore mais freqüente, isto é, definindo

$$f(x) = argmax_{as \in AS} Cont(AS, S) \tag{2.6}$$

que é a função que procura a árvore AS mais próvavel para a sentença S.

Uma fraqueza desse modelo é o fato de existir um grande número de parâmetros ($|AS| \times |S|$): o modelo pressupõe que uma entrada S, quando usada nos testes, terá sido

vista pelo menos uma vez nos dados do treinamento[1]. Surgem, então, duas questões com relação à parametrização, ainda segundo Collins (1999):

1. A quais objetos lingüísticos os parâmetros do modelo podem ser associados?
2. Como isso pode ser instanciado de uma forma que seja tratável?

Charniak (1997) apresenta um exemplo em seu modelo baseado em Gramáticas Livres de Contexto Probalilísticas (GLC-P) em que associa os parâmetros às regras de produção que formam as árvores sintáticas. Proposta originalmente por Booth e Thompson (1973), uma gramática probabilística é tão somente uma gramática livre de contexto em que cada regra gramatical tem uma probabilidade associada. Se o lado esquerdo da regra for mantido fixo, e as probabilidades sobre os diferentes lados direitos forem somadas, o resultado deve ser 1, uma vez que as probabilidades são associadas aos símbolos do lado esquerdo da regra.

Um problema que surge é que não há dados suficientes para estimar de uma maneira confiável as probabilidades das regras vistas no treinamento, deixando de alcançar qualquer robustez para regras não vistas. Isto inspirou David Magerman e Michael Collins a gerarem os lados direitos das regras dinamicamente. Fundamentando um modelo referido como "Gramática Baseada na História da Análise"[2] (Black e et al. 1992), Magerman (1995) introduziu a idéia de que uma árvore sintática é construída por uma seqüência de ações de derivações generalizadas, sendo a probabilidade da análise obtida a partir da totalidade das probabilidades das derivações. Assim, na montagem dos lados direitos das regras, cada sintagma-filho recebe a função que lhe corresponde. Essa função corresponde à posição (direita, esquerda, central ou única) e ainda à indicação de núcleo sintático, se for o caso. Surge aí, incorporada ao modelo estatístico, a noção de núcleo sintático; a idéia é propagar para cima o núcleo lexical para determinar fronteiras de sintagmas e preferências por ligações entre as palavras.

Collins, em seus trabalhos ((Collins 1996), (Collins 1997), (Collins 2000)) seguiu essas idéias e adicionou um pouco mais de elegância ao esquema, gerando o filho-núcleo primeiro, e então o resto dos filhos, num processo de Markov de segunda ordem, à esquerda e à direita dele. É interessante notar que, apesar de o condicionamento das probabilidades ser de cima para baixo na árvore, a análise é feita de baixo para cima, tal como nas GLC-Ps. Isso permite que as probabilidades sejam condicionadas a cadeias de palavras

[1] As entradas não vistas têm $Cont(AS, S) = 0$ e $P(AS|S, \hat{\Theta})$ é indefinido.
[2] *History-Based Grammar.*

dominadas pelo sintagma, feito em termos de distância entre o núcleo e o sintagma sendo gerado. Isso torna possível deixar indicadores de fronteiras de sintagmas, tal como pontuações, e dá ao modelo a chance de inferir preferências por associações (à direita, por exemplo). Collins também incorporou a noção de complementos lexicais e *wh-movement* em seu modelo. O primeiro é feito "roubando-se" complementos de uma lista hipotética como uma cadeia de Markov de irmãos do núcleo e gerando-se os elementos na ordem contida nessa lista. O modelo aprende as probabilidades por essas ações bastante sofisticadas de derivação sob várias condições.

Como acima citado, os métodos estatísticos sofreram evolução motivados basicamente pela necessidade de se considerar o contexto no processo de análise e também pela necessidade de uso de informação lexical na decisão de ligação entre palavras. Essa evolução surgiu naturalmente, conforme mostrado a seguir, iniciada nos modelos de Markov, seguida das Gramáticas Livres de Contexto Probabilísticas e dos Métodos Baseados na História da Análise. As seções seguintes têm por objetivo mostrar essa evolução, apresentando alguns dos modelos já aplicados para problemas de processamento de língua.

2.2 Modelo n-gramas

O modelo n-gramas é um dos modelos menos sofisticados, mas muito usado, principalmente para etiquetagem morfossintática. A modelagem n-gramas parte do pressuposto de que apenas as últimas $n-1$ palavras, para algum $n > 0$, fixo, numa seqüência, é que são importantes e, conseqüentemente, produzem algum efeito na probabilidade da próxima palavra (p), sendo esta condicionada na forma $P(p_n|p_1, ..., p_{n-1})$. Nos casos mais gerais, o número n é três, daí o nome do modelo tão famoso chamado *trigramas*. A equação que calcula a probabilidade da próxima palavra condicionada a n-gramas é reduzida em trigramas, tal como mostrado abaixo:

$$P(p_n|p_1, ..., p_{n-1}) = P(p_n|p_{n-2}, p_{n-1}) \tag{2.7}$$

E o modelo estatístico, para uma seqüência $p_{1,n}$, que vai da palavra p_1 até a palavra p_n é então calculado como:

$$P(p_{1,n}) = P(p_1)P(p_2|p_1)P(p_3|p_{1,2})...P(p_n|p_{1,n-1})$$

$$= P(p_1)P(p_2|p_1)P(p_3|p_{1,2})...P(p_n|p_{n-2,n-1})$$

$$= P(p_1)P(p_2|p_1)\prod_{i=3}^{n} P(p_i|p_{i-2,i-1}) \tag{2.8}$$

Charniak (1993) assume a existência de duas "pseudo-palavras" que ele chama de p_{-1} e p_0, e, segundo ele, poderiam ser encaradas como indicadores de começo e fim de texto e incorporadas ao modelo. Assim, a Equação 2.8 pode ser simplificada em

$$P(p_{1,n}) = \prod_{i=1}^{n} P(p_i|p_{i-2,i-1}) \tag{2.9}$$

O modelo, assim como proposto, tem as suas probabilidades estimadas baseadas em algumas contagens. Para estimar a probabilidade da palavra p_i, visto que são consideradas as duas palavras p_{i-2} e p_{i-1} anteriores, realiza-se a contagem no corpus de quantas vezes as três palavras foram vistas em co-ocorrência, em razão de quantas vezes foram vistas apenas as duas palavras anteriores a p_i, conforme a Equação 2.10:

$$\hat{P}(p_i|p_{i-2,i-1}) = \frac{Cont(p_{i-2,i})}{Cont(p_{i-2,i-1})} \tag{2.10}$$

A representação gráfica do modelo é uma cadeia de Markov, que nada mais é do que um autômato finito com probababilidades associadas aos elementos de transição. A Figura 2.2 ilustra um exemplo de cadeia de Markov para o modelo de trigrama da Equação 2.9, contendo apenas dois símbolos "a" e "b" (ignorando as pseudo-palavras). Cada arco no diagrama representa uma transição, indicando o símbolo, bem como a probabilidade associada àquele arco. A diferença para um modelo de Markov tradicional é que não é possível determinar o estado da máquina simplesmente com base na última saída. O estado é determinado com base nas duas últimas saídas (isto é, duas últimas palavras), sendo por isso chamada de cadeia de Markov de ordem 2.

O modelo tal como descrito parece perfeito, mas sofre com o problema da falta de dados. Segundo Jelinek (1990) observou, na coleta de estatísticas de trigramas de um corpus de 1.500.000 palavras, 25% dos trigramas ainda permanecem desconhecidos. A solução para resolver esse problema, conhecido como problema dos dados esparsos, é "suavizar" (*smooth*) as probabilidades usando bigramas e também unigramas, e, ao invés

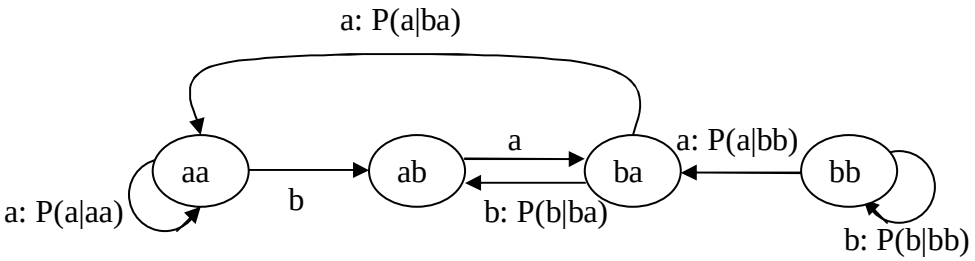

Figura 2.2: Cadeia de Markov para trigramas
(Charniak 1993)

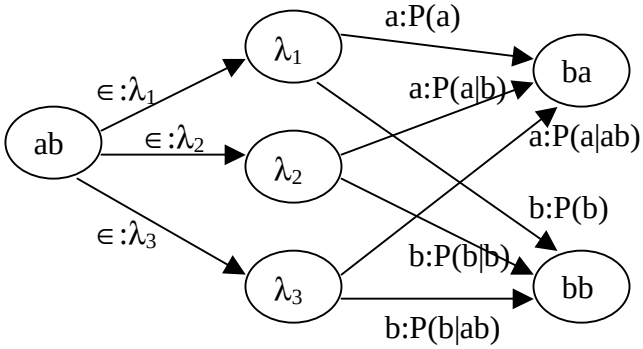

Figura 2.3: Modelo MOM
(Charniak 1993)

de usar a Equação 2.9, usar a Equação 2.11 abaixo:

$$P(p_i|p_{i-2,i-1}) = \lambda_1 \hat{P}(p_i) + \lambda_2 \hat{P}(p_i|p_{i-1}) + \lambda_3 \hat{P}(p_i|p_{i-2,i-1}) \tag{2.11}$$

na qual λ_1, λ_2, λ_3 são constantes não negativas, obtidas empiricamente, tal que sua soma seja 1 (por exemplo .1, .3 e .6).

2.2.1 Modelos Ocultos de Markov (Hidden Markov Models)

Os Modelos Ocultos de Markov (MOM) são uma generalização das cadeias de Markov, permitindo que um dado estado possa ter mais de uma transição saindo dele. Assim, segundo a Equação 2.11, a probabilidade atribuída a uma cadeia em um MOM é a soma de todos os caminhos possíveis através dele para aquela cadeia (vide Figura 2.3).

Um MOM é descrito formalmente como uma quádrupla $<e^1, E, P, A>$ na qual E é o conjunto de estados (e^1,e^2,...,e^m), $e^1 \in E$ é o estado inicial do modelo, P é um conjunto de símbolos de saída (p^1,p^2,...,p^n), e A um conjunto de arcos ou transições (a^1,a^2,...,$a^{\in}$).

Assim, uma transição que consome uma palavra p^k mudando de estado, é escrita como $e^i \rightarrow^{p^k} e^j$.

Como freqüentemente o que interessa são todos os estados possíveis que aquele MOM pode percorrer, supõe-se que o modelo tem transições para todas as saídas possíveis, indo para todos os possíveis próximos estados. As transições indesejadas podem receber probabilidade 0. Por exemplo, no caso da Figura 2.3, o estado a pode ser estado inicial para várias transições com símbolos iguais, mas estes vão para estados finais diferentes. Em casos como esse, não é possível saber qual é o próximo estado apenas olhando o símbolo consumido. É preciso conhecer a seqüência de estados seguida pelo MOM não dedutível a partir de sua entrada. Vem daí o nome *hidden*.

A probabilidade associada à transição $e^i \rightarrow^{p^k} e^j$, $P(e^i \rightarrow^{p^k} e^j)$, é definida como a probabilidade de que, num tempo t, o MOM produza como saída o $t_{ésimo}$ símbolo p^k e vá para o estado $t+1$, na seqüência de estados, e^j, dado o $t_{ésimo}$ estado e^i:

$$P(e^i \rightarrow^{p^k} e^j) =^{def} P(E_{t+1} = e^j, P_t = p^k | E_t = e^i) \quad 1 \leq t \tag{2.12}$$

ou, simplificando:

$$P(e^i \rightarrow^{p^k} e^j) = P(e^j, p^k | e^i) \tag{2.13}$$

Segundo a Equação 2.13, os modelos de Markov assumem que a única coisa que influencia a probabilidade de uma saída, ou seu próximo estado, é o estado anterior. Isso é chamado de *Suposição de Markov*. Ou:

$$P(p_n, e_{n+1} | p_{1,n-1}, e_{1,n}) = P(p_n, e_{n+1} | e_n) = P(e^i \rightarrow^{p^k} e^j) \tag{2.14}$$

Dada a Equação 2.14, pode-se mostrar formalmente como computar as probabilidades de seqüências usando MOM:

$$P(p_{1,n}) = \sum_{e_{1,n+1}} P(p_{1,n}, e_{1,n+1}) = \tag{2.15}$$

$$\sum_{e_{1,n+1}} P(p_1, e_2 | e_1) P(p_2, e_3 | e_2) ... P(p_n, e_{n+1} | e_n) \tag{2.16}$$

$$\sum_{e_{1,n+1}} \prod_{i=1}^{n} P(p_i, e_{i+1}|e_i) = \sum_{e_{1,n+1}} \prod_{i=1}^{n} P(e_i \rightarrow^{p_i} e_{i+1}) \tag{2.17}$$

2.2.2 Etiquetagem morfossintática (tagging)

O modelo MOM pode ser usado para atribuir etiquetas morfossintáticas a palavras numa sentença. A principal razão para usar tal modelo é a possibilidade de treinamento automático que ele proporciona. Nos trigramas, o que se deseja é encontrar parâmetros para melhor predizer as entradas. No etiquetador, os parâmetros que se quer melhorar são aqueles que atribuem as etiquetas morfossintáticas (tags), não necessariamente aqueles que atribuem probabilidades às entradas. Mas como adaptar o modelo para fazer isso? Os MOMs têm três componentes: saídas, transições e estados. As saídas são as palavras do corpus para treinar o MOM. A relação das etiquetas com a saída será da forma:

$$P(p_{1,n}) = \sum_{e_{1,n+1}} P(p_{1,n}, e_{1,n+1}) \tag{2.18}$$

na qual e_{n+1} é uma seqüência de $n+1$ etiquetas, ou seja, a percentagem de vezes que, ao se observar o corpus, as mesmas palavras aparecem com as mesmas etiquetas.

Assim, formalmente, uma solução para o problema de etiquetagem morfossintática pode se definir como:

$$e_n = argmax < e \rightarrow P(e_{1,n}|p_{1,n}) >= argmax \frac{P(p_{1,n}, e_{1,n})}{P(p_{1,n})} = argmax P(p_{1,n}, e_{1,n}) \tag{2.19}$$

Transformando essa equação num modelo MOM, obtém-se a relação que as etiquetas têm com os estados, com a suposição de que a probabilidade de uma palavra aparecendo em determinada posição, dado que sua etiqueta ocorre na mesma posição, é independente de todo o resto, e que a probabilidade de uma etiqueta vir depois é dependente somente da etiqueta anterior, isto é:

$$P(p_n|p_{1,n-1}, e_{1,n}) = P(p_n|e_n) \tag{2.20}$$

$$P(e_n|p_{1,n-1}, e_{1,n-1}) = P(e_n|e_{n-1}) \tag{2.21}$$

O modelo passa então a ter a equação:

$$P(p_{1,n}) = \sum_{e_{1,n+1}} P(p_{1,n}, e_{1,n+1}) \tag{2.22}$$

$$P(p_{1,n}) = \sum_{e_{1,n+1}} \prod_{i=1}^{n} P(p_i|e_i)P(e_{i+1}|e_i) \tag{2.23}$$

Se for assumido que cada estado representa duas etiquetas, a da palavra anterior e da atual, obtém-se o modelo de linguagem:

$$P(p_{1,n}) = \sum_{e_{1,n+1}} \prod_{i=1}^{n} P(p_i|e_i)P(e_{i+1}|e_i, e_{i-1}) \tag{2.24}$$

E, por exemplo, a estimativa das probabilidades na forma:

$$\hat{P}(e_{i+1}|e_i, e_{i-1}) = \frac{Cont(e_{i-1}, e_i, e_{i+1})}{Cont(e_{i-1}, e_i)} \tag{2.25}$$

2.3 Gramática Livre de Contexto Probabilística

O uso de técnicas estatísticas para o aprendizado de gramáticas foi inspirado no sucesso dessas técnicas para o processamento de fala (Charniak 1993). O modelo proposto em uma gramática livre de contexto probabilística (GLC-P) faz uma suposição de independência que considera a probabilidade de cada regra de sintagma independente de todos os outros sintagmas na sentença. A ordem de derivação não afeta o modelo. As probabilidades nas GLC-Ps, atribuídas às regras, são encaradas como a probabilidade do sintagma-pai usando tal regra, nos subelementos descritos, em comparação a todas as outras regras que expandem o mesmo sintagma. Por exemplo, a gramática probabilística da Tabela 2.1 gera as sentenças ambígüas da Figura 2.1[3]:

Dadas as probabilidades das regras individuais (note que a soma de todas as regras relativas a um não-terminal deve ser 1), a probabilidade de uma análise inteira é o produto das probabilidades de cada regra usada durante a análise. Isto é, se S é a sentença inteira, AS uma análise de S, $sint$ os sintagmas de AS, e $r(sint)$ a regra usada para expandir

[3] A descrição da gramática usa a seguinte notação: S, sentença; SN, sintagma nominal; SV, sintagma verbal e SP, sintagma preposicional;

Tabela 2.1: Gramática Probabilística

S → SN SV (1.0)	SN → determinante nome (0.8)
SN → SN SP (0.2)	SV → verbo SN SP (0.4)
SV → verbo SN (0.6)	SP → preposição nome (1.0)

sint, então:

$$P(AS, S) = \prod_{sint} P(r(sint)) \tag{2.26}$$

Ao se deparar com sentenças ambígüas, o parser pode escolher aquela cuja seqüência de aplicação das regras seja a de maior probabilidade, de acordo com o padrão de sentenças observadas na fase de treinamento. É na fase de treinamento que são recolhidas as probabilidades de cada regra.

As gramáticas probabilísticas têm muitas vantagens. Elas são extensões óbvias das gramáticas livres de contexto, muito conhecida dos pesquisadores da área. Além disso, os algoritmos usados para GLCs podem ser transportados para as GLC-Ps, permitindo que todas as possíveis análises possam ser encontradas num tempo de ordem n^3, em que n é o tamanho da sentença. Alguns autores, como Jelinek e Lafferty (1991), Stolcke (1995) e Goodman (1996a) apresentam alguns algoritmos usando as gramáticas probabilísticas. Um algoritmo bastante usado é o *Inside-Outside* (Charniak 1993). Usando um corpus grande e o algoritmo, o modelo pode ser treinado automaticamente de um modo completamente não supervisionado, considerando todas as análises possíveis da sentença no corpus de treinamento ((Baker 1975), (Kupiec 1991)), ou pode ser treinado por um modo restrito, maximizando a probabilidade das árvores sintáticas no corpus analisado ((Black, Lafferty, e Roukos 1992), (Pereira e Schabes 1992)).

Como obter uma gramática

Supondo que se queira usar uma GLC-P como mecanismo para um parser estatístico, e que se disponha de uma *treebank*. Para analisar uma sentença nova, são necessários:

1. Uma gramática na forma de GLC-P, tal que as novas sentenças possam ser analisadas de acordo com ela;

2. Um parser que aplica a GLC-P a uma sentença e encontra algumas ou todas as análises possíveis para a sentença;

3. A habilidade para encontrar as análises de mais alta probabilidade de acordo com a Equação 2.26.

Como descrito na seção anterior, existem algoritmos eficientes para as tarefas (2) e (3). Com isso, torna-se necessário cuidar apenas da tarefa (1). Há uma maneira trivial para resolver isso. A gramática pode ser facilmente obtida a partir da estrutura arbórea das sentenças previamente analisadas que compõem a *treebank*. Assim, numa subestrutura com pai S e filhos SN e SV, é possível montar a regra $S \rightarrow SN\ SV$. As probabilidades são calculadas em função de todas as regras que expandem nós não-terminais em comum, ou seja, pega-se todas as regras cujos pais são iguais, obtendo a partir daí as probabilidades de cada. Por exemplo, se forem observadas as regras $SN \rightarrow$ *determinante nome* e $SN \rightarrow$ *determinante nome SP*, e a regra *determinante nome* é usada 1.000 vezes, contra um total de 60.000 vezes que um SN em geral é usado, a probabilidade atribuída a essa regra é de $1.000/60.000 = .017$. Gramáticas baseadas em *treebank* desse estilo são avaliadas em (Charniak 1996), e uma generalização delas é usada em (Bod 1993). Os resultados obtidos com o seu uso chegam em média aos 75% de confiabilidade.

Um modelo sensível ao contexto

No começo da década de 90, os parsers construídos a partir de gramáticas livres de contexto probabilísticas supunham que as aplicações das regras eram completamente independentes do contexto no qual eram aplicadas. Se, por um lado, essa suposição de independência simplifica a estimativa dos modelos GLC-P, por outro lado, ela não é apropriada. A aplicação das regras não é independente de seu contexto, e por isso os modelos que levam em conta informação contextual fazem uma melhor análise da linguagem.

Uma primeira tentativa de incorporar contexto foi na construção de um modelo de parsing chamado Pearl (Magerman e Marcus 1991). Baseado nas gramáticas probabilísticas, Pearl adiciona ao modelo informações morfossintáticas de contexto quando atribui uma probabilidade à aplicação de uma regra. O modelo estima, também através do algoritmo *Inside-Outside*, a probabilidade $P(AS|S)$ de cada árvore AS, dadas as palavras na sentença S, assumindo que cada não-terminal e seus filhos imediatos são dependentes dos filhos e pais não-terminais e do trigrama de etiquetas centrados no começo de cada regra:

$$P(AS|S) = \prod_{A \in AS} P(A \rightarrow \alpha | C \rightarrow \beta A \gamma, e_0 e_1 e_2) \tag{2.27}$$

em que C é o não-terminal que imediatamente domina A, e_1 é a etiqueta associada à

palavra mais à esquerda do sintagma A, e e_0 e e_2 são as etiquetas das palavras à esquerda e à direita de e_1, respectivamente.

Treinado com 1.100 sentenças e testado com 40 sentenças, Pearl conseguiu uma precisão geral de 88%, segundo os autores. No entanto, foi treinado usando pouca quantidade de dados. Além disso era um modelo bastante simples apesar de considerar mais informações estruturais de contexto que as GLC-Ps, e também ignorar itens lexicais, ou seja, as palavras propriamente ditas.

2.4 Modelo Baseado na História da Análise

Apesar de produzirem bons resultados, os modelos descritos nas seções anteriores não conseguem capturar informações suficientes para lidar com um dos maiores problemas do parsing, a ambigüidade. Os modelos MOM só demonstraram poder para lidar com ambigüidade lexical, o que não é suficiente para desambiguação. As GLC-Ps tradicionais, por outro lado, já contam com uma certa noção de contexto, mas ainda restrita a características locais. Em problemas como no parsing, no qual a informação lexical de longa distância é crucial para desambiguação, modelos locais como as GLC-Ps e os modelos n-gramas são insuficientes. Com a proposta das gramáticas baseadas na história da análise, Black e et al. (1992) apresentaram um modelo provido de mecanismos para contar com informação contextual em qualquer posição da história do discurso, aumentando-se assim a sensibilidade estrutural do modelo.

O Modelo Baseado na História da Análise (MBHA) é um modelo gerativo probabilístico que usa a vantagem de possuir informação lingüística detalhada para resolver ambigüidade. Os autores abordam o sentido de contexto da forma mais fiel possível com o que nós humanos consideramos, racionalizando a quantidade de informação da sentença que é necessária e suficiente para determinar sua análise[4].

O projeto de um MBHA envolve três passos:

1. Representação - escolha de como representar as árvores: etiquetas sintáticas, etiquetas morfossintáticas, presença de núcleos lexicais associados, representação de palavras ou suas canônicas, etc.

[4]Black et al. usam uma árvore de decisão para determinar que aspectos da derivação, chamada de seqüências de decisões, têm influência na probabilidade de um determinado nó ser expandido. Isso permite que o modelo use qualquer informação em qualquer lugar na derivação parcial da árvore para determinar as probabilidades de expansão de um nó.

2. Decomposição - definição de um mapeamento um-para-um entre árvores AS e seqüências de decisão $< d_1..d_n >$. O modelo define uma probabilidade conjunta $P(AS, S)$ sobre todos os pares (AS, S) ou uma probabilidade condicional $P(AS|S)$ sobre todas as árvores candidatas para uma sentença particular. Dado um mapeamento entre árvores e seqüências de decisão, a probabilidade de uma árvore pode ser descrita como o produto das probabilidades de cada decisão, seguindo as fórmulas abaixo:

$$P(AS|S) = \prod_{i=1..n} P(d_i|d_1..d_{i-1}, S) \tag{2.28}$$

ou

$$P(AS, S) = \prod_{i=1..n} P(d_i|d_1..d_{i-1}) \tag{2.29}$$

Na probabilidade condicional, uma árvore é associada a uma seqüência de decisões tomadas por um parser ao recuperá-la. Nos modelos conjuntos, as decisões usualmente são passos em alguma derivação de cima para baixo da árvore, por exemplo, a seqüência de produções usadas numa derivação mais à esquerda de uma gramática livre de contexto.

3. Suposições de Independência - envolve a definição de uma função, Φ, que agrupa seqüência de decisões em classes equivalentes, reduzindo o número de parâmetros a proporções manuseáveis. O modelo final é da forma:

$$P(AS|S) = \prod_{i=1..n} P(d_i|\Phi(d_1..d_{i-1}, S)) \tag{2.30}$$

A escolha de Φ pode ser feita manualmente, ou automaticamente, usando uma técnica de aprendizado de máquina tal como árvore de decisão.

A distinção com a GLC-P ocorre no sentido de que cada estrutura de sintagma depende não somente da entrada, mas também da história inteira naquele ponto da sentença. A história é interpretada como qualquer elemento da árvore que já foi determinado, incluindo palavras anteriores, não-terminais, estruturas de sintagmas, e qualquer outra informação lingüística que é gerada como parte da estrutura da análise.

O modelo é capaz de lidar com qualquer formalismo de gramática que tenha um processo de derivação. Para o experimento feito em (Black e et al. 1992), os autores

definiram cerca de 50 categorias sintáticas (Syn) e cerca de 50 categorias semânticas (Sem). Cada não terminal tem um núcleo lexical primário NUC_1, que corresponde à noção lingüística de núcleo, e um núcleo lexical secundário NUC_2, que é apenas uma palavra que contém informação preditiva sobre o sintagma. Quando uma regra é aplicada a um não-terminal, ela indica qual filho gerará o núcleo primário e qual filho gerará o núcleo secundário.

No modelo MBHA, a predição das etiquetas sintática e semântica de cada sintagma é feita usando a etiqueta de seu pai (Syn_{pai}, Sem_{pai}), a etiqueta de seu núcleo primário e secundário, a regra na qual seu pai o deriva (R_{pai}) e seu índice dentro daquela regra (I). Tudo isso é feito de acordo com 5 fatores:

1. $P(Syn|R_{pai}, I, NUC_{1pai}, NUC_{2pai}, Syn_{pai}, Sem_{pai})$, que estima a probabilidade de o sintagma sendo gerado ter rótulo sintático Syn, dadas as informações de seu pai como etiquetas sintática e semântica, a regra, os núcleos primário e secundário, e o índice do sintagma na regra;

2. $P(Sem|Syn, R_{pai}, I, NUC_{1pai}, NUC_{2pai}, Syn_{pai}, Sem_{pai})$, que estima a probabilidade de o sintagma ter rótulo semântico Sem, considerando as informações de seu pai e do rótulo sintático Syn, previamente gerado;

3. $P(R|Syn, Sem, R_{pai}, I, NUC_{1pai}, NUC_{2pai}, Syn_{pai}, Sem_{pai})$, que estima a probabilidade de o sintagma ter regra de reescrita R, dadas as informações de seu pai e as etiquetas sintática e semântica geradas previamente;

4. $P(NUC_1|R, Syn, Sem, R_{pai}, I, NUC_{1pai}, NUC_{2pai})$, que estima a probabilidade de o sintagma ter núcleo principal NUC_1, dadas as informações de seu pai, juntamente com as informações geradas previamente;

5. $P(NUC_2|NUC_1, R, Syn, Sem, R_{pai}, I, Syn_{pai})$, que estima a probabilidade de o sintagma ter núcleo secundário NUC_2, dadas as informações de seu pai e as suas, geradas previamente;

A Figura 2.4 mostra um exemplo de representação de um sintagma preposicional "with a list" em MBHA.

Dois modelos bastante referenciados na literatura são exemplos de uso dos MBHAs: o SPATTER (Magerman 1994) e os métodos de Collins (Collins 1999). A seguir, esses modelos são descritos de uma forma mais detalhada.

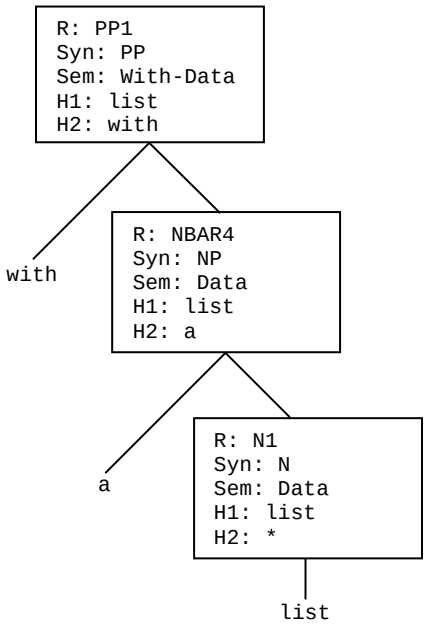

Figura 2.4: Exemplo de uma árvore usando MBHA.
(Magerman 1995)

2.4.1 O Parser SPATTER

Magerman (1994) apresenta com o SPATTER um modelo de probabilidade condicional que aborda o problema da descoberta automática de critérios de desambiguação para todas as decisões tomadas durante um processo de parsing. Ele propõe um algoritmo de classificação que pode aprender, através de coletas de informações estatísticas de um corpus, esses diferentes critérios pelos quais as decisões de desambiguação são tomadas.

O parser usa modelos probabilísticos para predizer as etiquetas morfossintáticas e as etiquetas sintáticas dos sintagmas para uma dada sentença. Ele trabalha, no sentido de baixo para cima, através de uma seqüência de ações que montam a estrutura sintática da sentença. Cada característica de decisão é modelada por uma árvore de decisão estatística, refinada através de aplicações do algoritmo *Expectation-maximization (E-M)*, estimando a probabilidade de cada alternativa dado o seu contexto. Uma vez que cada decisão é condicionada ao contexto inteiro daquela análise parcial, a ordem na qual as decisões são tomadas afeta a probabilidade da análise, que é a soma sobre todas as derivações possíveis daquela árvore, e a probabilidade de uma derivação é o produto das probabilidades das decisões atômicas que resultaram na construção da árvore sintática.

Através da implementação baseada no algoritmo CART (Breiman, Friedman, Olshen, e Stone 1984), o SPATTER mostra que uma tecnologia de árvore de decisão pode

gerar modelos que seletivamente escolhem elementos do contexto que contribuem para decisões de desambiguação e têm somente os parâmetros necessários, podendo com isso, serem implementados com recursos já existentes.

A árvore sintática é codificada em termos de quatro traços (*features*): palavras, etiquetas morfossintáticas (*etiqMorf*), etiquetas sintáticas (*etiqSint*) e extensões, com vocabulários fixos e representação única. O primeiro passo é a atribuição do traço *palavra*, definindo um mapeamento de uma seqüência de itens lexicais obtidos da sentença em um vocabulário fixo de palavras, incluindo a palavra *desconhecida*. Em seguida é atribuído o traço *etiqMorf* para cada palavra na sentença. O vocabulário de etiquetas morfossintáticas é o mesmo visto nos dados de treinamento. O próximo passo é atribuir a cada uma dessas palavras com suas respectivas etiquetas morfossintáticas, localizadas na folha da árvore, uma extensão (direção), que pode ser direita, esquerda, meio, ou unária, na qual esquerda corresponde ao nó inicial de um sintagma, direita ao nó final de um sintagma, meio, aos nós que estão entre os dois, e unário, significa que o nó é filho único. A partir de valores consecutivos de extensão dos nós, os sintagmas são formados. Quando o parser detecta a seqüência de nós que correspondem a um sintagma, ele gera um novo nó pai e recomeça o processo de atribuição de traços àquele nó (a primeira é a etiqueta sintática, que codifica informação sobre o sintagma abaixo daquele nó).

Os nós internos da árvore terão a palavra e a etiqueta morfossintática atribuídas deterministicamente, e etiqueta sintática e extensão, preditas pelo modelo. Cada nó tem ainda uma palavra associada, chamada de representativa, que corresponde à noção lingüística de núcleo. A palavra representativa é escolhida através de regras determinísticas, chamadas *Tree Head Table*, baseada na etiqueta sintática do sintagma e nas etiquetas morfossintática e sintática dos elementos desse sintagma.

Um quinto traço é usado para representar coordenação: trata-se de um bit a mais, sendo que, se tiver valor verdadeiro, significa que o nó é parte de um sintagma com coordenação.

As decisões são tomadas de uma forma não determinística com base na probabilidade de cada escolha. Estas são estimadas usando modelos de árvore de decisão estatísticas, em que a probabilidade de uma árvore sintática completa (AS) de uma sentença (S) passa a ser o produto da probabilidade de cada decisão di condicionada a todas as decisões anteriores:

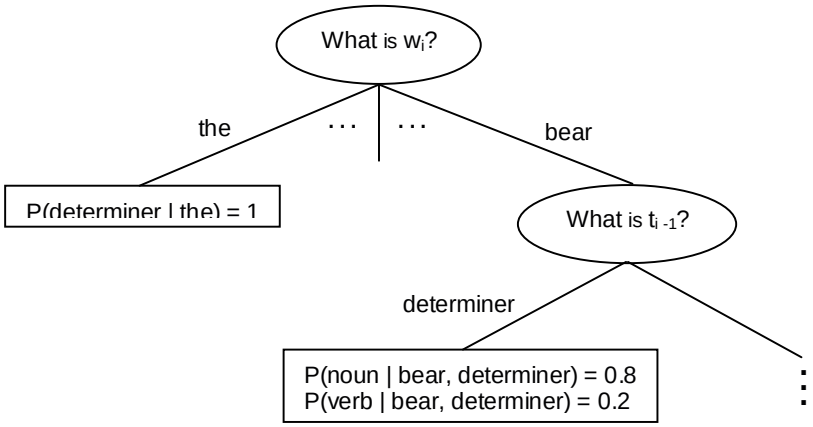

Figura 2.5: Exemplo de árvore de decisão
(Magerman 1995)

$$P(AS|S) = \prod_{d_i \in AS} P(d_i|d_{i-1}d_{i-2}...d_1, S) \tag{2.31}$$

A árvore sintática escolhida é aquela cuja seqüência de decisões produz a maior probabilidade cumulativa (vide Figura 2.5). As decisões em consideração envolvem a identificação dos sintagmas e suas etiquetas em sentenças de língua natural.

Num processo geral, são três os objetos que descrevem uma árvore de decisão:

- História: codifica todas as informações necessárias para tomar uma decisão. Inclui qualquer aspecto relevante, como uma árvore parcial, com informações léxicas, sintáticas, número de nós, relacionamentos entre nós, etc.;
- Questões: e suas possíveis respostas, codificam conhecimento heurístico sobre o que é importante sobre a história. A cada nó interno, são associadas uma questão e sua resposta, que é um valor finito, e gera-se um novo nó na árvore de decisão;
- Futuro: refere-se a uma das possíveis escolhas que a árvore pode fazer. O conjunto de escolhas é o *vocabulário futuro*.

Assim, após a identificação dos traços que são relevantes para cada decisão e a atribuição das probabilidades para as possíveis escolhas, o parser trabalha de baixo para cima até que uma árvore mínima seja construída, e a partir daí seleciona a melhor segundo os valores dos traços relevantes.

Modelo de Parsing do SPATTER

Magerman argumenta que o SPATTER é um "modelo de parsing", pois atribui probabilidade a uma árvore sintática AS dada a sentença S, $P(AS|S)$. Essa nomenclatura é para diferenciar do que ele chama de "modelo gerativo", que é aquele que atribui probabilidade a uma árvore sintática e uma sentença, $P(AS, S)$.

A distribuição de probabilidade para cada valor de traço é estimada usando modelos condicionais na forma de árvores de decisão estatísticas. Os modelos estatísticos no SPATTER podem lançar mão de informações sobre todas as atribuições de traços feita nos passos anteriores. Devido a restrições computacionais, como tempo de processamento e explosão da árvore de decisão, por exemplo, os modelos estão limitados a dois sintagmas à esquerda, dois à direita do nó atual, e quatro filhos desse nó.

Em seu trabalho, as árvores de decisão são aplicadas a um número diferente de problemas de tomada de decisão: atribuição de etiquetas morfossintáticas a palavras, atribuição de etiquetas sintáticas aos sintagmas, determinando suas fronteiras, decisão sobre o escopo das conjunções, etc.

Os modelos condicionais de árvores de decisão usados são: o modelo *etiqMorf*, o modelo *etiqSint*, o modelo extensão e o modelo coordenação, descritos a seguir:

- Modelo *etiqMorf*

  A predição do valor do traço etiqueta morfossintática é condicionada a duas palavras à esquerda, duas à direita, e todas as suas informações:

  $$P(e_i|p_i p_{i-1} p_{i-2} p_{i+1} p_{i+2} e_{i-1} e_{i-2} e_{i+1} e_{i+2} N^{k-1} N^{k-2} N^{k+1} N^{k+2}) \tag{2.32}$$

  em que e é a etiqueta morfossintática, p_i é a i-ésima palavra, N^k o nó sintagma atual e N^{k-i} o i-ésimo sintagma anterior. No exemplo da Figura 2.6, ao observarmos a atribuição da etiqueta morfossintática para a palavra *is*, temos, portanto, p_i = is, p_{i-1} = computer, p_{i-2} = the, p_{i+1} = listed, p_{i+2} = NULL (não existe), e_{i-1} = NN1, e_{i-2} = AT, N^{k-1} = by the computer, N^{k-2} = used.

  A árvore de decisão pode também levantar questões diversas sobre qualquer um dos nós na janela dos nós que considera, como por exemplo, presença de marcas de pontuação.

- Modelo *etiqSint*

  A predição do valor do traço etiqueta sintática é condicionada a todas as informações dos dois nós à direita e dos dois nós à esquerda do nó atual, e informações gerais

sobre esses nós.

$$P(N_l^k|N^{k-1}N^{k-2}N^{k+1}N^{k+2}N^{c1}N^{c2}N^{c-1}N^{c-2}) \tag{2.33}$$

em que N^{c1} e N^{c2} representam as informações sobre os filhos do nó atual.

- Modelo Extensão

  A predição do valor do traço extensão é condicionada à informação do nó sendo estendido, e também todas as informações dos dois nós à esquerda e dois nós à direita, e os dois filhos mais à esquerda e os dois mais à direita do nó atual.

$$P(N_e^k|N_p^k N_t^k N_l^k N_c^k N^{k-1}N^{k-2}N^{k+1}N^{k+2}N^{c1}N^{c2}N^{c-1}N^{c-2}) \tag{2.34}$$

- Modelo Coordenação

  A predição do valor do traço coordenação é condicionada às mesmas informações que do traço extensão.

$$P(N_c^k|N_p^k N_t^k N_l^k N^{-1}N^{-2}N^{1}N^{2}N^{c1}N^{c2}N^{c-1}N^{c-2}) \tag{2.35}$$

O modelo geral é definido em termos das aproximações anteriores. A probabilidade de uma árvore sintática, dada a sentença, é a soma sobre todas das derivações daquela árvore:

$$P(AS|S) = \sum_d P(AS, d|S) \tag{2.36}$$

E a probabilidade de uma derivação de uma árvore sintática é o produto de cada uma das atribuições dos valores dos traços naquela derivação e a probabilidade de cada seleção de nó ativo naquela derivação:

$$P(AS, d|S) = \prod_{N \in AS, j<|d|} P(ativo = N|contexto(d_j))P(N_x|contexto(d_j)) \tag{2.37}$$

em que x varia sobre os valores de traços preditos em um nó.

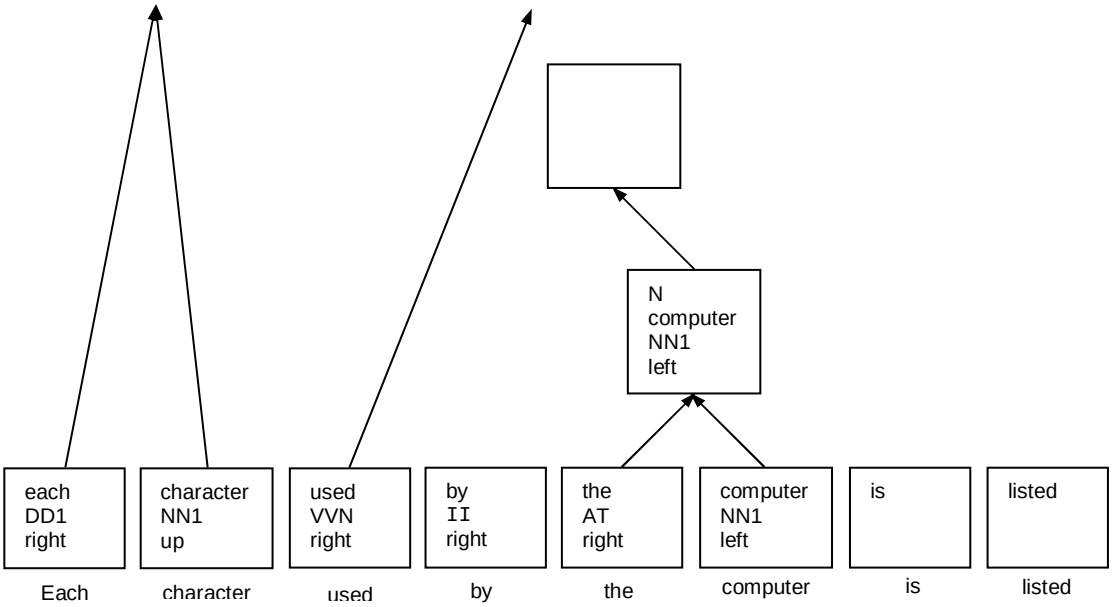

Figura 2.6: Exemplo de construção de uma árvore sintática no SPATTER (Magerman 1995)

Treinamento do SPATTER

Inicialmente, no treinamento, supõe-se que os modelos são uniformes: cada evento em cada derivação contribui igualmente para o processo que seleciona questionamentos sobre a história, de forma a predizer cada valor de traço. No entanto, não são. Os eventos gerados pelas melhores derivações deveriam contribuir mais para o processo de construção da árvore de decisão. O algoritmo *Forward-Backward* ((Poritz 1988), (Charniak 1993)) é responsável por localizar no treinamento inicial cada evento que pode contribuir para a construção de um novo conjunto de árvores de decisão, de forma que os eventos que contribuem mais, passam a ter pesos maiores, num processo de refinamento até que se encontre um modelo que melhor representa os dados.

O algoritmo *Forward-Backward*, que é um tipo especial do algoritmo *E-M*, reestima os parâmetros de um modelo P_i para gerar um novo modelo P_{i+1} que se garante ter probabilidades mais altas para o corpus de treinamento do que o modelo original, $P_{i+1}(C) \geq P_i(C)$. Ou seja, o algoritmo parte do pressuposto de que, quando há múltiplos caminhos de um estado inicial para um estado final, os diferentes caminhos deveriam contribuir para o modelo de acordo com suas probabilidades relativas. No algoritmo, cada derivação é vista como um caminho entre um estado inicial comum, as palavras na sentença, para um estado final comum, a árvore sintática completa. Se consideramos s_h um estado que precede s e uma função $f(s_h, s)$ como a atribuição do valor do traço que faz

passar do estado s_h para o estado s, temos que a probabilidade do estado s é computada por

$$P(s) = \sum_{sh} P(s_h)P(f(s_h, s)|s_h) \tag{2.38}$$

Esse passo é chamado de *forward*, e a probabilidade $P(s)$ é referida como $\alpha(s)$. A probabilidade *backward* do estado, referido como $\beta(s)$, é calculada de acordo com a fórmula recursiva:

$$\beta(s) = \sum_{s} \beta(s)P(f(s_h, s)|s_h) \tag{2.39}$$

em que a probabilidade *backward* do estado final é igual à probabilidade *forward* do estado final, $\beta(s_{final}) = \alpha(s_{final})$. A contagem associada a cada atribuição de valor de traço, $f(s_h, s)$ é:

$$Cont(f(s_h, s)) = \frac{\beta(s)\alpha(s_h)P(f(s_h, s)|s_h)}{\alpha(s_{final})} \tag{2.40}$$

O valor $Cont(f(s_h, s))$ é a contribuição do evento $f(s_h, s)$ para a distribuição que prediz o valor do traço f dada a história h. A reestimativa da probabilidade de uma atribuição de um valor de traço, dada uma história, é dada pela razão entre o total de contagens para aquela atribuição sobre todas as atribuições dada aquela história, segundo a Equação 2.41:

$$P_{novo}(f|h) \approx \frac{Cont(f(s_h, s)}{\sum_{s'} Cont(f(s_h, s'))} \tag{2.41}$$

Algoritmo da árvore de decisão

Cada folha na árvore de decisão representa a distribuição de uma classe de histórias. Os parâmetros dessa distribuição são atualizados usando o algoritmo *Forward - Backward* descrito acima. Assim, a seqüência de treinamento dos modelos de árvores de decisão segue os passos:

1. Construa as árvores de decisão iniciais (M_1) baseadas nos modelos uniformes

2. Crie M_2 podando árvores em M_1 numa profundidade máxima de 10

3. Construa árvores de decisão (M_3) a partir das contagens *Forward-Backward* de M_2
4. Faça a reestimativa *Forward-Backward* das folhas das árvores de decisão em M_3

Resultados obtidos

Aplicado a sentenças do *Wall Street Journal* WSJ (Marcus e et al. 1993), o trabalho avançou quanto aos seguintes aspectos: tratou sentenças grandes (maiores de 40 palavras) e em domínios irrestritos (jornal); foi treinado automaticamente a partir da gramática; foi o primeiro a introduzir o uso de itens lexicais em seu processamento; teve uma precisão de 84%, contra os 74% das gramáticas probabilísticas não lexicalizadas de Charniak (Charniak 1997).

2.4.2 Modelos de Collins

Segundo Collins, o modelo SPATTER é um modelo bem plausível, mas peca na pouca importância que dá ao passo 2 no processo de construção de MBHAs, que é na decomposição da árvore. O SPATTER representa a árvore sintática como uma seqüência de decisões feitas em uma análise de baixo para cima. Os parâmetros do modelo são associados a pares $<movimento, contexto>$. Segundo Collins, em alguns casos faltam informações importantes de desambiguação, e em outros casos os dados foram fragmentados sem necessidade. Os parâmetros seriam, assim, deficientes em termos de poder discriminativo e compactação.

E o que motivaria a escolha da decomposição no parsing? Segundo Collins, o princípio da localidade. O domínio da localidade de um evento é a região do espaço-tempo que ele pode afetar. A localidade para um evento seria a influência causal que ele pode ou não exercer. Formalismos altamente lexicalizados, como as Gramáticas Léxico Funcionais (LFG)(Kaplan e Bresnan 1982) e Minimalismo (Chomsky 1995) apontam a localidade da influência dos núcleos lexicais em uma árvore sintática. Cada palavra em uma sentença afeta um domínio limitado na árvore. Em LFG, representações ricas de restrições são associadas aos núcleos lexicais. Na discussão da teoria X-barra no minimalismo, Chomsky, na página 72 diz: " Uma estrutura X-barra é composta de projeções dos núcleos selecionados do léxico. Relações básicas, então, envolverão o núcleo como um único termo. Além disso, as relações básicas são tipicamente "locais"."

O objetivo de Collins então é, segundo essa teoria, escolher uma ordem de decomposição que permita uma parametrização que reflita a influência local de núcleos lexicais.

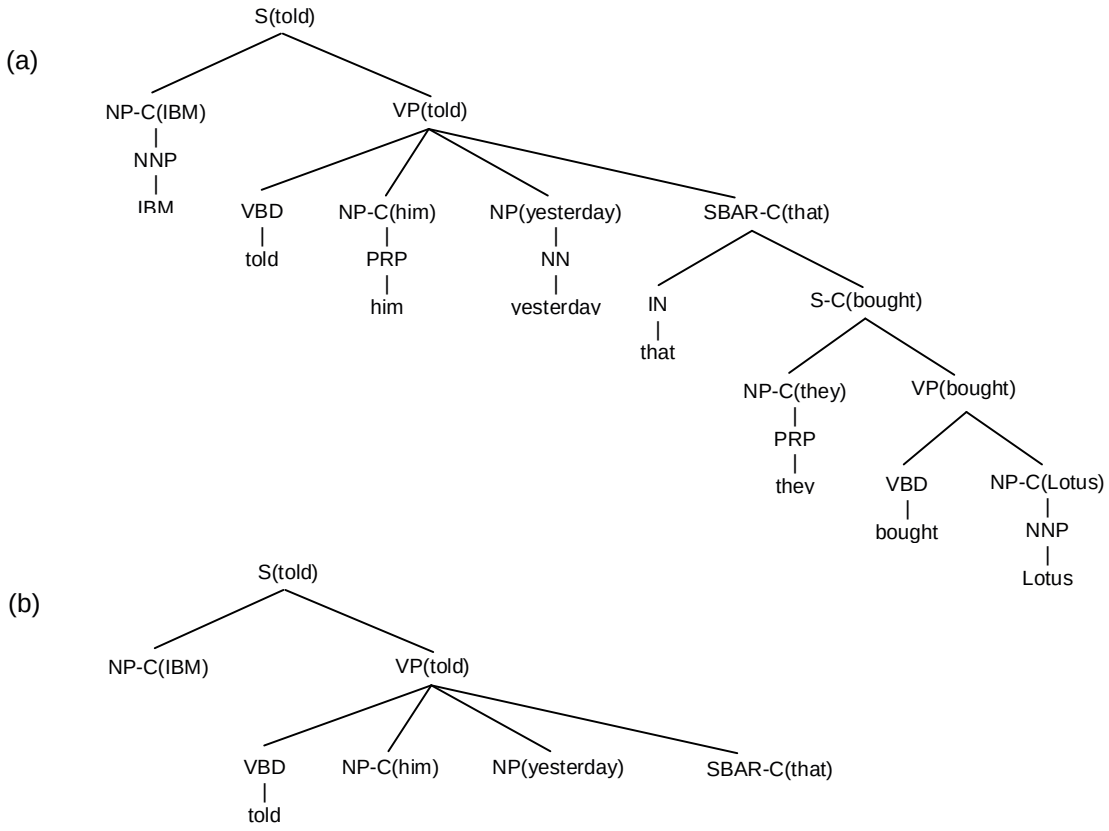

Figura 2.7: Exemplo de derivação da sentença "IBM told him yesterday that they bought Lotus". Em (a) uma árvore sintática, na qual os núcleos de cada não-terminal estão entre parênteses. A palavra com **-C** indica que é um complemento, em oposto a adjunto. Em (b) o domínio da localidade de *told* na árvore, e somente as partes lhe são diretamente dependentes

.

Assim, surge uma restrição importante da decomposição: o núcleo lexical deve ser gerado antes da estrutura que lhe é dependente, numa derivação centrada no núcleo (*head-centered*). Dada essa escolha de decomposição, suposições de independência refletindo o domínio de localidade de cada núcleo lexical surgem naturalmente. No modelo proposto por Collins, uma árvore sintática é representada como uma seqüência de decisões em uma ordem de cima para baixo, com uma derivação centrada no núcleo, em que cada decisão tem uma probabilidade associada.

Exemplo de estimativa de parâmetros

Considere a árvore da Figura 2.7, que ilustra um exemplo de árvore sintática com seus respectivos núcleos e um exemplo de localidade. A primeira derivação é a que escolhe o nó do topo da árvore. Essa decisão tem probabilidade

$$P(S(told)|START)$$

e produz como saída uma subárvore com S(told) como sua raiz.

A próxima parte da derivação - uma subseqüência de decisões - contribuirá para as probabilidades condicionadas a *told*. A entrada para essa subderivação é S(told), que acabou de ser gerado. A saída é construída incrementalmente em uma série de estágios. Cada estágio contribui com um tipo diferente de parâmetro, que são probabilidades correspondendo a:

1. *Parâmetros de projeção do núcleo*

   Há duas decisões envolvendo a geração do eixo principal da árvore: a probabilidade de um nó S com *told* como palavra-núcleo ter um nó SV[5] como seu núcleo (P(SV|S,told)); e a probabilidade de um nó SV com núcleo *told* tendo um nó verbo (Verbo) como seu núcleo (P(Verbo|SV,told)).

   E assim por diante, os outros não terminais também gerarão suas subárvores com seus respectivos núcleos.

2. *Escolha de parâmetros de subcategorização:*

   Nesse passo, são feitas decisões de subcategorização. Os complementos à esquerda e à direita são escolhidos para serem adicionados a cada nível da árvore:

   - Probabilidade de *told* ter um sujeito simples (um complemento único à esquerda)
     $P(SN^6 - C|pai = S, filho\_nucleo = SV, told, esquerda)$
   - Probabilidade de *told* não ter complemento à direita no nível S/SV
     $P(\{\}|pai = S, filho\_nucleo = SV, told, direita)$
   - Probabilidade de *told* não ter complementos à esquerda no nível SV/Verbo
     $P(\{\}|pai = SV, filho\_nucleo = Verbo, told, esquerda)$
   - Probabilidade de *told* levar tanto SN quanto SBAR[7] como complemento à direita no nível SV/Verbo
     $P(\{\text{SN-C,SBARC}\}|pai = SV, filho = Verbo, told, direita)$

[5] VP.
[6] NP.
[7] Cláusula subordinada.

Esses parâmetros permitem ao modelo aprender uma distribuição de probabilidade sobre todos os possíveis quadros de subcategorização para cada entrada no léxico.

Assim, de acordo com os passos descritos acima, uma seqüência é gerada de dentro para fora, ou seja, primeiro são escolhidos os não-terminais mais internos e depois os mais externos. A cada ponto, um não-terminal ou um símbolo STOP - que, quando gerado, pára a geração, é escolhido com alguma probabilidade. A probabilidade é condicionada ao pai e aos filhos não-terminais, ao núcleo e à direção relativa a esse núcleo.

Resultados obtidos

Nos testes feitos na seção 23 do WSJ, os modelos propostos por Collins conseguiram uma taxa de 88.3%/88.0% na recuperação dos sintagmas. Isso representa uma taxa relativa de redução de erro em torno de 25% sobre os resultados do SPATTER, quando treinados e testados com o mesmo conjunto de dados.

2.4.3 Modelo Baseado em Máxima Entropia

Ratnaparkhi (1999) propõe um modelo de parser baseado em uma técnica de aprendizado de máquina chamada modelagem de máxima entropia. A construção da árvore sintática é feita com ações semelhantes às de um parser de *empilhar-reduzir*. A seqüência de ações $a_1..a_n$ para construir uma árvore completa é chamada de derivação de AS. O parser parte de uma derivação $d = a_1..a_n$ e prediz alguma ação a_{n+1} para criar uma nova derivação $d' = a_1...a_{n+1}$. Não há uma gramática que dita quais ações são possíveis. Os modelos são treinados com as árvores da *treebank*. Todas as ações que levam a árvores bem-formadas são permitidas e os modelos de máxima entropia são usados para pontuar essas ações. A pontuação de cada ação na derivação é usada para computar a pontuação da derivação inteira. Para analisar uma sentença, o parser usa uma rotina de busca que explora o espaço de todas as árvores possíveis e tenta encontrar a de maior probabilidade.

São 4 as rotinas aplicadas em 3 passos sobre uma sentença de entrada: a rotina TAG, a rotina CHUNK, e as rotinas BUILD e CHECK. TAG atribui as etiquetas morfos-sintáticas; CHUNK coloca marcas nas palavras pré-identificando os sintagmas[8]; BUILD e CHECK se revesam para juntar as partes remanescentes formando a árvore resultante.

[8] Atribui etiquetas *Start X*, *Join X*, ou *Other*, em que X é uma etiqueta de sintagma. Por exemplo: "I/PRP-StartNP saw/VBD-Other the/DET-StartNP man/NN-JoinNP with/IN-Other the/DT-StartNP telescope/NN-JoinNP".

O parser também usa uma abordagem baseada na história, na qual uma probabilidade $p_X(a|b)$ é usada como a pontuação de uma ação a de procedimento $X \in TAG, CHUNK, BUILD, CHECK$, dependendo da derivação parcial b (também chamada de contexto ou história) que está disponível na hora da decisão. Os modelos condicionais p_X são estimados usando-se máxima entropia, que pode usar diversas informações no contexto b para computar a probabilidade de uma ação a da mesma rotina X. O modelo de máxima entropia trabalha com traços. Cada traço recebe um peso que corresponde ao quanto ele é útil para modelar os dados. São traços que usam noções como núcleos, combinação de núcleos, previsão olhando passos à frente (*lookahead*), e outras informações menos específicas.

Um traço é implementado como uma função $pc : C \rightarrow verdadeiro, falso$, chamada de predicado conceitual. Um predicado conceitual checa a presença ou ausência de informação útil no contexto $c \in C$ e retorna verdadeiro ou falso, conforme o que for observado. São várias os padrões que derivam os predicados. Os derivados de um padrão cons(0) procuram por núcleos de sintagmas, enquanto cons(-1,0) procuram por combinações entre núcleos. TAG procura por uma palavra e uma janela de duas palavras à esquerda e duas à direita. Os predicados conceituais para CHUNK procuram por duas palavras antes, as etiquetas morfossintáticas e as etiquetas sintáticas, bem como a palavra atual e as duas palavras e etiquetas morfossintáticas seguintes. BUILD usa o núcleo das duas palavras anteriores e as árvores correntes e os dois *chunks* seguintes. CHECK procura por duas palavras ao redor e os núcleos dos filhos do sintagma proposto.

Os predicados conceituais são usados para codificar as derivações na *treebank* como um conjunto de eventos de treinamento $AS_x = (a_1, b_1), ..., (a_n, b_n)$. Cada $(a, b) \in AS_X$ representa uma ação de rotina X na derivação e é codificado como $(a, cp_1, ..., cp_k)$, na qual *cps* são os predicados conceituais tal que $cp_i \in CP_X$ e $cp_i(b) = verdadeiro$ se b com ação a ocorreu na rotina X.

Resultados obtidos

Com esse modelo, Ratnaparkhi alcança resultados de 87.5%/86.3% de precisão/cobertura em sentenças retiradas da Penn Treebank, sendo melhores que os produzido pelo SPATTER.

2.5 Modelo Híbrido: Projeto RASP

Uma tendência, que também tem dado bons resultados, é a construção de sistemas híbridos, tais como o RASP, que integra duas abordagens bastante conhecidas na construção de parsers e hibridiza-as num novo produto. O projeto RASP (Briscoe e Carrol 2002) consiste de um ambiente integrado com todas as ferramentas necessárias para a realização de análise sintática automática. Os autores[9] prometem uma nova ferramenta de parsing que permitirá a construção de parsers de domínios irrestritos, treinados em ambientes adaptáveis a quaisquer conjuntos de dados; também produzirá a integração de técnicas estatísticas para desambiguação e técnicas de indução de gramática em caso de falha do parser.

O modelo é híbrido pois junta características dos parsers estatísticos lexicalizados, como os das seções anteriores, e as máquinas de estados finitos, aumentadas com heurísticas para proporcionar ligações de longa distância e construções de análises parciais (Abney 1996). A escolha pelas máquinas de estados finitos se deu por serem menos específicas ao domínio e não requererem grandes quantidades de dados anotados. O produto é o parser RADISP, que foi desenvolvido aproveitando os pontos fortes de cada uma dessas abordagens.

RASP se consiste num sistema que tem vários estágios. Primeiro, o texto é separado em seus itens lexicais. Segundo, os itens lexicais são anotados morfossintaticamente usando o modelo MOM. Após isso, são feitas a análise morfológica e a lematização dos itens lexicais anotados. As etiquetas morfossintáticas são, então, analisadas sintaticamente usando uma gramática de grande cobertura desenvolvida manualmente. Como última ação, as n melhores análises são selecionadas da floresta produzida usando um modelo de seleção estatístico condicionado ao contexto estrutural da análise e informação lexical, quando disponível.

A análise morfológica serve para habilitar os módulos posteriores a fazerem uso de informações lexicais associadas a lemas. Esse par $lema{+}etiqueta\_morfossintática\_afixada$ é passado para um parser probabilístico LR (Inui e et al. 1997), aumentado com informação lexical limitada, codificando a probabilidade de algumas combinações frasais (como por exemplo, verbo mais preposição/particípio), e com a probabilidade condicional de verbos de média e alta freqüências que aparecem com um dos 23 quadros de subcategorização pré-definidos.

[9] Conta com participantes de duas universidades: John Carrol, Diana McCarthy e Mark McLauchlan, da Universidade de Sussex e Ted Briscoe, Anna Korhonen e Judita Preiss, da Universidade de Cambridge.

A gramática, desenvolvida manualmente, consiste de cerca de 400 regras de unificação de estrutura de sintagmas (Briscoe e Carrol 1995a) e é projetada para enumerar possíveis valências para predicados (verbos, adjetivos e nomes) incluindo regras separadas para cada padrão de complementação possível no Inglês.

Na Figura 2.8, é apresentada a análise para a sentença "We describe a robust accurate domain-independent approach to statistical parsing incorporated into the new release of the ANLT toolkit, and publicly available as a research tool.". As partes em maiúsculas indicam a categoria-pai, que vem seguida de sua etiqueta morfossintática-núcleo. Os filhos do sintagma aparecem indentados sob seu pai e possuem a etiqueta sintática e a palavra com sua respectiva etiqueta morfossintática. A anotação segue a Teoria X-Barra de Chomsky dentro de um esquema de sintagmas baseado em traços.

Resultados obtidos

A gramática encontra no mínimo uma árvore cuja raiz é S em 80% dos casos. Nos outros casos, o parser retorna uma seqüência de análises parciais que cobrem a análise. Esse parser leva um tempo médio $O(n^2)$, em que n é o tamanho da sentença analisada. Ao corpus Susanne (Carrol 1994), atribui uma média de 1.28^n árvores sintáticas para uma sentença de n itens lexicais, e alcança um Fscore de 76.5%[10].

2.6 Outros trabalhos na área

Bikel (2000) e Bikel e Chiang (2000) exploram o modelo Collins e o modelo BBN, que usa *Tree Adjoining Grammars* (Chiang 2000), quando aplicados ao chinês. Com cerca de 3000 sentenças para treinamento, 353 para ajustes finais e 348 para testes, os modelos chegaram a patamares de 76.9%/77.2% de precisão/cobertura.

Charniak (2000), também propõe um modelo gerativo usando as mesmas idéias do modelo Collins. A diferença surge na forma como são recolhidas as estimativas, neste caso, baseadas em máxima entropia. Segundo ele, os resultados alcançam média de 90.1% de precisão nas seções do Wall Street Journal.

Sarkar (2001), propõe um novo método para treinar parsers estatísticos[11] usando pequenas quantidades de sentenças analisadas (9.625), um dicionário de estruturas lexicalizadas possíveis para cada palavra no conjunto de treinamento, e uma grande quan-

[10] Fscore $= \frac{(2*cobertura*precisão)}{cobertura+precisão}$

[11] Com uma técnica de aprendizado de máquina chamada de *Co-Training*.

```
(|T/txt-sc1/---|
 (|S/np_vp| |We:1_PPIS2|
  (|V/np_pp| |describe:2_VV0|
   (|NP/det_n| |a:3_AT1|
    (|N1/ap_n1/-| (|AP/a1| (|A1/a| |robust:4_JJ|))
     (|N1/ap_n1/-| (|AP/a1| (|A1/a| |accurate:5_JJ|))
      (|N1/n1_nm| |domain-independent:6_NN1|
       (|N1/n| |approach:7_NN1|)))))
   (|PP/p1|
    (|P1/p_npl| |to:8_II|
     (|AP/cj_int/+|
      (|AP/a1|
       (|A1/a_s| |statistical:9_JJ|
        (|S/n1_vp| (|N1/n| |parsing:10_NN1|)
         (|V/pp| |incorporate+ed:11_VVN|
          (|PP/p1|
           (|P1/p_np| |into:12_II|
            (|NP/det_n| |the:13_AT|
             (|N1/ap_n1/-| (|AP/a1| (|A1/a| |new:14_JJ|))
              (|N1/n_of| |release:15_NN1|
               (|PP/p1|
                (|P1/p_np| |of:16_IO|
                 (|NP/name_n1|
                  (|NP/det_n| |the:17_AT| (|N1/n| |ANLT:18_NP1|))
                  (|N1/n| |toolkit:19_NN1|)))))))))))))
      |,:20_,|
      (|AP/cj_end| |and:21_CC|
       (|AP/a1|
        (|A1/adv_a1| (|AP/a1| (|A1/a| |publicly:22_RR|))
         (|A1/a_pp1| |available:23_JJ|
          (|PP/p1|
           (|P1/p_np| |as:24_II|
            (|NP/det_n| |a:25_AT1|
             (|N1/n1_nm| |research:26_NN1|
              (|N1/n| |tool:27_NN1|)))))))))))))
```

Figura 2.8: Análise para a sentença ‘‘We describe a robust accurate domain-independent approach to statistical parsing incorporated into the new release of the ANLT toolkit, and publicly available as a research tool.’’.

tidade de texto não etiquetado (30.137 sentenças). Os resultados obtidos chegaram a 80.02%/79.64% de precisão/cobertura para as sentenças do Wall Street Journal.

Clark, Hockenmaier, e Steedman (2002) e Hockenmaier e Steedman (2002) apresentaram recentemente, uma proposta de parser que usa uma gramática baseada em combinação de categorias[12] (Steedman 2000) para derivar estruturas de dependências[13]. Com essa gramática, o parser é capaz de capturar dependências de longa distância como coordenação, dependências argumento-predicado, etc., que não são tratadas pelos parsers estatísticos anteriores. O parser deve escolher a estrutura de dependência mais provável para uma sentença S. Chiang (2000), já apresentava uma alternativa parecida usando as *Tree Adjoining Grammars* em lugar das gramáticas livres de contexto dos trabalhos-referência na área.

2.7 Conclusão

Este capítulo apresentou a evolução dos modelos de linguagem estatísticos, partindo de modelos bastante simples como n-gramas, passando pelas gramáticas livres de contexto probabilísticas, chegando até os modelos baseados na história da análise.

Como visto, os modelos n-gramas se preocupam apenas com a ordem linear das palavras e das etiquetas (quando usadas). São modelos atrativos para lidar com problemas linguísticos porque assumem uma localidade ao determinar as probabilidades e assim não sofrem tanto com o problema de dados esparsos (por exemplo: um modelo baseado em trigramas pode ser adaptado para lidar com dados esparsos estimando também probabilidades entre bigramas e unigramas). O problema que resolvem é a estimativa de uma probabilidade conjunta de uma seqüência de n eventos, assumindo que a probabilidade de cada evento é dependendente de uma subseqüência desses eventos. A etiquetagem morfossintática é a aplicação de maior sucesso de tais modelos, uma vez que a necessidade por informação de longa distância não é tão necessária (Magerman 2003). No entanto, para parsing os modelos têm se mostrado insuficiente. Devido à complexa estrutura da linguagem, são necessários modelos que consigam capturar tal estrutura. Kupiec (1992) já observa a necessidade de se aumentar o modelo MOM dos etiquetadores morfossintáticos para reconhecer estruturas hierárquicas rudimentares, ao invés da simples ordem linear das palavras, tornando-os assim eficientes em problemas como parsing.

[12] *Combinatory Categorial Grammar.*
[13] A *treebank* usada tem que ser adaptada para representar essas estruturas de dependências.

A primeira tentativa com relativo sucesso em parsing se deu na adição de probabilidades às regras das gramáticas livres de contexto, criando-se um modelo chamado de gramática livre de contexto probabilística (GLC-P), com resultados satisfatórios e melhorias de performance bastante significativas. O problema dos modelos iniciais com relação à falta de tratamento de palavras, por exemplo, para determinar a análise correta de uma estrutura ambígua, foi solucionado através da lexicalização das gramáticas, fazendo com que os sintagmas pudessem ser associados a núcleos lexicais e o parser pudesse ter um controle na seleção das palavras. Apesar do sucesso, as GLC-Ps assumem que a expansão de um não-terminal é independente da expansão de outros não-terminais. No entanto, a escolha de uma expansão para um nó pode depender da localização desse nó na árvore sintática. O modelo também não leva em consideração o contexto ou a co-ocorrência lexical. Isto significa que, em algumas vezes, as análises sugeridas podem não ser as melhores. A modelagem através das GLC-P tende a ser robusta, mas a desvantagem de não considerar o contexto pode fazer com que um modelo de trigrama alcance melhor resultado, ainda que não considerando a estrutura interna da linguagem. Manning e Schütze (1999) argumentam que mesmo estruturalmente o modelo é deficiente. Ele não leva em conta a posição na qual um sintagma aparece na sentença (por exemplo: um sintagma nominal funcionando como sujeito tem, quase sempre, uma expansão diferente de um sintagma nominal que funciona como objeto). Além disso, não há uma *treebank* grande o suficiente para treinar todas as probabilidades, e a maior limitação é que não conseguem capturar relações envolvendo itens lexicais fora do sintagma sendo trabalhado. Magerman e Marcus (1991), com o parser Pearl, evitam o problema da falta de informação de contexto, condicionando as probabilidades a algumas seqüências de etiquetas morfossintáticas.

O modelo do Pearl motivou o surgimento dos modelos baseados na história da análise, que usam uma carga maior de informações lingüísticas como contexto e informações lexicais, para recuperar formas estruturais, úteis no parsing. Tais modelos têm alguns fortes representantes como o SPATTER, o modelo, baseado em máxima entropia, de Ratnaparkhi, e os modelos de Collins (base para esta tese).

O SPATTER ilustra a eficiência do modelo em incorporar grande quantidade de informação conceitual pela aplicação de algoritmos de aprendizagem, como por exemplo, os baseados em árvores de decisão, sobre uma *treebank*. Segundo Magerman, a vantagem do modelo usado pelo SPATTER está no fato de que ele produz um parser preciso sem a ajuda de uma base de conhecimento complicada ou uma gramática, e que conta apenas com uma informação lingüística bastante limitada. Outra vantagem está no fato de o modelo baseado em árvores de decisão ser capaz de desambiguar informações lexicais de

longa distância, escolhendo seletivamente elementos do contexto que contribuem para as decisões de desambiguação. O sistema não faz simplesmente uma atribuição prévia das etiquetas das sentenças e escolhe aquelas com a melhor seqüência; ele usa um modelo probabilístico para determinar as etiquetas para as palavras, e considera todas as seqüências possíveis de acordo com as probabilidades que lhe são atribuídas.

Ratnaparkhi (1999) melhora o modelo SPATTER usando representações lingüísticas mais naturais para palavras e sintagmas, ao invés da aplicação de técnicas para binarização (o SPATTER só trabalha com informações binarizadas). O modelo baseado em máxima entropia alcança resultados bem próximos aos conseguidos por Collins, e a principal vantagem é que usa fatos simples para construir o conhecimento para o aprendizado. Isso faz com que o modelo possa ser usado para outras pesquisas, como por exemplo, as mostradas por Charniak (2000).

No entanto, apesar de não usar técnicas tão eficientes de aprendizagem de máquina, os modelos de Collins são os que se comportam melhor nesse cenário de sentenças irrestritas. Seu modelo baseado na noção de dependência entre palavras, sendo uma núcleo e outra modificador, e a adição do parâmetro distância, que captura contexto (e pode ser estendido, lidando com mais palavras e etiquetas morfossintáticas), faz com que a recuperação de toda a estrutura arbórea seja feita mais naturalmente.

Fica claro, portanto, que, independentemente de quais técnicas sejam usadas para o processo de desambiguação, toda a informação necessária para a realização de tal processo deve estar disponível. As palavras são claramente necessárias para tomar as decisões e, em alguns casos, a informação estrutural de longa distância é também necessária. Modelos estatísticos para parsing devem considerar muito mais traços de uma sentença do que podem ser manuseados por modelos n-gramas. Os modelos descritos nessa seção ilustram como grandes quantidades de informação contextual podem ser incorporadas em um modelo estatístico para parsing aplicando-se algoritmos de aprendizado como árvores de decisão ou métodos de estimativas baseados em contagens a partir de uma *treebank*.

Capítulo 3

Parsing Probabilístico baseado na noção de núcleo como elemento central na análise

3.1 Introdução

Collins, em seus trabalhos, evolui seu modelo inicial, baseado em dependências lexicais entre bigramas, propondo três novos modelos gerativos, todos eles baseados na noção *head-centering*, em que o núcleo é o elemento principal e direcionador de todo o processo de geração de uma árvore sintática. A seguir, esses trabalhos são detalhados, uma vez que servem de base para a construção do parser PAPO, para o Português brasileiro. Detalhes de implementação também podem ser encontrados em Bonfante e Nunes (2001).

3.2 Modelo Baseado em Dependências Lexicais

Collins (1996) propõe um modelo baseado em dependências lexicais entre bigramas. Modelo mais simples que o SPATTER, usa informações lexicais para modelar relações núcleo-modificador[1].

Conforme observado no capítulo anterior, dois componentes são necessários para essa abordagem estatística: o modelo, que atribui probabilidade para cada árvore sintática de

[1] A palavra "modificador" é usada para expressar a noção lingüística tanto de argumento quanto de adjunto.

uma sentença, e o parser, que é a ferramenta usada para encontrar o AS_{best}, segundo a fórmula:

$$AS_{best}(S) = argmax < S \rightarrow P(AS|S) > \tag{3.1}$$

Dada uma sentença S e uma árvore AS, o modelo estima a probabilidade condicional $P(AS|S)$. A idéia chave é: cada árvore AS pode ser representada por um conjunto de *SN_Bases* (B) [2] e um conjunto de *dependências* (D) [3]. Assim,

$$AS = (B, D)$$

e

$$P(AS|S) = P(B, D|S) = P(B|S) \times P(D|S, B) \tag{3.2}$$

em que S é um conjunto de palavras (p_n) com suas respectivas etiquetas morfossintáticas (e_n) tal que $S = <(p_1, e_1), (p_2, e_2)...(p_n, e_n)>$.

Com tais relações estabelecidas, o objeto de trabalho passa a ser uma sentença reduzida, formada a partir da sentença S inicial, sem as pontuações e com apenas os núcleos dos SN_Bases. O modelo de dependência é baseado no relacionamento entre as palavras na sentença reduzida. O mapeamento entre árvores e estruturas de dependência é central para o modelo de dependência. Ele é definido em dois passos:

1. Para cada sintagma $Sint\_pai \rightarrow <Sint\_filho_1...Sint\_filho_n>$, identifica-se qual filho de $Sint\_pai$ é o núcleo. Cada núcleo é propagado para seu pai, e o processo se repete sucessivamente;

2. Extraem-se os relacionamentos núcleo-modificador. As dependências são marcadas por triplas $<M, P, NUC>$, em que M é o sintagma modificador, P é o sintagma pai e NUC é o sintagma núcleo. Por exemplo, a tripla <SN,S,SV> representa a dependência sujeito-verbo.

O relacionamento é denotado por $REL(j) = (nuc_j, R_j)$ representando que a j_{esima}

[2] O conjunto SN_Base é formado por núcleos dos sintagmas nominais. Collins parte do pressuposto de que apenas os núcleos de tais sintagmas é que são importantes no processo análise da sentença como um todo; as outras palavras só agem dentro do escopo do próprio sintagma.

[3] Dependência entre núcleos e modificadores.

palavra na sentença reduzida é um modificador para seu núcleo nuc_j, com relacionamento R_j. O modelo assume que as dependências são independentes entre si, tal que a probabilidade final do modelo possa ser calculada como o produto de cada relacionamento existente na sentença reduzida:

$$P(D|S,B) = \prod_{j=1}^{m} P(REL(j)|S,B) \tag{3.3}$$

Mas, como estimar $P(REL(j)|S,B)$?

Como é provável que a mesma sentença não apareça em ambos, treinamento e teste, é necessário usar tratamentos para lidar com o problema dos dados esparsos. São definidas as seguintes funções, onde V é o vocabulário de palavras vistas no treinamento, E é o conjunto de todas as etiquetas morfossintáticas, e $TREIN$ o conjunto de treinamento, de sentenças reduzidas:

- $Cont(<a,\alpha>,<b,\beta>)$ para $a,b \in V$ e $\alpha,\beta \in E$: número de vezes que $<a,\alpha>$ e $<b,\beta>$ são vistos na mesma sentença reduzida nos dados de treinamento.

- $Cont(R,<a,\alpha>,<b,\beta>)$: número de vezes que $<a,\alpha>$ e $<b,\beta>$ são vistos na mesma sentença, e $<a,\alpha>$ modifica $<b,\beta>$ com relacionamento R.

- $P(R|<a,\alpha>,<b,\beta>)$: probabilidade com que $<a,\alpha>$ modifica $<b,\beta>$ com relacionamento R, dado que $<a,\alpha>$ e $<b,\beta>$ aparecem na mesma sentença reduzida. Segundo o método de estimativa ML ((Berger, Pietra, e Pietra 1996), (Chi 1998)), $P(R|<a,\alpha>,<b,\beta>)$ pode ser estimado como o número de vezes (Cont) que o relacionamento R entre as palavras a e b foi observado, dividido pelo número de vezes que as palavras a e b aparecem juntas:

$$\hat{P}(R| < a,\alpha >, < b,\beta >) = \frac{Cont(R, < a,\alpha >, < b,\beta >)}{Cont(< a,\alpha >, < b,\beta >)} \tag{3.4}$$

É possível, então, fazer a seguinte aproximação:

$$P(REL(j) = (nj, Rj)|S,B) \approx \frac{\hat{P}(R_j| < \bar{p}_j, \bar{e}_j >, < \bar{p}_{nj}, \bar{e}_{nj} >)}{\sum_{k=1..m,k\neq j,t\in T} \hat{P}(t| < \bar{p}_j, \bar{e}_j >, < \bar{p}_k, \bar{e}_k >)} \tag{3.5}$$

na qual T é o conjunto de todas as triplas de não-terminais; e o denominador, um fator de normalização, que garante que

$$\sum_{k=1..m,k\neq j,t\in T} P(REL(j)=(k,t)|S,B)=1 \tag{3.6}$$

Assim, a probabilidade do conjunto de dependências, dadas as sentenças reduzidas e o conjunto de SN_Bases, pode ser estimada como o produto de todos os relacionamentos entre pares modificador-núcleo observados para montá-lo.

$$P(D|S,B)\approx\prod_{j=1}^{m}\frac{\hat{P}(R_j|<\bar{p}_j,\bar{e}_j>,<\bar{p}_{nucj},\bar{e}_{nucj}>)}{\sum_{k=1..m,k\neq j,t\in T}\hat{P}(t|<\bar{p}_j,\bar{e}_j>,<\bar{p}_k,\bar{e}_k>)} \tag{3.7}$$

Como o denominador de 3.7 é uma constante, maximizar $P(D|S,B)$ sobre D para um (B,S) fixo é equivalente a maximizar o produto dos numeradores $N(D|S,B)$:

$$N(D|S,B)=\prod_{j=1}^{m}\hat{P}(R_j|<\bar{p}_j,\bar{e}_j>,<\bar{p}_{nucj},\bar{e}_{nucj}>) \tag{3.8}$$

Distância

Collins introduz um conceito de distância nesse modelo baseado em dependências entre bigramas. Segundo ele, a distância é uma variável crucial quando se decide se duas palavras estão relacionadas. O modelo é estendido com a variável de distância Δ. Por exemplo $Cont(<a,\alpha>,<b,\beta>,\Delta)$ é o número de vezes que $<a,\alpha>$ e $<b,\beta>$ são vistos na mesma sentença com uma distância Δ separando-as. Assim, ao invés de 3.8, o modelo passa a maximizar a seguinte equação:

$$N(D|S,B)=\prod_{j=1}^{m}\hat{P}(R_j|<\bar{p}_j,\bar{e}_j>,<\bar{p}_{nucj},\bar{e}_{nucj}>,\Delta_{j,nucj}) \tag{3.9}$$

Para medir o fator Δ, o autor recolhe uma combinação de 6 características, obtidas heuristicamente:

- (Questão 1) a palavra *nucj* precede a palavra *j*?
- (Questão 2) elas são adjacentes?
- (Questão 3) há um verbo entre elas?

Essa pergunta é feita para apontar verbos que se sobrepõem e ajudar na escolha de ligações entre as palavras.

- (Questões 4, 5 e 6) há 0,1,2 ou mais que 2 vírgulas (**,** e **:**) entre elas? há uma vírgula seguindo imediatamente o núcleo e o modificador? há uma vírgula precedendo imediatamente o núcleo e seu modificador?

Dados Esparsos

A estimativa ML, apresentada na Equação 3.10, provavelmente terá problemas de dados esparsos. Para resolver o problema, é usada uma estratégia de tratamento de dados esparsos para amenizar as probabilidades de $Cont(<\bar{p}_j, \bar{e}_j>,<\bar{p}_{nucj}, \bar{e}_{nucj}>, \Delta_{j,nucj})$, e, assim, evitar que a estimativa seja indefinida.

$$\hat{P}(R| < a, \alpha >, < b, \beta >) = \frac{Cont(R, < a, \alpha >, < b, \beta >)}{Cont(< a, \alpha >, < b, \beta >)} \tag{3.10}$$

Para se chegar à estimativa da Equação 3.10, há quatro estimativas, E1, E2, E3, E4, baseadas respecivamente em: (1) tanto palavras quanto etiquetas morfossintáticas; (2) $\bar{p}_j$ e duas etiquetas morfossintáticas; (3) $\bar{p}_{nucj}$ e duas etiquetas morfossintáticas; (4) as etiquetas morfossintáticas sozinhas.

$$E_1 = \frac{\eta_1}{\delta_1} \quad E_2 = \frac{\eta_2}{\delta_2} \quad E_3 = \frac{\eta_3}{\delta_3} \quad E_4 = \frac{\eta_4}{\delta_4}$$

em que:

$$\delta_1 = C(<\bar{p}_j, \bar{e}_j>,<\bar{p}_{nucj}, \bar{e}_{nucj}>, \Delta_{j,nucj})$$

$$\delta_2 = C(<\bar{p}_j, \bar{e}_j>,<\bar{e}_{nucj}>, \Delta_{j,nucj})$$

$$\delta_3 = C(<\bar{e}_j>,<\bar{p}_{nucj}, \bar{e}_{nucj}>, \Delta_{j,nucj})$$

$$\delta_4 = C(<\bar{e}_j>,<\bar{e}_{nucj}>, \Delta_{j,nucj})$$

$$\eta_1 = C(R_j,<\bar{p}_j, \bar{e}_j>,<\bar{p}_{nucj}, \bar{e}_{nucj}>, \Delta_{j,nucj})$$

$$\eta_2 = C(R_j,<\bar{p}_j, \bar{e}_j>,<\bar{e}_{nucj}>, \Delta_{j,nucj})$$

$$\eta_3 = C(R_j,<\bar{e}_j>,<\bar{p}_{nucj}, \bar{e}_{nucj}>, \Delta_{j,nucj})$$

$$\eta_4 = C(R_j,<\bar{e}_j>,<\bar{e}_{nucj}>, \Delta_{j,nucj})$$

Segundo o autor, as estimativas 2 e 3 competem, no sentido de que, para um dado par de palavras nos dados de teste, ambas podem existir, e são igualmente "específicas"

para o exemplo. Assim, são combinadas da seguinte forma:

$$E_{23} = \frac{\eta_2 + \eta_3}{\delta_2 + \delta_3} \tag{3.11}$$

As três estimativas E_1, E_2 e E_4 ocorrem na forma, com pequenas modificações, conforme é descrito em (Jelinek 1990), segundo um método de *deleted interpolation*, que dá uma estimativa mais suavizada para tratar dados esparsos:

se E_1 existe, i.e. $\delta_1 > 0$ **então**

$$\hat{P}(R_j| < \bar{p}_j, \bar{e}_j >, < \bar{p}_{nucj}, \bar{e}_{nucj} >, \Delta_{j,nucj}) = \lambda_1 \times E_1 + (1 - \lambda_1) \times E_{23} \tag{3.12}$$

senão

se E_{23} existe, i.e. $\delta_2 + \delta_3 > 0$ **então**

$$\hat{P}(R_j| < \bar{p}_j, \bar{e}_j >, < \bar{p}_{nucj}, \bar{e}_{nucj} >, \Delta_{j,nucj}) = \lambda_2 \times E_{23} + (1 - \lambda_2) \times E_4 \tag{3.13}$$

senão

$$\hat{P}(R_j| < \bar{p}_j, \bar{e}_j >, < \bar{p}_{nucj}, \bar{e}_{nucj} >, \Delta_{j,nucj}) = E_4 \tag{3.14}$$

fim se

fim se

Collins simplifica as fórmulas propostas em (Jelinek 1990) para encontrar λ:

$$\lambda_1 = \frac{\delta_1}{\delta_1 + 1}$$

$$\lambda_2 = \frac{\delta_2 + \delta_3}{\delta_2 + \delta_3 + 1}$$

O modelo SN_Base

O modelo SN_Base serve como redutor dos elementos da sentença que não são importantes no processo geral de decisão por ligações. Os elementos internos de um sintagma nominal não participam da contagem das co-ocorrências nos dados de treinamento. Apenas as palavras identificadas como os núcleos dos SNs são consideradas no modelo princi-

pal, podendo assim, contribuir de uma forma mais significativa na identificação de relações de adjacência e ligações entre as palavras.

Grosso modo, nesse modelo são etiquetados os espaços entre as palavras, com o objetivo de identificar as fronteiras de cada sintagma nominal encontrado na sentença. Por exemplo, a sentença "John Smith, the president of IBM, has announced his resignation yesterday" receberá etiquetas S(tart), C(ontinue), E(nd), B(etween) ou N(ull) de acordo com a característica de cada uma das fronteiras, conforme mostrado abaixo:

[John Smith] [the president] of [IBM] has announced [his resignation] [yesterday]

S John **C** Smith **B** the **C** president **E** of **S** IBM **E**

has **N** announced **S** his **C** resignation **E** yesterday.**N**

As estimativas das probabilidades são baseadas em contagens de pares de palavras sucessivas nas sentenças de treinamento não reduzidas, nas quais as fronteiras entre os SN_Base definem a característica do espaço(gap)(S,C,E,B ou N). A probabilidade de uma seqüência de SN_Base em uma seqüência não reduzida S é então definida como:

$$\prod_{i=2..n} \hat{P}(G_i|p_{i-1}, e_{i-1}, p_i, e_i, ci) \tag{3.15}$$

em que c_i indica se há uma vírgula entre duas palavras, no caso com valor 1, ou não, com valor 0.

Parser

O algoritmo de busca é um chart parser[4]. Cada dependência com uma tripla de não-terminais que não foi vista nos dados de treinamento terá probabilidade 0. Collins usa um algoritmo de programação dinâmica que segue alguns critérios: se dois sintagmas propostos expandem o mesmo conjunto de palavras, têm a mesma etiqueta, núcleo, e distância do núcleo ao lado esquerdo e ao lado direito, a probabilidade mais baixa pode ser descartada. Vide Figura 3.1.

Avaliação

[4]No Apêndice C é apresentada uma descrição sobre os fundamentos dos chart parsers.

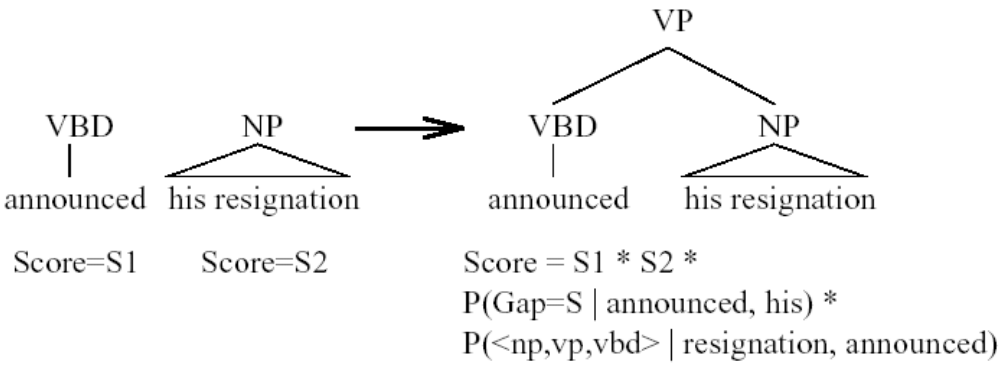

Figura 3.1: Exemplo de formação dos sintagmas
(Collins 1996)

O parser de Collins foi treinado nas seções 2-21 do WSJ, anotadas e reunidas no Penn Treebank (Marcus e et al. 1993), contendo aproximadamente 40.000 sentenças e testado na seção 23 (2.416 sentenças). Na avaliação foram usadas as medidas do PARSEVAL (Black e et al. 1991), observando critérios como precisão e cobertura. Uma outra medida também usada foi a de *Crossing Brackets* (CBs)[5], que mede o número de sintagmas que violam as fronteiras de sintagmas quando comparados aos sintagmas na treebank. Os melhores resultados são mostrados na Tabela 3.1, extraídos de testes feitos numa Sun SPARCServer 1000E, usando 100% do processador de 60Mhz. O parser usa cerca de 180 MB de memória, o carragamento da tabela de bigramas leva cerca de 8 minutos, e o treinamento com 40.000 sentenças, 15 minutos. Algumas melhorias foram otimizaram tais testes: uma constante de probabilidade (*threshold*) foi usada para que qualquer sintagma com menor probabilidade do que o limiar estabelecido seja descartada. Se uma análise for encontrada, deverá ser a de probabilidade mais alta, uma vez que todas as descartadas, teoricamente, não fariam parte do final. Uma outra estratégia de busca é usada: para cada conjunto de palavras na sentença, a probabilidade P mais alta do sintagma é gravada. Todos os sintagmas derivando as mesmas palavras devem ter probabilidade maior que $\frac{P}{B}$, para alguma constante B.

[5]No Apêndice B é apresentada uma revisão bibliográfica sobre critérios e medidas usadas na avaliação de parsers

Tabela 3.1: Alguns resultados do modelo de dependência entre bigramas. "Não" em informação lexical significa uso de etiquetas morfossintaticas somente. "Não" em distância significa uso da Questão 1, que apenas verifica se o núcleo ($nucj$) precede o modificador (j).

Distância	Informação lexical	Cobertura	Precisão	CBs
sim	sim	85.0%	85.1%	1.39
sim	não	76.1%	76.6%	2.26
não	sim	80.9%	83.6%	1.51

3.3 Modelos Gerativos Lexicalizados para Análise Sintática Estatística

Collins (1997), introduz três modelos (Modelos 1, 2 e 3) que usam uma nova abordagem para melhorar o modelo de bigramas. O modelo define uma probabilidade conjunta $P(AS, S)$ sobre pares árvores-sentenças. Usa um modelo baseado na história da análise: uma árvore sintática é representada como uma seqüência de decisões, a partir de uma derivação top-down e centrada no núcleo da árvore sintática. Segundo o autor, a representação da árvore sintática dessa forma permite que suposições de independência sejam feitas, levando a parâmetros condicionados a núcleos lexicais: parâmetros de projeção do núcleo, subcategorização, colocação de complemento/adjunto, dependência, distância, etc.

A seguir são apresentados cada um dos modelos. O Modelo 2 representa uma evolução em relação ao Modelo 1; e o Modelo 3, em relação ao Modelo 2.

3.3.1 Modelo 1

O modelo 1 apresenta uma proposta de como estender uma Gramática Livre de Contexto Probabilística (GLC-P) para uma gramática lexicalizada (que considera itens lexicais) de maneira que o resultado seja bastante similar ao modelo descrito na seção anterior. O Modelo 1 tem ainda parâmetros que correspondem a dependências entre pares de núcleos; a distância também é incorporada como uma medida, generalizando o modelo para uma abordagem baseada na história da análise.

Vantagens do modelo em relação ao anterior:

- o modelo não é deficiente (i.e., $\sum P(AS, S) = 1$); regras unárias são manuseadas de

uma forma bastante natural pelo modelo;

- a medida de distância é melhorada. Por exemplo, a variável de adjacência passa a corresponder diretamente a estruturas de ligação à direita;
- o modelo mostra uma melhora de 1.9%/1.5% de precisão/cobertura;
- o modelo pode condicionar sua estimativa usando qualquer estrutura previamente gerada;
- a etiquetagem morfossintática é naturalmente incorporada ao modelo;
- o modelo define uma medida de probabilidade conjunta $P(AS, S)$, podendo ser treinado de uma maneira não supervisionada pelo algoritmo *Expectation Maximization.*

A geração do lado direito da regra é quebrada em uma seqüência de pequenos passos. Cada regra passa a ter a forma

$$Pai(nuc) = E_n(pe_n)...E_1(pe_1)NUC(nuc)D_1(pd_1)...D_m(pd_m) \tag{3.16}$$

em que $NUC(nuc)$ representa o núcleo do sintagma, que recebe o item lexical nuc de seu pai Pai; $E_1...E_n$ e $D_1...D_m$ são seus modificadores, à esquerda e à direita, com itens lexicais pe e pd, respectivamente. As seqüências à direita e à esquerda são aumentadas com um símbolo $STOP$, de forma que permita um processo de Markov para o modelo. Assim, $E_{n+1} = D_{m+1} = STOP$.

A regra de probabilidade pode ser reescrita usando a regra da cadeia de probabilidades:

$$P(E_{n+1}(pe_{n+1})...E_1(pe_1)NUC(nuc)D_1(pd_1)...D_{m+1}(pd_{m+1})|Pai(nuc)) = P_{nuc}(NUC|Pai(nuc))\times$$

$$\prod_{i=1..n+1} P_{esq}(E_i(pe_i)|E_1(pe_1)...E_{i-1}(pe_{i-1}), Pai(nuc), NUC)\times$$

$$\prod_{j=1..m+1} P_{dir}(D_j(pd_j)|E_1(pe_1)...E_{n+1}(pe_{n+1}), D_1(pd_1)...D_{j-1}(pd_{j-1}), Pai(nuc), NUC)$$

Para o modelo ser MBHA, cada modificador poderia depender de qualquer função Φ dos modificadores anteriores, categoria do núcleo/pai e núcleo.

$$P_{esq}(E_i(pe_i)|E_1(pe_1)...E_{i-1}(pe_{i-1}), Pai(nuc), NUC) =$$
$$P_{esq}(E_i(pe_i)|\Phi(E_1(pe_1)...E_{i-1}(pe_{i-1}), Pai(nuc), NUC))$$

$$P_{dir}(D_j(pd_j)|E_1(pe_1)...E_{n+1}(pe_{n+1}), D_1(pd_1)...D_{j-1}(pd_{j-1}), Pai(nuc), NUC) =$$
$$P_{dir}(D_j(pd_j)|\Phi(E_1(pe_1)...E_{n+1}(pe_{n+1}), D_1(pd_1)...D_{j-1}(pd_{j-1}), Pai(nuc), NUC))$$

Fazendo a suposição de independência de que os modificadores são gerados independentemente uns dos outros, ou seja, fazendo Φ ignorar tudo a não ser P, NUC e nuc, temos

$$P_{esq}(E_i(pe_i)|E_1(pe_1)...E_{i-1}(pe_{i-1}), Pai(nuc), NUC) = P_{esq}(E_i(pe_i)|Pai(nuc), NUC)$$

$$P_{dir}(D_j(pd_j)|E_1(pe_1)...E_{n+1}(pe_{n+1}), D_1(pd_1)...D_{j-1}(pd_{j-1}), Pai(nuc), NUC) =$$
$$P_{dir}(D_j(pd_j)|Pai(nuc), NUC)$$

A geração de um lado direito de um regra, dado o lado esquerdo, é então feita em três passos, sucessivamente, até que toda a árvore seja construída: (1) gera-se o núcleo (NUC); (2) geram-se modificadores à esquerda (E) e (3) geram-se modificadores à direita (D).

Adicionando Distância

Assim como na proposta anterior (Collins 1996), Collins também adiciona distância a esse modelo (Collins 1997). Essa adição é importante para capturar preferências por estruturas de ligação à direita (que quase sempre traduz a preferência por dependências entre palavras adjacentes) e a preferência por dependências que não cruzam um verbo. A distância pode ser incorporada adicionando uma quantidade de dependência entre os

modificadores.

$$P_{esq}(E_i(pe_i)|Pai, NUC, nuc, E_1(pe_1)...E_{i-1}(pe_{i-1}) =$$
$$P_{esq}(E_i(pe_i)|NUC, Pai, nuc, distancia_{esq}(i-1))$$

$$P_{dir}(D_i(pd_i)|Pai, NUC, nuc, D_1(pd_1)...D_{i-1}(pd_{i-1}) =$$
$$P_{dir}(D_i(pd_i)|NUC, Pai, nuc, distancia_{dir}(i-1))$$

A distância é um vetor contendo duas informações: adjacência (que permite aprender preferências por ligações à direita) e existência de um verbo entre eles (que permite aprender a preferência pela modificação do verbo mais recente).

3.3.2 Modelo 2

O Modelo 2, proposto por Collins, introduz a distinção entre complemento/adjunto. Os complementos[6] são acrescidos do sufixo "C". Assim, o modelo é estendido para fazer essa distinção e também para ter parâmetros que correspondam diretamente a distribuições de probabilidade sobre subcategorizações para núcleos.

O processo gerativo passa então a incluir escolha probabilística de subcategorização à esquerda ou à direita:

1. Escolhe o núcleo com probabilidade $P_{nuc}(NUC|Pai, nuc)$

2. Escolhe subcategorizações à esquerda e à direita, E-C e D-C, com probabilidades $P_{esq}(E-C|Pai, NUC, nuc)$ e $P_{dir}(D-C|Pai, NUC, nuc)$. Cada subcategorização é um conjunto que especifica os complementos que o núcleo requer como modificadores à direita ou à esquerda.

3. Gera modificadores à esquerda e à direita com probabilidades

$$P_{esq}(E_i(pe_i)|NUC, Pai, nuc, distancia_{esq}(i-1), E-C)$$

[6] Palavra usada num sentido geral, incluindo tanto complemento quanto especificadores, segundo a terminologia usada na teoria *Government and Binding* (Chomsky 1995).

e

$$P_{dir}(D_i(pd_i)|NUC, Pai, nuc, distancia_{dir}(i-1), D-C)$$

Conforme os complementos são gerados, eles são removidos do conjunto de subcategorização (SUBCAT) apropriado. A probabilidade de gerar o símbolo STOP é 1 quando SUBCAT estiver vazio, e a probabilidade de gerar um complemento será 0 quando ela não estiver no SUBCAT.

3.3.3 Modelo 3

O Modelo 3 é estendido da gramática de estrutura de frase generalizada para possibilitar tratamento de *Wh-moviment*. Introduz parâmetros *TRACES* e *Wh-Moviment*. Por exemplo, na frase "The store that IBM bought last week", o modelo usaria as regras para gerá-la[7]:

1. SN → SN SBAR(+gap)
2. SBAR(+gap) → Wh_{sn} S-C(+gap)
3. S(+gap) → SN−C SV(+gap)
4. SV(+gap) → Verbo $Trace$ SN

Casos especiais

Os SN_Base não sofrem suposições de independência. A probabilidade da regra

SN_Base(cão) → Determinante(o) Nome(cão), é estimada como:

$$P_{nuc}(Nome|SN\_Base, cão) \times P_{esq}(Determinante(o)|SN\_Base, Nome, cão) \times$$

$$P_{esq}(STOP|SN\_Base, Nome, cão) \times P_{dir}(STOP|SN\_Base, Nome, cão)$$

Coordenação

Ao invés de a coordenação ser gerada num processo normal, como mais um modificador, o processo gerativo foi alterado para gerar a coordenação e o sintagma seguinte num único passo; assim, um não-terminal e um flag binário *coord* são gerados; *coord* = 1 se há um

[7]SBAR é a representação para subcláusula; *gap* é a indicação de que falta algo naquele espaço

relacionamento de coordenação. Por exemplo, a regra $SN(homem) \rightarrow SN(homem)\ C(e)\ SN(cão)$, teria a seguinte distribuição de probabilidade:

$$P_{nuc}(SN|SN(homem)) \times P_{esq}(STOP|SN, SN, homem)$$

$$P_{dir}(SN(cão), coord = 1|SN, SN, homem) \times P_{dir}(STOP|SN, SN, homem)$$

$$P_c(C, e|SN, SN, SN, homem, cão)$$

3.4 Conclusão

Neste capítulo foram detalhados dois métodos estatísticos propostos por Collins, evoluídos de forma a poder tratar alguns casos problemáticos encontrados pelo parser.

Capítulo 4

PAPO: um ambiente integrado e um parser probabilístico para língua portuguesa do Brasil

4.1 Introdução

O sistema PAPO consiste em um ambiente de aprendizagem para análise sintática automática (parsing) de sentenças do Português do Brasil. É composto por dois supermódulos (Figura 4.1), (i) o gerador de modelos, que é o módulo de aprendizado do sistema, recebendo como entrada uma base de exemplos corretos de análise sintática e gerando os modelos probabilísticos inferidos, e (ii) o de processamento, ou o parser propriamente dito, que, com base nos modelos gerados, executa a análise sintática de sentenças de entrada, num algoritmo de busca pela árvore de derivação sintática mais provável para uma dada sentença.

Ainda na Figura 4.1, pode-se observar o cenário de uso mais comum do PAPO, em que ainda é necessário um módulo de pré-processamento da base de exemplos para gerar uma entrada no formato esperado pelo gerador de modelos. O detalhamento do pré-processamento representado na figura corresponde ao que se realizou no caso específico dos experimentos desta tese e será melhor explicado no Capítulo 5.

Apesar de não estar representado diretamente na Figura 4.1 por ser conceitual, um item vital ao sistema é o modelo probabilístico de análise, subjacente tanto ao gerador de modelos, que o instancia, quanto ao parser, que aplica suas instanciações. No caso do

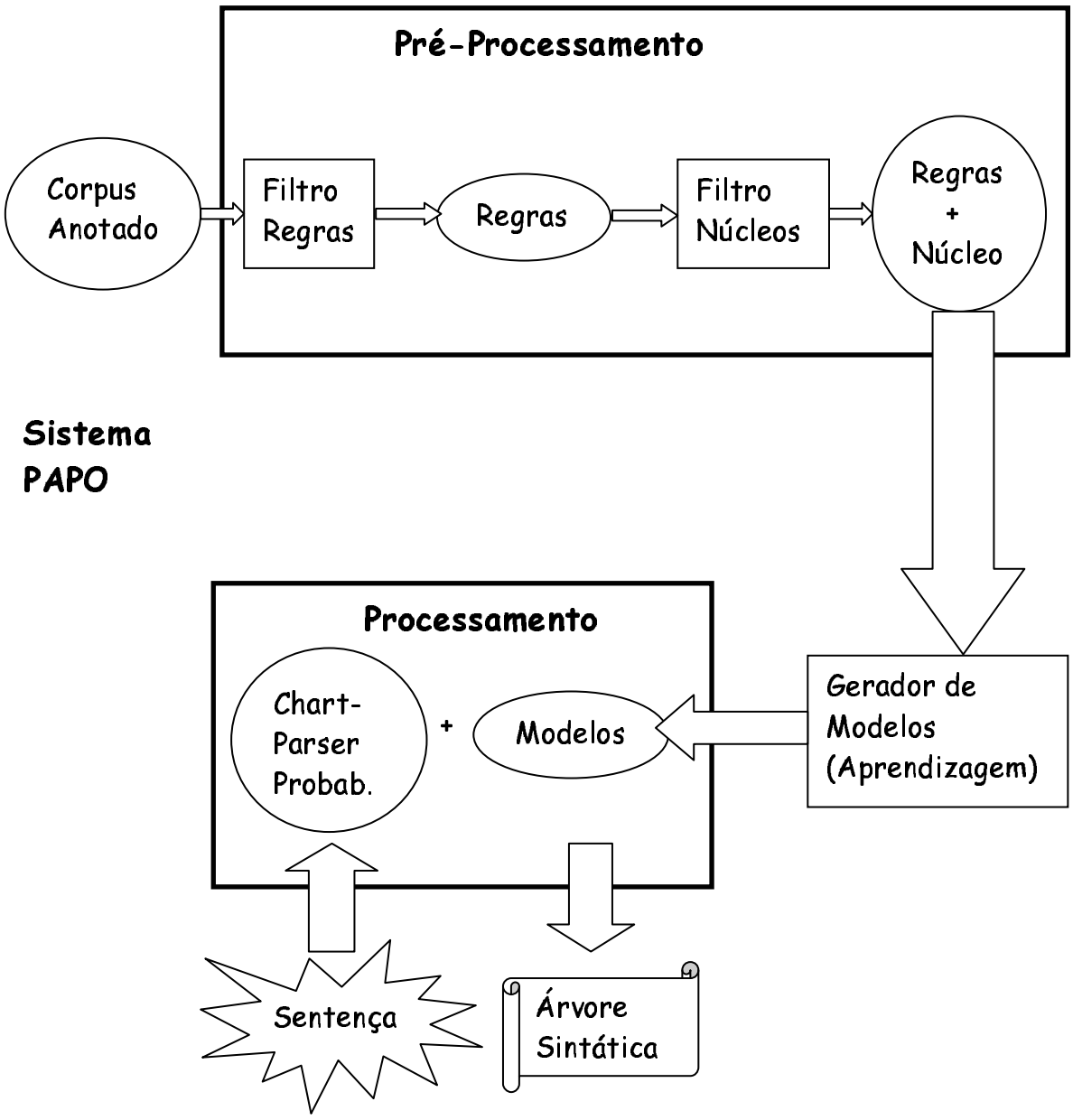

Figura 4.1: Sistema PAPO

PAPO, há dois modelos probabilísticos alternativos, a partir de agora referenciados como PAPO-I e PAPO-II. De certa forma, há, portanto, dois geradores de modelo, mas um único parser, que, não por coincidência, é capaz de aplicar a ambos. O PAPO-II surgiu em resposta aos resultados não tão animadores obtidos por PAPO-I nos experimentos realizados, que serão relatados no Capítulo 5. Ambos os modelos são descritos a seguir, antes de tratar-se de cada módulo.

4.2 Modelos Probabilísticos Subjacentes: PAPO-I e PAPO-II

Os modelos PAPO-I e PAPO-II em muito se assemelham ao Modelo 1 de Collins, salvo algumas diferenças, menores porém significativas, a saber: (i) simplificam o parâmetro distância, que passa a ser só a adjacência, e (ii) modelam seqüências de símbolos em regras n-árias, $n > 2$, como processos de Markov, o que os aproxima do Modelo 2 de Collins, no sentido de que também prevêem um mecanismo, simples que seja, para tentar garantir que um sintagma tenha certos elementos essenciais, adicionais ao núcleo.

Por serem probabilísticos, os modelos de análise do PAPO entendem árvores de derivação como resultado de seqüências de eventos aleatórios. No entanto, há entre eles uma diferença radical: os eventos de PAPO-I, assim como os de Collins, procedem a uma construção de cima para baixo, enquanto os do PAPO-II, de baixo para cima, na árvore sintática.

Por ser muito semelhante ao Modelo 1 de Collins, descrito no Capítulo 3, PAPO-I não será aqui detalhado, apenas PAPO-II. Basta ter-se em mente que ambos modelam a seqüência de símbolos em regras n-árias de forma completamente análoga. Bonfante e Nunes (2002) apresentam mais detalhes de PAPO-I.

Para auxiliar a descrição de PAPO-II, serão introduzidas, ao longo da discussão, algumas definições, as primeiras das quais são as seguintes:

- pred(M, P): retorna a subárvore que precede M no sintagma P, no sentido do núcleo para as extremidades.
- rot(T): rótulo (sintático) da raiz de T
- núcleo(T) = subárvore que é núcleo do sintagma T

- pai(T) = árvore de que T é filho direto
- headword(T):

  se T é folha **então**
  (palavra, etiqueta_morfossintática) de T
  senão
  headword(núcleo(T))
  fim se

- raiz(T) = (rot(T), headword(T))
- à_direita(T1, T2) = 1 sse o sintagma T1 está à direita de T2 na árvore considerada

A partir dessas definições, PAPO-II modela a construção das árvores como uma composição dos seguintes tipos de eventos aleatórios:

- folha(T)=f - evento de o rótulo sintático de um nó-folha T ser f;
- é_núcleo(T) - evento de a subárvore T ser núcleo de algum sintagma;
- rótulo_pai(T)=p - evento de o nó-pai de uma subárvore T ter rótulo sintático p;
- adj(T1, T2) - evento do sintagma T1 ser adjacente ao sintagma T2;
- modifica(M, P) - evento de M pertencer ao sintagma P como modificador, dado que ou isso é verdade ou M será um sintagma adjacente a P;
- próximo(M, P)=(l, r, próximoM, próximoN) - evento de, sendo M um modificador de P à esquerda/direita de seu núcleo N:

  i - sse l/r = 1, M ser o filho mais à esquerda/direita do sintagma P;
  ii - sse l/r = 0, o modificador seguinte a M (no sentido do núcleo para as extremidades) no sintagma P ter raiz com rótulo sintático próximoM;
  iii - sse r/l = 1 e M for adjacente a N, N é a extremidade direita/esquerda de P.
  iv - sse r = 0 e se M for adjacente a N pela esquerda, o primeiro modificador à direita do núcleo ter raiz com rótulo próximoN;

Este evento subsume o STOP de Collins, bem como será usado para estabelecer um processo de Markov na seqüência dos filhos (símbolos do corpo) de uma árvore (regra), no sentido do núcleo para as extremidades.

Dependências

Apresentados os eventos do PAPO-II, cabe estipular de que condições são dependentes para que seja possível estimar suas probabilidades.

- folha(T) é uma variável aleatória que depende exclusivamente do par (palavra, etiqueta_morfossintática) contido na folha T.
- é_núcleo(T) é uma variável aleatória que depende exclusivamente de raiz(T).
- rótulo_pai(T) é uma variável aleatória que depende exclusivamente de raiz(T).
- próximo(M, P) é uma variável aleatória que depende exclusivamente de rot(P), raiz(núcleo(P)), raiz(M) e modifica(M, P), adj(M, núcleo(P)) e à_direita(M, núcleo(P)).

Para auxiliar a especificação das dependências de modifica(M, P), fazem-se necessárias as seguintes definições:

- rot_esperado(M, P):

se M à esquerda de núcleo(P) **então**
  se próximo(M, P) = (0, _, próximoM, _) **então**
    retorna próximoM
  senão
    retorna STOP
  fim se
senão

  se próximo(M, P) = (_, 0, próximoM, _) **então**
    retorna próximoM
  senão
    retorna STOP
  fim se

fim se

- outro_esperado(M, P):

  se próximo(M, P) = (_, 0, _, próximoN) **então**
  retorna próximoN
  senão
  retorna STOP
  fim se

Informalmente, pode-se entender rot_esperado(M, P) como retornando o rótulo esperado para o modificador que vier a seguir M dentro do sintagma P. A função outro_esperado(M, P), por sua vez, será aplicada apenas a modificadores adjacentes a um núcleo pela esquerda e retorna o rótulo esperado para o modificador que vier a ser adjacente ao núcleo pela direita. O símbolo STOP é um rótulo jamais apresentado por qualquer árvore e é um artifício para implementar o fechamento de um sintagma.

- modifica(M, P) é uma variável aleatória que depende, sempre, de raiz(núcleo(P)), raiz(P), é_núcleo(M)=0, raiz(M) e à_direita(M, núcleo(P)) e, excepcional e adicionalmente, dos seguintes fatores:

  se M é adjacente à direita e há modificador M2 adjacente ao núcleo pela esquerda **então**
  outro_esperado(M2, P) = rot(M)
  fim se
  se M não é adjacente ao núcleo **então**
  rot_esperado(pred(M, P), P) = rot(M)
  fim se

- adj(T1, T2) é uma variável aleatória que, dado modifica(M, pai(T2)) e é_núcleo(T2), depende de rot(T1), rot(pai(T2)) e rot(T2).

A título de exemplo, considere o cálculo da probabilidade da árvore presente na Figura 4.2.

$P(S) = P(S|L2, L1, N1, R1, R2) \times P(L2) \times P(L1) \times P(N1) \times P(R1) \times P(R2)$

$P(L1) = P(folha(L1) = rot(L1)|headword(L1))$
$P(L2) = P(folha(L2) = rot(L2)|headword(L2))$

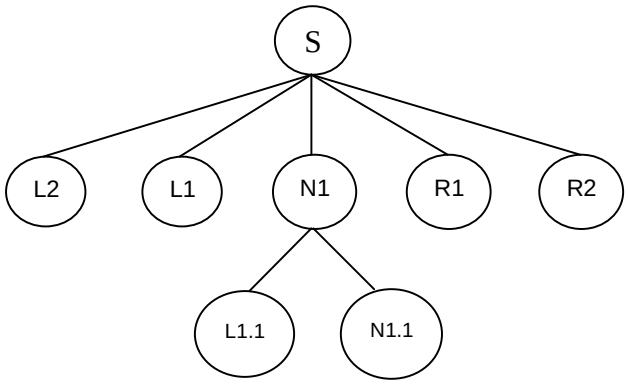

Figura 4.2: Exemplo de árvore sintática

$$P(N1) = P(N1|L1.1, N1.1) \times P(L1.1) \times P(N1.1)$$
$$\mathrm{P(R1)} = P(folha(R1) = rot(R1)|headword(R1))$$
$$P(R2) = P(folha(R2) = rot(R2)|headword(R2))$$
$$P(L1.1) = P(folha(L1.1) = rot(L1.1)|headword(L1.1))$$
$$P(N1.1) = P(folha(N1.1) = rot(N1.1)|headword(N1.1))$$
$$P(N1|L1.1, N1.1) =$$
$$P(\acute{e}\_n\acute{u}cleo(N1.1)|raiz(N1.1))\times$$
$$P(r\acute{o}tulo\_pai(N1.1) = rot(N1)|raiz(N1.1))\times$$
$$P(\acute{e}\_n\acute{u}cleo(L1.1) = 0|raiz(L1.1))\times$$
$$P(modifica(L1.1, N1)|rot(N1), raiz(N1.1), \acute{e}\_n\acute{u}cleo(L1.1) = 0,$$
$$raiz(L1.1), \grave{a}\_direita(L1.1, N1) = 0)\times$$
$$P(adj(L1.1, N1.1)|modifica(L1.1, N1), \acute{e}\_n\acute{u}cleo(N1.1),$$
$$rot(L1.1), rot(N1), rot(N1.1))\times$$
$$P(pr\acute{o}ximo(L1.1, N1) = (1, 1, \_, \_)|rot(N1), raiz(N1.1),$$
$$raiz(L1.1), modifica(L1.1, N1), adj(L1.1, N1), \grave{a}\_direita(L1.1, N1) = 0)$$

$$P(S|L2, L1, N1, R1, R2) =$$
$$P(\acute{e}\_n\acute{u}cleo(N1)|raiz(N1))\times$$
$$P(r\acute{o}tulo\_pai(N1) = rot(S)|raiz(N1))\times$$
$$P(\acute{e}\_n\acute{u}cleo(L2) = 0|raiz(L2))\times$$
$$P(\acute{e}\_n\acute{u}cleo(L1) = 0|raiz(L1))\times$$
$$P(\acute{e}\_n\acute{u}cleo(R1) = 0|raiz(R1))\times$$
$$P(\acute{e}\_n\acute{u}cleo(R2) = 0|raiz(R2))\times$$
$$P(modifica(L1, S)|rot(S), raiz(N1), \acute{e}\_n\acute{u}cleo(L1) = 0,$$
$$raiz(L1), \grave{a}\_direita(L1, N1) = 0)\times$$

$$
\begin{aligned}
&P(adj(L1,N1)|modifica(L1,S),é\_núcleo(N1),\\
&\qquad rot(L1),rot(S),rot(N1))\times\\
&P(rot\_esperado(L1,S)=rot(L2)|rot(S),raiz(N1),\\
&\qquad raiz(L1),modifica(L1,S),adj(L1,N1),à\_direita(L1,N1)=0)\times\\
&P(modifica(L2,S)|rot(S),raiz(N1),rot\_esperado(L1,S)=rot(L2),\\
&\qquad à\_direita(L2,N1)=0)\times\\
&P(adj(L2,N1)=0|modifica(L2,S),é\_núcleo(N1),\\
&\qquad rot(L2),rot(S),rot(N1))\times\\
&P(rot\_esperado(L2,S)=STOP|rot(S),raiz(N1),raiz(L2),\\
&\qquad modifica(L2,S),adj(L2,N1)=0,à\_direita(L2,N1)=0)\times\\
&P(outro\_esperado(L1,S)=rot(R1)|rot(S),raiz(N1),\\
&\qquad raiz(L1),modifica(L1,S),adj(L1,N1),à\_direita(L1,N1)=0)\times\\
&P(modifica(R1,S)|rot(S),raiz(N1),é\_núcleo(R1)=0,raiz(R1),\\
&\qquad outro\_esperado(L1,S)=rot(R1),à\_direita(R1,N1))\times\\
&P(adj(R1,N1)|modifica(R1,S),é\_núcleo(N1),rot(R1),rot(S),rot(N1))\times\\
&P(rot\_esperado(R1,S)=rot(R2)|rot(S),raiz(N1),raiz(R1),modifica(R1,S),\\
&\qquad adj(R1,N1),à\_direita(R1,N1))\times\\
&P(modifica(R2,S)|rot(S),raiz(N1),é\_núcleo(R2)=0,raiz(R2),\\
&\qquad rot\_esperado(R1,S)=rot(R2),à\_direita(R2,N1))\times\\
&P(adj(R2,N1)=0|modifica(R2,S),é\_núcleo(N1),rot(R2),rot(S),rot(N1))\times\\
&P(rot\_esperado(R2,S)=STOP|rot(S),raiz(N1),raiz(R2),\\
&\qquad modifica(R2,S),adj(R2,N1)=0,à\_direita(R2,N1))
\end{aligned}
$$

4.3 Gerador de Modelos

O gerador de modelos recebe como entrada uma base de todas as regras observadas na *treebank*, com repetição. Uma regra de uma *treebank* nada mais é que qualquer dupla (raiz(P), [raiz(F1), raiz(F2), raiz(F3), ..., raiz(Fn)]), onde P é um nó-pai qualquer observado e [F1, F2, F3, ..., Fn] a sua descendência direta, listada da esquerda para direita, e estritamente de cardinalidade > 1. Perceba que, implicitamente, cada regra terá seu núcleo identificado devido ao fato de que $headword(P) = headword(núcleo(P))$.

Dada uma base de regras, estas podem ser processadas para estimar as probabilidades do modelos PAPO-I e PAPO-II, visto ambos serem baseados na dicotomia núcleo-

modificador. Outro ponto comum a ambos os modelos e naturalmente herdados do modelo de Collins é que todos os eventos tratam, no máximo, da relação entre dois elementos: um modificador e um núcleo. Em específico, focalizam-se a raiz de um modificador, a raiz de um núcleo, o rótulo do sintagma-pai, a posição do modificador relativa ao núcleo (em termos de direção - esquerda/direita - e adjacência), se há um sucessor do modificador (incluindo aqui o excepcional e adicional "sucessor" à direita do modificador adjacente à esquerda) ou se este é um extremo do sintagma. Portanto, a entrada para a geração de ambos os modelos é naturalmente uma listagem única de todas as instâncias desse tipo de relacionamento observadas na *treebank*. Esse é exatamente o papel do script chamado "geraRelacoes.pl", gerar tal listagem a partir de todas as regras da *treebank* já com seus núcleos devidamente identificados. Além disso, não gera entradas duplicadas, antes, anota cada entrada com sua freqüência absoluta.

Por exemplo, a primeira regra da Tabela 4.1 gera as relações apresentadas na Tabela 4.2.

Tabela 4.1: Regras da *treebank*.

S(revela,v-fin) → SUBJ(ibama,prop) P(revela,v-fin) ACC(contrabando,n)
ACC(contrabando,n) → H(contrabando,n) N<(de,prp)
N<(madeira,de) → H:(de,prp) P<(madeira,n)

A partir desse ponto, os modelos PAPO-I e PAPO-II são gerados por processos distintos de estimativa de probabilidade, salvo quanto ao submodelo de adjacência. É interessante notar desde já que o resultado da geração de PAPO-I e PAPO-II tem formato idêntico, tanto que estes modelos são aplicados por um parser único, muito embora tenham significados bastante diferentes. As Tabelas 4.3 e 4.4 apresentam os submodelos componentes de PAPO-I e PAPO-II, respectivamente. Nessas tabelas, cada linha relaciona (i) uma base de dados (Submodelo), (ii) seu conteúdo (geralmente uma estimativa de probabilidade, sempre obtida pelo método ML, e excepcionalmente o mapeamento "possíveis-rótulos", explicado a seguir), (iii) a origem dos dados (Origem) a partir dos quais é calculado o conteúdo em questão e (iv) o script (Gerador) que realiza o cálculo propriamente dito. O mapeamento "possíveis-rótulos" retorna uma lista não-ranqueada de todos rótulos sintáticos observados para folhas com uma determinada headword.

Merece considerações adicionais a coluna "Origem" de ambas as tabelas, em especial as células contendo "rels" e "ukn". Em primeiro lugar, algumas bases de dados de origem são obtidas pela concatenação de duas outras, o que se denota pelo sinal de adição.

Tabela 4.2: Saída produzida pelo script "geraRelacoes.pl" para a regra "S(revela, n) → SUBJ(ibama, prop)P(revela, v-fin) ACC(contrabando, n)".

S(revela, v-fin)→ SUBJ(ibama, prop) P(revela, v-fin) ACC(contrabando, n)
relação 1: raiz(modificador): (SUBJ, (ibama, prop)) raiz(núcleo): (P, (revela, v-fin)) rot(pai): S dir: esquerda adj: 1 freq: 1 próximo(modificador, pai) = (1, 0, STOP, ACC)
relação 2: raiz(modificador): (ACC, (contrabando, n)) raiz(núcleo): (P, (revela, v-fin)) rot(pai): S dir: direita adj: 1 freq: 1 próximo(modificador, pai) = (0, 1, STOP, STOP)

A base "rels", por sua vez, é resultado da aplicação do script "geraRelacoes" à base de regras, com núcleo identificado, observadas na *treebank*. Para que o parser possa lidar com palavras desconhecidas, bem como levantar hipóteses de análise genéricas para classes morfossintáticas de palavras (em vez de apenas para itens lexicais específicos), essa base por vezes é complementada com dados relativos à palavra desconhecida ("ukn"), mas de etiqueta morfossintática conhecida.

As bases complementares em questão são "uknI" e "uknII", diretamente derivadas de "rels". Na geração de ambas, cada entrada de "rels" virtualmente corresponde a três novas entradas, cada qual substituindo ora um ora outro ora ambos os itens lexicais da relação pela string "ukn". Na realidade, não são geradas entradas duplicadas, ou seja, mais uma vez fazendo-se constar de cada entrada um valor de freqüência absoluta. A única diferença entre "uknI" e "uknII" está na obtenção desse valor: em "uknI" , cada entrada virtual traz a freqüência da entrada original, que é acumulada; em "uknII", considera-se que cada entrada virtual tem uma freqüência de 1, que também é acumulada. Considerou-se uknII como mais pertinente para estimar as probabilidades que envolvem a escolha do rótulo de um núcleo/pai, evento crucial no processamento do parser. Desejou-se, assim, evitar que a alta freqüência de uma determinada palavra acabasse por influenciar esse

passo, já que se sabe que uma palavra desconhecida qualquer não será definitivamente a palavra em questão e deverá se comportar como a média das palavras conhecidas, de forma independente de sua freqüência.

Pode-se notar ainda na Tabela 4.4 a falta de bases de dados estimando os valores $P(\acute{e}\_n\acute{u}cleo(T)|...)$ e $P(modifica(M,P)|...)$. De fato, para o caso do tratamento de conjunto de etiquetas sintáticas arbitrárias, isso seria estritamente necessário, mas implicaria o custo adicional de (i) um remodelamento do parser bem como (ii) a implementação de um procedimento excepcional para tratar "modifica", já que este não pode ser estimado a partir apenas das relações observadas (constantes da base "rels"), mas também a partir das relações potenciais da *treebank* que, por um motivo ou por outro, não chegam a se realizar. Entretanto, o conjunto de etiquetas específico usado nos experimentos desta tese, legado por Bick, apresenta características que praticamente dispensam os modelos faltantes. Em primeiro lugar, suas etiquetas morfossintáticas incorporam informação tanto de modificação quanto de direção de modificação, sempre indicado pelos sinais $<$ e $>$ (assim, por exemplo, >N/N< indica modificador pré/pós-nominal), o que supre a falta de "modifica". Além disso, verificou-se que jamais uma etiqueta morfossintática de modificador rotulava um núcleo e vice-versa, implicitando "*é_núcleo*". Como argumento final, basta lembrar que, por meio de seu conjunto de etiquetas, Bick representa árvores sintáticas inteiras e não-ambíguas como uma lista, sem qualquer estrutura, de pares (palavra, rótulo sintático).

4.4 O Parser

O parser PAPO é um chart parser[1] probabilístico e facilmente extensível para que se torne também heurístico, embora essa possibilidade não tenha ainda sido explorada. Seu projeto foi inspirado inicialmente na descrição de alto nível oferecida por Collins para seu parser, da qual poucos traços se encontram na versão final do PAPO. Muitas adaptações e inovações foram necessárias para lidar com o numeroso conjunto de etiquetas usado nos experimentos desta tese, bem como a implementação dos processos de Markov que modelam as seqüências de símbolos em regras n-árias. Isso porque PAPO baseia-se essencialmente num algoritmo de busca best-first (Russel e Norvig 1995) e ambos os fatores citados têm ascendência direta sobre a busca exponencial, notadamente a ordem de complexidade do algoritmo, que é exponencial para tempo e memória no tamanho da sentença de entrada.

[1] Para uma introdução ao chart parsing, consulte o Apêndice C.

Tabela 4.3: Estimativas ML dos submodelos do PAPO-I.

Submodelo	Conteúdo	Origem	Gerador
ModAdj	$P(adj(T1,T2)\|$ $modifica(T1,pai(T2)),$ $é\_núcleo(T2), rot(T1),$ $rot(pai(T2))\ e\ rot(T2))$	rels	modeloAdj.pl
ModNucleo	$P(rot(núcleo(P))\|$ $rot(P), headword(P))$	rels + uknII	modeloNucleo.pl
ModRel	$P(raiz(M)\|modifica(M,P),$ $rot(P), raiz(núcleo(P)),$ $adj(M,nucleo(P)),$ $à\_direita(M,núcleo(P)))\times$ $P(proximo(M,P)\|raiz(M),$ $modifica(M,P), rot(P),$ $raiz(núcleo(P)), adj(M,núcleo(P)),$ $à\_direita(M,núcleo(P)))$	rels + uknI	modeloRel.pl
ModLabel	$possíveis\_rótulos(headword(T)\|$ $é\_folha(T))$	regras sem núcleo	modeloLabel.pl

Tabela 4.4: Estimativas ML dos submodelos do PAPO-II.

Submod	Conteúdo	Origem	Gerador
ModAdj	$P(adj(T1,T2)\|$ $modifica(T1,pai(T2)),$ $é\_núcleo(T2),$ $rot(T1), rot(pai(T2))\ e\ rot(T2))$	rels	modeloAdj.pl
ModPai	$P(rot(pai(N))\|$ $raiz(N), é\_núcleo(N))$	rels + uknII	modeloPai.pl
ModProx	$P(proximo(M,P)\|$ $raiz(M), modifica(M,P),$ $rot(P), raiz(núcleo(P)),$ $adj(M,núcleo(P)),$ $à\_direita(M,núcleo(P)))$	rels + uknI	modeloProx.pl
ModFolha	$P(rot(T)\|$ $headword(T), é\_folha(T))$	regras sem núcleo	modeloFolha.pl

Instanciando a terminologia própria da solução de problemas por busca, cada nó do espaço de busca corresponde a um arco; nós finais, a arcos que cubram toda a sentença; e a expansão de um nó/arco, ao processo de se tentar combinar o arco em questão com

todos os seus vizinhos no chart, *de todas as formas possíveis*[2] gerando novos arcos a serem inseridos na agenda do algoritmo de busca.

A agenda do PAPO é ordenada pelo custo heurístico dos arcos a serem expandidos[3]. Atualmente, o custo heurístico de um arco é simplesmente definido como o inverso de sua probabilidade, portanto não fazendo qualquer previsão (heurística) do custo adicional apresentado pela melhor solução que inclua o arco em questão. Isso deixa PAPO à mercê da totalidade da exponenciação (no que tange à ramificação da árvore de busca) presente no modelo probabilístico.

Combinação e Expansão de Arcos

A combinação entre dois arcos adjacentes A e B, denotada genericamente por $A <? B$, pode se dar de duas formas diferentes:

- **inauguração:** se A e B são arcos completos, então a combinação inaugura um novo sintagma, denotado por $A < B$, com A como núcleo, possivelmente ainda incompleto em uma e/ou outra extremidade. Nesse caso, já ficam determinados que tipos de sintagma são esperados nas extremidades receptoras;

- **extensão:** se A está incompleto e receptivo a B e este está completo, então a combinação cria um novo *arco* (não um novo *sintagma*), denotado por $A < +B$, que representa a inclusão no sintagma A do novo modificador B. Neste momento, fica determinado se $A < +B$ está completo pelo lado de B ou se passa a esperar um outro modificador específico seguinte a B.

A combinação de um par específico de arcos ocasionará ou inauguração ou extensão, nunca ambas. Mesmo assim, dado um único par de combinantes A e B, a inauguração/extensão é uma operação não-determinística, cujo conjunto de todos os possíveis resultados - ou *fechamento*, denotado por $\{A < B\}/\{A < +B\}$ ou, genericamente, $\{A <?B\}$ - costuma ser numeroso. Acontece que é exatamente o cálculo de diversos $\{A <?B\}$ o que importa à operação de expansão, que pode, pelo menos por ora, ser definida assim:

$$expansão(A) = \bigcup_{adj(A,B)}(\{A <?B\} \cup \{B <?A\})$$

O fechamento da combinação de dois arcos é determinado a partir dos dados do submodelo ModRel/ModProx (PAPO-I/II), que contém todo tipo de relação de modificação observado na *treebank* para um par de arcos quaisquer.

[2] Como será descrito em seguida, a combinação é uma operação não-determinística.
[3] Isso caracteriza PAPO, pelo menos formalmente, como um algoritmo de busca A*.

Detalhes de implementação

Para (i) facultar alguns experimentos preliminares com PAPO ainda nessa situação, (ii) permitir a própria especulação acerca de possíveis heurísticas e (iii) embasar uma implementação futura bastante eficiente quando heurísticas adequadas forem formuladas, enfatizou-se, no projeto do parser, medidas para racionalizar o uso dos recursos disponíveis (memória e tempo).

Dando continuidade a esta subseção, serão descritas as medidas de racionalização tomadas e, por fim, discutir-se-á como PAPO aplica as generalizações inferidas para classes morfossintáticas.

Racionalização do Uso da Memória

Dado o grande número de arcos comumente gerados na análise de qualquer sentença, todos em geral devendo ser mantidos em memória até o fim do processamento, tentou-se otimizar o armazenamento de cada arco por meio de compartilhamento de dados, quando possível. Uma boa oportunidade para estabelecer o compartilhamento surge quando da combinação de dois arcos, na expansão. Fato é que, dado um par de combinantes quaisquer A e B, todos os elementos de $\{A <?B\}$ coincidem nos mesmos 50% de seus dados, pelo menos. Por esse motivo, decidiu-se explorar essa coincidência, como mostra a Figura 4.3, que representa três diferentes sintagmas "o gato"(ACC, SUBJ e P<, respectivamente, objeto direto, sujeito e complemento de preposição) inaugurados ao se combinar um modificador pré-nominal de raiz (>N, o, ART) com um núcleo nominal de raiz (H, gato, N).

Como se pode observar, cada arco virtual tem seus dados distribuídos entre duas estruturas: uma da classe *Edge::Shared*, contendo os dados compartilhados, e outra da classe *Edge*, contendo os demais dados, bem como uma referência à instância de *Edge::Shared* que lhe cabe. Apresenta-se, ainda na Figura 4.3, um outro nível de compartilhamento implementado. Trata-se do compartilhamento de dados de headword (que não se restringem ao par (palavra, etiqueta_morfossintática)) entre arcos de headwords coincidentes.

Racionalização do Fator de Ramificação Efetivo

Outra preocupação para racionalizar o uso de recursos pelo parser foi tentar minimizar a priori, à revelia da heurística aplicada, o fator de ramificação efetivo da busca. Duas medidas foram tomadas que atacam esse fator, a saber: (i) evitar a inclusão de árvores duplicadas no chart, o que ocasionaria um fator de ramificação efetivo maior que

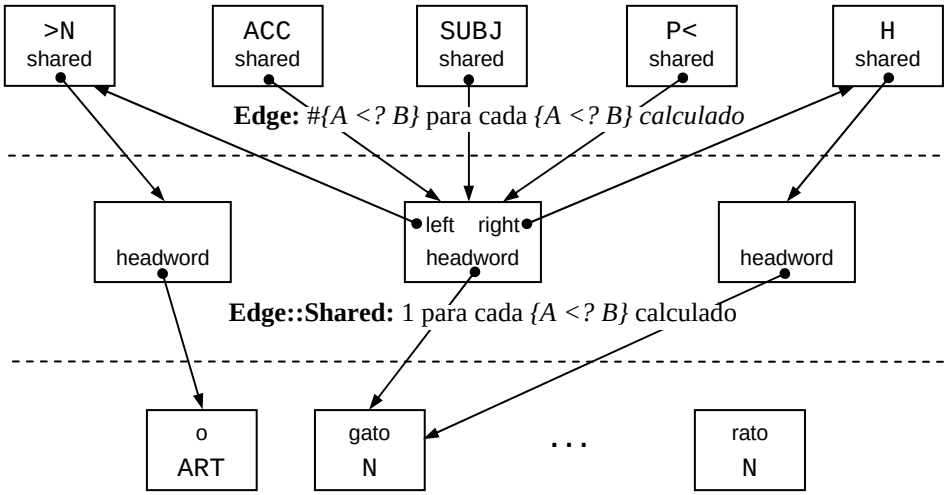

Figura 4.3: Compartilhamento de memória entre arcos.

o inerente ao modelo de entrada e acaba sendo tentado especialmente devido a arcos parciais gerados na aplicação de regras n-árias, n > 2, e (ii) inserção *lazy* (preguiçosa ou postergada) de arcos na agenda/chart, o que, em conjunto com uma boa heurística, pode reduzir drasticamente o fator de ramificação efetivo (objetivo último do algoritmo A*).

A inserção lazy de arcos é implementada pela *lateralização* e *serialização* da expansão. A lateralização se refere à seguinte revisão da definição de expansão:

$$expansão(A) = \bigcup_{adj(A,B)}(\{A <?B\} \cup comb_condicional(B, A))$$

onde $comb_condicional(X, Y)$ vale $\{X <?Y\}$, se X tiver saído da agenda antes de Y entrar, ou é vazio, caso contrário. Ou seja, devem-se considerar apenas os casos em que o arco expandido apareça como operando direito da combinação, a menos que essa restrição acabe por impedir definitivamente o surgimento dos arcos por ela inibidos. A idéia por trás dessa nova definição é a de que, se o vizinho inibido não for tão ruim, logo sairá da agenda para funcionar como operando direito.

Por sua vez, a serialização da expansão (já lateralizada) refere-se a (i) ranquear os elementos de uma expansão por mérito (ordem crescente de custo heurístico), (ii) fazer cada elemento da expansão referenciar seu sucessor no ranking e (iii) inserir, na agenda e no chart, apenas o elemento de maior mérito. Daí, para todo arco A saído da agenda, deve-se não só expandir A da mesma forma, como também inserir, na agenda e no chart, seu sucessor. O aspecto da agenda resultante está representado na Figura 4.4.

Racionalização do Tempo de Busca

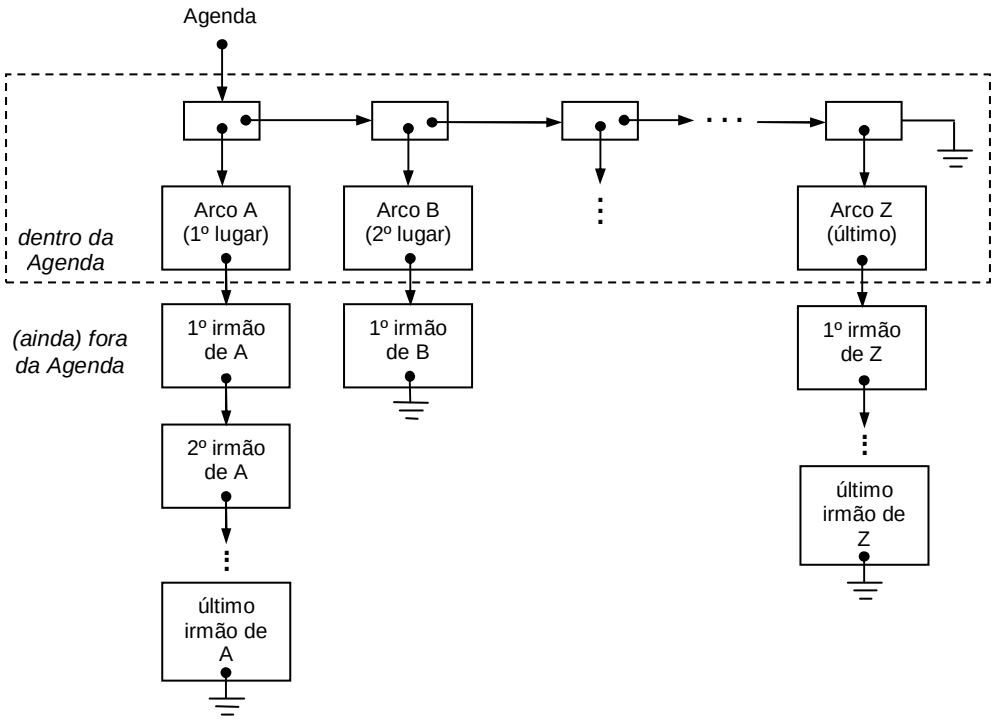

Figura 4.4: Serialização da expansão.

Tanto para expandir um arco quanto para efetivamente calcular a probabilidade dos novos arcos gerados numa expansão, notadamente operações realizadas maciçamente pelo parser, todas as bases de dados (submodelos) montadas pelo gerador de modelos são, ao menos conceitualmente, consultadas. Conseqüentemente, tempo de busca é um fator crítico no desempenho do parser e deve ser minimizado. Para tanto, recorreu-se a três expedientes: (i) codificação de rótulos sintáticos e headwords como chaves numéricas, para agilizar comparações e até permitir a implementação de tabelas de associação como simples arrays acessados com o valor das chaves; (ii) unificação de todos os submodelos numa estrutura de dados única (modelo unificado) que disponibiliza simultaneamente todas as informações necessárias num dado momento, favorecendo, inclusive, (iii) a realização de buscas parciais, ou seja, que ficam a meio caminho do objetivo final, em certas paradas estratégicas, para servir de ponto de partida para novas buscas (parciais ou não) e assim evitar percursos redundantes na estrutura de dados.

Um esquema da estrutura do modelo unificado está presente na Figura 4.5. O array *Model* é a porta de acesso ao modelo unificado e projeta cada headword nuclear (codificada) em uma *SynTbl*, tabela que multiplexa as informações relativas a cada possível acepção sintática nuclear que essa headword pode assumir. Ou seja, dada uma headword, essa tabela projeta um dado rótulo sintático (codificado) num *SynInfo*, um registro que

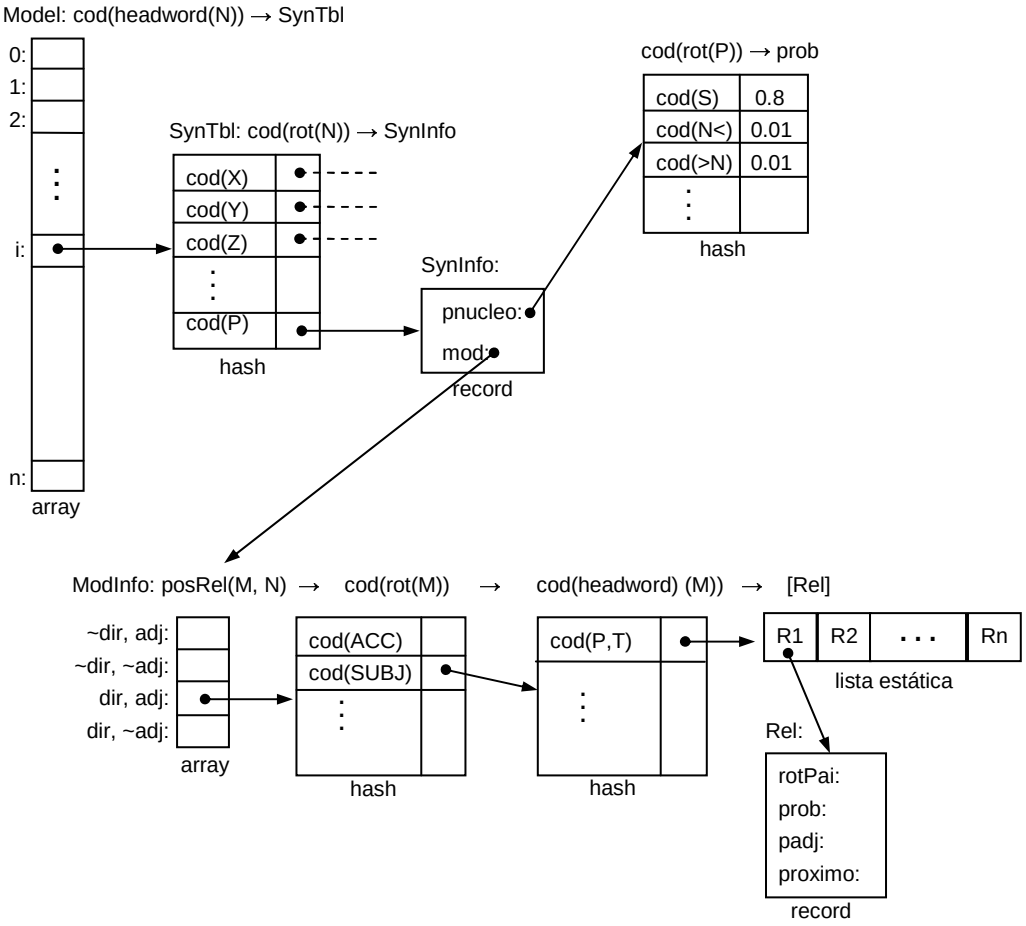

Figura 4.5: Estrutura do modelo unificado.

reúne tudo o que é necessário saber dada a raiz de um núcleo N. No campo *pnucleo/ppai* (PAPOI/II), tem-se acesso às informações do submodelo ModNucleo/ModPai aplicáveis ao núcleo em questão. O campo *mod* de *SynInfo* inicia uma seqüência de acesso que, dada a posição relativa de um modificador M ao núcleo N ($posRel(M, N)$), projeta raiz(M) na lista[4] de estruturas (*Rels*) informativas de todas as relações que se podem estabelecer entre M e N. Cada uma dessas estruturas se refere a uma possível relação, informando (i) o rótulo do pai envolvido (*rotPai*), (ii) a probabilidade proveniente do submodelo ModRel/ModProx (*prob*), (iii) a probabilidade de adjacência proveniente de ModAdj (*padj*) e (iv) o resultado do evento "próximo" a ser assumido numa combinação.

Por fim, a Figura 4.6 exemplifica a busca parcial, não só explorada nas situações representadas. Como se pode observar, toda headword já referencia diretamente duas *SynTbl*s - uma que lhe é específica e outra que compartilha com todas as palavras de

[4]É a partir dessa lista que se calcula $\{M <?N\}$.

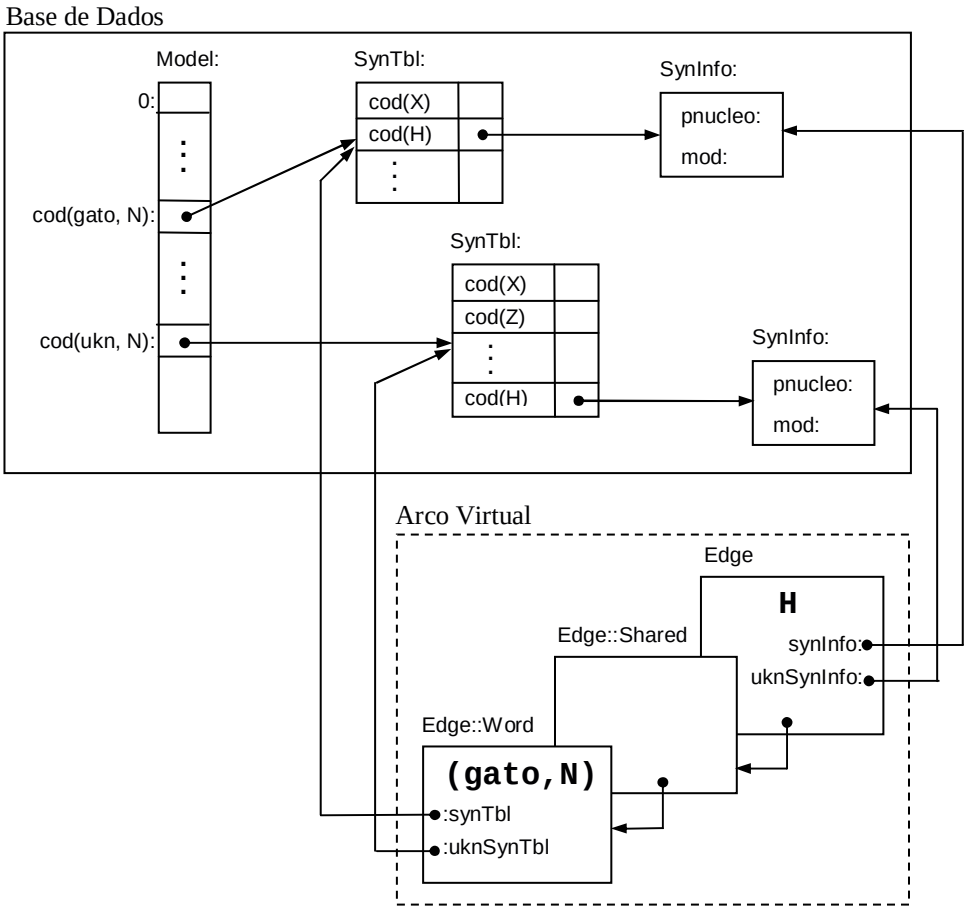

Figura 4.6: Exemplos de busca parcial.

mesma classe morfossintática (acessada pela palavra artificial "ukn") - em dois acessos ao modelo unificado que nunca serão repetidos no processamento de uma mesma sentença. Analogamente, cada arco referencia diretamente os *SynInfo*s que lhe cabem, obtidos pela projeção de seu rótulo sintático (codificado) pelas *SynTbl*s de sua headword. Seu campo "uknSynInfo" aponta, portanto, para todas as generalizações inferidas para a classe morfossintática de sua headword, que, como será explicado abaixo, também são usadas quando da expansão do arco.

Emprego das Generalizações para Classes Morfossintáticas

Devido a um tratamento especial realizado pelo gerador de modelos e já descrito anteriormente, o acesso a qualquer dos submodelos para uma headword (ukn, etiqueta_morfossintática) resulta em valores generalizados para etiqueta_morfossintática. Por meio disso, PAPO é capaz de lidar com sentenças contendo palavras jamais observadas na

treebank[5]. No entanto, essa não é a única utilidade dessas generalizações, que devem servir também para suprir a falta de dados para palavras conhecidas mas raramente aplicadas na *treebank*.

Para isso, PAPO permite que se associe a cada headword W conhecida um fator específico de confiança c(W), que representa a probabilidade de que os dados do modelo para W levarão à análise correta de qualquer sentença de entrada que contenha W. Esse valor é usado da seguinte forma para calcular os elementos de $\{A <?B\}$ e suas probabilidades, dados dois combinantes A e B quaisquer:

- Rslt $\leftarrow \emptyset$
- inicialize uma lista RelList para receber pares $(Rel, PFinal)$, onde Rel é uma possível relação a ser estabelecida entre A e B e $PFinal$ a probabilidade final da combinação que resultará de Rel. A função de inserção dessa lista é especial, entretanto, e não permite entradas com relações duplicadas. No caso da tentativa de inserção de um par contendo uma relação já inserida, essa função simplesmente acumula a probabilidade do par de entrada na do par pré-existente;
- para cada (X, Y, C) em[6]
  - $(A, B, c(A) * c(B))$
  - $(ukn(A), B, (1 - c(A)) * c(B)))$
  - $(A, ukn(B), c(A) * (1 - c(B)))$
  - $(ukn(A), ukn(B), (1 - c(A)) * (1 - c(B)))$
- faça
  - Rels $\leftarrow$ relações previstas no modelo para a combinação $X <?Y$
  - para cada R em Rels, insira em RelList o par $(R, C * P(R))$
- para cada $(Rel, PFinal)$ em RelList faça
  - NovoArco $\leftarrow$ uma combinação de A com B implementando Rel
  - NovoArco.prob $\leftarrow$ PFinal

[5] Os campos *synTbl* e *uknSynTbl* de uma headword desconhecida coincidem, bem como os campos "synInfo" e "uknSynInfo" de um arco com headword desconhecida. Nesse caso, os campos *ukn** ficam desabilitados no processo de expansão.

[6] Denota-se por ukn(X) o próprio arco X com headword considerada desconhecida.

– Rslt $\leftarrow$ Rslt $\cup$ {NovoArco}

Tendo em vista a descrição dos modelos probabilísticos aqui implementados, bem como a forma na qual o parser PAPO lida com problemas como o tempo de busca, a memória utilizada e o tratamento de dados esparsos, resta agora, avaliar seu comportamento na análise de sentenças da língua portuguesa do Brasil. No Capítulo 5 é apresentada essa avaliação, ainda que preliminar, mas muito necessária nessa fase de identificação de problemas.

Capítulo 5

Avaliação Preliminar do PAPO

Nesta tese, não se chegou a realizar uma avaliação abrangente e robusta do PAPO, em nenhum de seus modelos. Preferiu-se realizar, num primeiro momento, uma investigação preliminar qualitativa do desempenho do sistema, com o intuito de identificar problemas mais aparentes que surgissem na análise de um conjunto seleto de sentenças, incluindo algumas de mais difícil análise. Foi exatamente essa avaliação preliminar que motivou o surgimento de PAPO-II, que, salvo em um único caso de teste, jamais teve desempenho inferior a PAPO-I, na verdade superando-o significativamente na maioria dos casos.

Entretanto, não se pode comprovar a superioridade de PAPO-II, considerando-se o volume muito pequeno de casos de teste e que também, nessa avaliação preliminar, foram identificados (i) problemas sérios de ruído na base de regras (provavelmente inserido no pré-processamento da treebank, e não originário desta) e (ii) características do modelo de análise sintática dessa treebank que virtualmente impedem um aprendizado efetivo por parte do PAPO (I ou II) e que poderiam ser neutralizadas de forma relativamente fácil por meio de um pré-processamento, ainda automático, mais elaborado. Uma vez sanadas essas deficiências da base de exemplos, não se pode garantir que a disparidade entre o desempenho de PAPO-I e PAPO-II será mantida.

O ponto alto deste capítulo consiste, portanto, numa análise qualitativa do desempenho de PAPO-I e PAPO-II instanciados para uma mesma treebank. Adicionalmente, inclui-se uma análise quantitativa dos resultados obtidos com base nas medidas de precisão e cobertura.

Em todos os experimentos realizados nesta tese, o sistema PAPO utiliza, como fonte de exemplos de análise, o CETENFolha, um corpus de cunho jornalístico anotado com

o parser simbólico e o esquema de anotação propostos e descritos por Eckhard Bick, em (Bick 2000). Como esse esquema de anotação tem características bastante peculiares que tornam o pré-processamento não-trivial, a ponto de inclusive ser uma fonte significativa do ruído existente na entrada do gerador de modelos, apresenta-se primeiramente uma subseção sobre a treebank original e seu pré-processamento.

5.1 Alimentando PAPO

A anotação usada por Bick é baseada num paradigma de gramática de dependência. A anotação, plana, é enriquecida com a adição de marcadores de direção (> e <) e etiquetas de forma e função (a Tabela 5.1 mostra um exemplo de sentença com essa anotação[1]), o que permite a transformação posterior para árvores de constituintes. Com alguns scrips em Perl, na fase de pré-processamento, os grupos e as fronteiras dos constituintes (sintagmas) são identificados, delimitados, a parentização rotulada (etiquetada) de uma forma mais complexa (usando constituintes mais gerais, como: np - sintagma nominal, pp - sintagma preposicional, etc) e, finalmente, cada constituinte recebe uma etiqueta de função, derivada da etiqueta sintática de seu núcleo.

Tabela 5.1: Um exemplo do sistema de anotação do parser de Bick para a sentença *Erros abalam credibilidade da imprensa*

$SUBJ : n(MP)Erros$
$P : v - fin(PR\|3P\|IND)abalam$
$ACC : np$
$= H : n(FS)credibilidade$
$= N< : pp$
$== H : prp(<sam->)de$
$== P< : np$
$===> N : art(<-sam> FS)a$
$=== H : n(FS)imprensa$

A estrutura de constituintes em forma de regras obtida do pré-processamento serve então, como base para a geração do modelo estatístico, segundo as propostas de parametrização descritas a seguir.

[1] em que SUBJ = sujeito; n = nome; M = masculino; P = plural; P= predicado; v-fin = verbo finito; PR = tempo passado; 3P = terceira pessoa; IND = indicativo; ACC = objeto direto acusativo; np = sintagma nominal; H = núcleo; F = feminino; S = singular; N pré/pós nominal; pp = sintagma preposicional; prp = preposição; $< sam- >$ = primeira parte de uma fusão morfológica de uma palavra (de); art = artigo.

O mecanismo de busca pela estrutura sintática da sentença é um chart parser probabilístico, que recupera a árvore partindo das palavras, ligando-as em seus constituintes maiores até que se recupere todo o seu eixo principal. O que o difere de um chart parser tradicional é que a versão probabilística não possui um conjunto de regras previamente codificadas, e sim, utiliza as informações do modelo estatístico produzido.

As seções seguintes apresentam uma descrição da anotação sintática do CETENFolha e em seguida a arquitetura do sistema PAPO com mais detalhes.

5.1.1 A anotação do CETENFolha

O CETENFolha foi anotado com o parser "Palavras", proposto por Bick (2000), um parser simbólico que segue o esquema das gramáticas de restrições, introduzido por Karlsson em (Karlsson 1990).

A gramática de restrições consiste numa abordagem com facilidades de desambiguação por estabelecer regras pelas quais uma palavra pode ser escolhida com base no contexto ao qual se encontra. Por exemplo, para a sentença "nunca como peixe", a entrada na gramática, após passar pelo analisador morfológico seria:

1. Entrada[2]:
   "<nunca>"
   "nunca" ADV
   "<como>"
   "como" <rel> ADV
   "como" <interr> ADV
   "como" KS
   "como" <vt> V PR 1S VFIN
   "<peixe>"
   "peixe" N M S
   "<.>"

2. A palavra "como" tem 4 etiquetas diferentes. Neste tipo de situação, uma regra de restrição para desambiguar a anotação seria:

[2] ADV = advérbio, KS = conjunção subordinativa, V = verbo, N = nome, PR = tempo presente, S = singular, M = masculino, 1 = primeira pessoa, VFIN = verbo finito, <rel> = relativo, <interr> = interrogativo, <vt> = monotransitivo

SELECT(VFIN)IF(NOT*-1 VFIN)(NOT*1 VFIN)[3]

3. Com essa ambigüidade sintática desfeita, o sistema pode usar a anotação produzida para etiquetar a sentença sintaticamente, na forma[4]:
   "<nunca>"
   "nunca" ADV @ADVL
   "<como>"
   "como" <vt> V PR 1S VFIN @FMV
   "<peixe>"
   "peixe" N M S @SUBJ @ACC @SC @OC
   "<.>"

4. Ao adicionar as etiquetas sintáticas para a sentença, restou ainda uma ambigüidade na anotação da palavra "peixe", que pode ser desambiguada retirando as outras etiquetas e deixando apenas a correta, que é a de objeto direto (@ACC), usando as seguintes regras:
   REMOVE(@$SUBJ$)IF(0 N)(NOT*-1 V3)(NOT*1 V3)[5]
   REMOVE(@SC)IF(NOT*-1<vK>)(NOT*1<vK>)[6]
   REMOVE(@OC)IF(NOT*-1@ACC)(NOT*1@ACC)[7]

Essa característica do uso incremental das regras, com contextos definidos e regras seguras, faz com que o parser sempre produza algum resultado, e seja robusto diante de construções incompletas, recebendo pelo menos uma instrução que sobreviverá dentre as muitas restrições. Uma gramática de boa qualidade para o nível morfológico contém de 1000 a 2000 regras.

A política usada pela gramática de restrições é que ela usa um poderoso mecanismo de desambiguação não por construir uma estrutura específica para uma sentença, mas por descartar tudo aquilo que não pode fazer parte dessa estrutura. Tradicionalmente, a gramática de restrição nasceu de analisadores morfológicos e, sendo assim, suas informações são restritas às fronteiras das palavras. Como conseqüência disso, o que se

[3]Selecione a forma VFIN se não há (daí o NOT) à esquerda (*-1) nem à direita (*-1) outra palavra que possa ser VFIN.

[4]@ADVL = sintagma adverbial, @FMV = verbo principal finito; @SUBJ = sujeito, @ACC = objeto direto, @SC = complemento do sujeito, @OC = complemento do objeto.

[5]Descarte a etiqueta @$SUBJ$ se a palavra é um nome e se não há um verbo na 3a. pessoa.

[6]Descarte a etiqueta complemento do sujeito(@SC) se não há um verbo de ligação (<vK>) na sentença.

[7]Descarte o complemento do objeto (@OC) se não houver um objeto direto (@ACC) na sentença.

obtém é uma gramática "plana", horizontal. O parser "Palavras" faz uso desse tipo de representação para descrever a estrutura sintática das sentenças.

Seu conjunto de etiquetas contém 14 classes principais de categorias de palavras que combinam com 24 etiquetas para categorias de inflexão (vide Apêndice A para o conjunto completo de etiquetas). A descrição sintática contém informações tanto sobre função sintática (ex: @$SUBJ$ e @ACC) quanto sobre estrutura de constituinte (forma sintática). Essa estrutura de constituinte é representada através dos marcadores de dependência (< e >), que apontam para o núcleo da unidade sintática ao qual pertence, formando uma espécie de fronteira, que delimita todos os constituintes na sentença. Quando um núcleo não for o verbo principal, ele será marcado no ponto da dependência (ex: N para núcleo nominal, A para núcleo do adjunto[8]). Se há uma etiqueta de função (ex: @SUBJ, @ADVL, @N<PRED), o marcador de dependência será ligado àquela etiqueta. Em casos nos quais a função é incluída no status do modificador, o marcador é deixado sem etiqueta (ex: @>N para modificadores prenominais).

Dessa forma, cada palavra necessita lembrar apenas sua relação de dependência ascendente (isto é, a palavra da qual ela é dependente), e toda a estrutura sintática da sentença pode ser descrita localmente (na forma de etiquetas nas fronteiras das palavras). Por exemplo, na sentença da Tabela 5.2, a palavra *muito* "sabe" que seu adjunto adverbial (@>A) está ligado a *velho*. Este, por sua vez, está ligado ao seu ascendente esquerdo como um pós-nominal (@N<) a *castelo*. *Castelo*, por sua vez, sabe que é um objeto direto (@ACC) de um verbo principal à esquerda (<), *temos*, que funciona como raiz, ou núcleo principal, da sentença.

Tabela 5.2: A anotação da sentença *Temos neste país uns castelos muito velhos.*

Temos	[ter]<vt> V PR 1P IND VFIN	@FMV
em	[em]<sam-> PRP	@<$ADVL$
este	[este]<-sam><dem> DET M S	@>N
país	[país]<top>N M S	@P<
uns	[um]<art> DET M P	@>N
castelos	[castelo]<hus> N M P	@<ACC
muito	[muito]<quant> ADV	@>A
velhos	[velho] ADJ M P	@N<

[8] De acordo com essa anotação, os núcleos de adjuntos são núcleos de sintagmas adjetivos "Aps" ou de sintagmas adverbiais "ADVP".

5.1.2 Módulos de Pré-Processamento do CETENFolha

Para treinar o PAPO com o corpus CETENFolha foi necessário proceder a uma preparação dos dados originais para o formato de entrada do Gerador de Modelos, por meio dos dois módulos de pré-processamento descritos a seguir: o filtro de regras e o filtro de núcleos.

5.1.2.1 Filtro de Regras

O módulo de filtro de regras é responsável por pegar as sentenças do CETENFolha e transformá-las em regras, para que possam ser usadas na geração dos modelos desta tese. Como exemplo, considere a sentença *Ibama revela contrabando de madeira*, segundo a anotação de Bick, tal como mostrado na Tabela 5.1[9], e enriquecida com os marcadores de direção (> e <), mostrado na Tabela 5.3. Alguns scripts são acionados para que ocorra o processo de transformação. Através do script "tradlongline.pl", de autoria de E. Bick, uma sentença inteira com todas as suas palavras e etiquetas é transformada em uma longa linha de texto.

Tabela 5.3: Exemplo de anotação para a sentença: *Ibama revela contrabando de madeira.*

ibama	[Ibama] PROP M/F S	@$SUBJ>$
revela	[revelar] <fmc> V PR 3S IND VFIN	@FMV
contrabando	[contrabando] N MS	@$<ACC$
de	[de] PRP	@$N<$
madeira	[madeira] N FS	@$P<$

A seguir, através do uso do script "brackets.pl", de autoria de E. Bick, são identificadas as fronteiras dos grupos e cláusulas na descrição plana, delimitando-se os constituintes, e então marcando-os com marcadores de fronteiras e formas mais complexas (por exemplo, np, pp, icl, etc.), e finalmente atribuindo a cada constituinte complexa uma etiqueta de função derivada da etiqueta sintática de seu núcleo. O formato de saída do "brackets.pl" pode ser visto na Tabela 5.4, na qual a etiqueta "H" representa o núcleo do sintagma.

Os próximos a serem usados são o "formaregras" e o "pegapartesregras". O script "formaregras.pl" pega os dados de saída do "brackets" e transforma num formato visto

[9]Em que PROP = nome próprio; M = masculino; F = feminino; S = singular; @$SUBJ$ = sujeito; V = verbo; PR = presente; 3S = 3a. pessoa singular; IND = indicativo; @ACC = objeto direto; @N = adjeto adnominal; @P = predicado; @FMV = verbo principal; v-fin = verbo finito; N = nome; PRP = preposição;

Tabela 5.4: Saída produzida pelo script "brackets.pl".

@SUBJ:(np)
|- @H:prop ibama
@FMV:v-fin revela
@ACC:(np)
|- @H:n contrabando
|- @N<:(pp)
|-|- @H:prp de
|-|- @P<:n madeira

nas regras da Tabela 5.5. Como última fase desse processo, o script "pegapartesregras.pl" pega as informações (anotação sintática - AS, anotação morfossintática AM, palavra PAL) de cada elemento que compõe a regra. Todas as palavras são transformadas em minúsculas e a saída produzida no formato LHS*/*RHSs, ou seja, ASpai\|AMpai\|PALpai */* AS$filho_1$\|AM$filho_1$\|PAL$filho_1$ */* ... */* AS$filho_n$\|AM$filho_n$\|PAL$filho_n$, em que LHS é o pai da regra e RHS, seus filhos.

Tabela 5.5: Saída produzida pelo script "formaregras.pl".

S → SUBJ:prop ibama P:v-fin revela ACC:(np)
ACC(np) → H:n contrabando N<(pp)
N<(pp) → H:prp de P<:n madeira

5.1.2.2 Filtro de Núcleos

No módulo *filtro de núcleos*, os núcleos são identificados para cada regra. Por exemplo, para cada um dos sintagmas acima (S, ACC, e $N<$) são identificados seus núcleos, conforme mostrado na Tabela 5.6. No caso, para preencher o núcleo do sintagma ACC é necessário que seja recuperada a regra na qual ele é pai. Conseqüentemente, se esta não possuir núcleo, este tem que ser identificado. Esse processo é recursivo para cada um dos sintagmas que não possuem núcleos. Os núcleos para cada regra são deterministicamente atribuídos, através de um conjunto de templates com prioridades de ocorrências.

Tabela 5.6: Núcleos identificados para a sentença "Ibama revela contrabando de madeira".

S = revela,v-fin
ACC = contrabando,n
N< = de,prp

5.2 Experimentos com PAPO

Os casos de teste a que PAPO-I e II foram submetidos (Tabela 5.7) provêm de (Luft 1994) e são sentenças absolutamente inéditas no sentido de não terem sido observadas na treebank. Antes de serem processadas pelo parser, de acordo com o modelo de análise do PAPO, essas sentenças foram anotadas morfossintaticamente com as tags da treebank[10]. Para cada sentença, configurou-se PAPO para que obtivesse, no máximo, as dez análises mais prováveis encontradas. Caso PAPO-I ou II não terminasse a análise de uma sentença no prazo máximo de cinco minutos, considerou-se que não havia solução naquele caso.

Os resultados dos experimentos são apresentados a seguir sob três pontos de vista: em primeiro lugar, fazem-se algumas considerações sobre o tempo de resposta do parser; em seguida, uma análise quantitativa resumida; e, por fim, a análise qualitativa, foco deste capítulo.

5.2.1 Considerações sobre o Tempo de Resposta

Na Tabela 5.8 é especificado o tempo de resposta de PAPO-II para cada um dos casos de teste. Apenas se trata de PAPO-II aqui, primeiro por ser em geral mais rápido e segundo por pretender-se apenas dar uma idéia da seriedade do problema da explosão combinatória, que já se faz sentir em sentenças tão curtas e não é um mérito ou falha de nenhum dos dois modelos. Antes, trata-se da deficiência do algoritmo de busca que não usa ainda nenhuma heurística para racionalizar sua complexidade de tempo e memória no caso médio ou desistir da busca por uma solução. Este problema já foi melhor discutido no capítulo anterior, na descrição do parser, e terá projetos de solução propostos no capítulo seguinte.

[10] Considere aqui, a treebank como constituída apenas dos exemplos que compõem o conjunto de treinamento

Tabela 5.7: Casos de teste dos experimentos.

No.	Sentença
#1	nenhum PRON-DET aluno N conhece V-FIN o ART livro N
#2	a ART terra N é V-FIN um ART planeta N
#3	este PRON-DET livro N é V-FIN de PRP muito PRON-DET valor N
#4	o ART filho N obedece V-FIN a PRP o ART pai N
#5	algum PRON-DET professor N emprestou V-FIN o ART livro N a PRP aquele PRON-DET aluno N
#6	carlos PROP não ADV foi V-FIN a PRP o ART colégio N ontem ADV
#7	pedro PROP colocou V-FIN o ART livro N em PRP a ART biblioteca N hoje ADV
#8	o ART pai N fala V-FIN e CONJ-C os ART filhos N escutam V-FIN
#9	eu PRON-PERS sei V-FIN que CONJ-S a ART terra N é V-FIN redonda ADJ
#10	todos_os PRON-DET filhos N precisam V-FIN de PRP que CONJ-S os ART pais N os PRON-PERS ajudem V-FIN
#11	paulo PROP conhece V-FIN o ART homem N que PRON-INDP falou V-FIN
#12	a ART casa N que PRON-INDP eu PRON-PERS vi V-FIN tem V-FIN três NUM quartos N grandes ADJ
#13	amo V-FIN a ART terra N onde PRON-INDP nasci V-FIN
#14	é V-FIN frances ADJ o ART homem N de PRP quem PRON-INDP eu PRON-PERS te PRON-PERS falei V-FIN
#15	o ART poeta N cuja PRON-DET obra N o ART professor N citou V-FIN é V-FIN camões PROP
#16	tenho V-FIN como ADV provar V minha PRON-DET inocência N
#17	aquele PRON-DET homem N é V-FIN admirado V-PCP por PRP quantos PRON-DET o PRON-PERS conhecem V-FIN
#18	maria PROP faltou V-FIN a PRP a ART aula N porque CONJ-S esteve V-FIN doente ADJ
#19	o ART pai N trabalha V-FIN para PRP que CONJ-S os ART filhos N possam V-FIN estudar V
#20	embora CONJ-S luís PROP estivesse V-FIN doente ADJ ele PRON-PERS foi V-FIN a PRP a ART aula N
#21	meus PRON-DET colegas N virão V-FIN se CONJ-S tiverem V-FIN tempo N
#22	esse PRON-DET rapaz N estudou V-FIN tanto ADV que CONJ-S adoeceu V-FIN
#23	maria PROP é V-FIN mais ADV inteligente ADJ do_que CONJ-S teresa PROP

5.2.2 Resumo Quantitativo

Procedeu-se a uma análise quantitativa baseada nas medidas clássicas de precisão e cobertura apenas para a primeira resposta de PAPO-I e II para cada caso de teste. Como usual em avaliação de parsers, ambas as medidas são calculadas com base nas regras observadas

Tabela 5.8: Tempo de resposta de PAPO-II para cada caso de teste (Testes realizados num AMD Atlon XP 1900 1.6 Gz. 512MB Ram).

No.	Senteça
#1	1s
#2	1s
#3	4s
#4	2s
#5	8s
#6	58s
#7	20s
#8	9s
#9	4s
#10	13s
#11	7s
#12	4min27s
#13	4s
#14	53s
#15	4min37s
#16	11s
#17	1min6s
#18	3min36s
#19	11s
#20	45s
#21	2s
#22	4s
#23	4s
Média:	~47s

na árvore gerada pelo parser e na árvore correta segundo o modelo da treebank. Denotando (i) como regras(T) o conjunto de todas regras observadas numa árvore, (ii) como #A a cardinalidade (número de elementos) de um conjunto qualquer A, (iii) como Parser(S) a árvore mais provável gerada para uma sentença S pelo parser Parser (ora PAPO-I, ora PAPO-II) e (iv) como ideal(S) a árvore correta esperada, temos as seguintes definições:

$$precisão(S, Parser) = \frac{\#(regras(Parser(S)) \cap regras(ideal(S)))}{\#regras(Parser(S))} \tag{5.1}$$

$$cobertura(S, Parser) = \frac{\#(regras(Parser(S)) \cap regras(ideal(S)))}{\#regras(ideal(S))} \tag{5.2}$$

Na Tabela 5.9 é apresentado, para cada sentença S entre os casos de teste, precisão(S, PAPO-I), precisão(S, PAPO-II), cobertura(S, PAPO-I) e cobertura(S, PAPO-II), bem como as médias. Pode-se observar nessa tabela que, exceto pelo caso #6, PAPO-II sempre se comporta melhor. Ainda assim, esses dados não fazem juz à superioridade de PAPO-II nesses casos de teste se consideradas mais respostas que apenas a primeira gerada. Algumas vezes, estruturas complexas, corretas ou minimamente erradas, nem ensaiadas por PAPO-I, são construídas por PAPO-II antes da décima resposta gerada. Detalharemos esses casos na seção seguinte.

Tabela 5.9: Precisão e cobertura de PAPO-I e II para cada caso de teste.

Caso	precisão PAPO-I	precisão PAPO-II	cobertura PAPO-I	cobertura PAPO-II
#1	1	1	1	1
#2	1	1	1	1
#3	.5	.55	.5	.6
#4	1	1	1	1
#5	.67	.77	.6	.77
#6	.8	.63	.8	.7
#7	.64	.75	.5	.64
#8	.8	.9	.56	.64
#9	.9	.9	.9	.9
#10	.87	.93	.87	.93
#11	.75	.75	.6	.6
#12	.5	.71	.5	.7
#13	.43	.43	.3	.3
#14	.57	.64	.67	.75
#15	0	.67	0	.53
#16	.63	.75	.56	.67
#17	.38	.38	.35	.36
#18	0	.73	0	.61
#19	.88	1	.88	1
#20	0	.93	0	.93
#21	.88	.9	.7	.9
#22	.88	.88	.7	.7
#23	1	1	1	1
Média:	.75	.79	0.6	.75

5.2.3 Análise Qualitativa

Apresenta-se aqui a totalidade dos casos de teste agrupados por provável causa prioritária da falha ou sucesso e devidamente comentados quanto a possíveis soluções. Sempre será apresentada a árvore mais provável produzida por PAPO-II, uma vez que, em quase todos os casos de teste, PAPO-I no máximo se lhe equipara. No entanto, sempre que pertinente, será descrito o comportamento deste, bem como mencionados os casos em que um melhor resultado foi obtido (por PAPO-I ou II) numa árvore menos provável entre as dez mais prováveis obtidas.

5.2.3.1 Falha por Treebank Insuficiente

O mau comportamento de ambos os modelos para alguns casos de teste devem-se, pelo menos parcialmente, à carência de exemplos na treebank que tornem mais provável ou até possível a análise esperada. Nesse caso, a solução seria incluir novos exemplos bem como balanceá-los. Em todos os casos apresentados a seguir, é gerada uma análise possível, mas não a que se desejaria fosse a mais provável.

A análise das sentenças #5, #7 e #10 falha pela simples ausência, na treebank, dos verbos em questão ("emprestou", bitransitivo e "precisam", transitivo indireto) ou sua presença ("colocou", transitivo com complemento adverbial), muito pouco freqüente, entretanto, e numa aplicação que não corresponde à desejada no caso. Portanto, o resultado é que sejam tratados como verbos desconhecidos, caso em que as probabilidades favorecem uma interpretação como verbo transitivo direto.

Vale observar que, apesar disso, a análise correta da sentença #10 é gerada, apenas por PAPO-II, em quarto lugar. O resultado de PAPO-I para as sentenças #7 e #10 manifesta problemas de ruído, enquanto para a sentença #5 PAPO-I manifesta a mesma insuficiência da treebank.

```
SUBJ:np
   >N: pron*det('algum' <quant> M S)   Algum
   H: n('professor' M S) professor
P: v*fin('emprestar' PS 3S IND) emprestou
ACC: np
   >N: art('o' <artd> M S)         o
   H: n('livro' M S)       livro
PIV:pp
   H: prp('a' <sam->)    a
      P<: np
          >N: pron*det('aquele' <-sam> <dem> M S)       aquele
          H: n('aluno' M S)         aluno
```

Figura 5.1: Análise do *Palavras* para a sentença #5 *Algum professor emprestou o livro àquele aluno* (Oração de verbo transitivo direto e indireto)

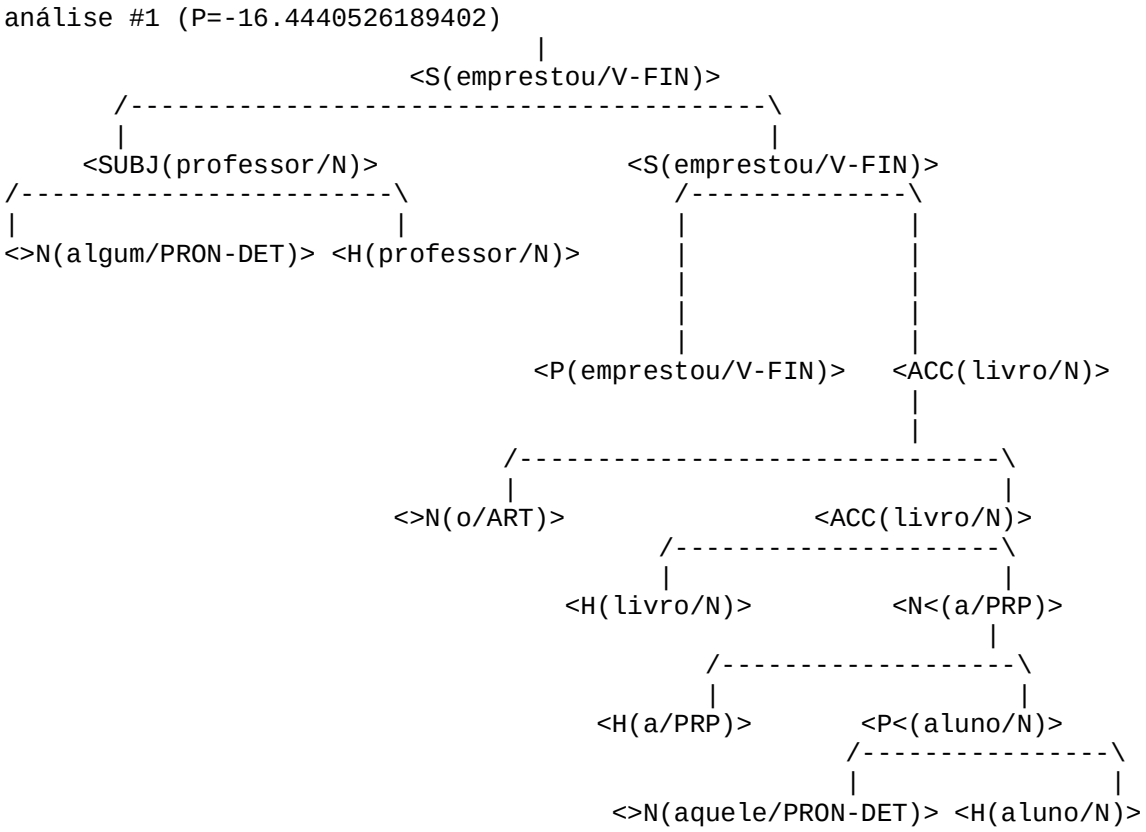

Figura 5.2: Análise do PAPO-II para a sentença #5 *Algum professor emprestou o livro àquele aluno* (Oração de verbo transitivo direto e indireto)

```
SUBJ: prop('Pedro' M S)          Pedro
P: v*fin('colocar' PS 3S IND)    colocou
ACC: np
   >N: art('o' <artd> M S)       o
   H: n('livro' M S)       livro
ADVO: pp
   H: prp('em' <sam->)  em
   P<:np
      >N:art('o' <-sam> <artd> F S)      a
      H:n('biblioteca' F S)        biblioteca
ADVL:adv('hoje')         hoje
```

Figura 5.3: Análise do *Palavras* para a sentença #7 *Pedro colocou o livro na biblioteca hoje* (Oração transitiva indireta com locativo - complemento: advérbio de lugar)

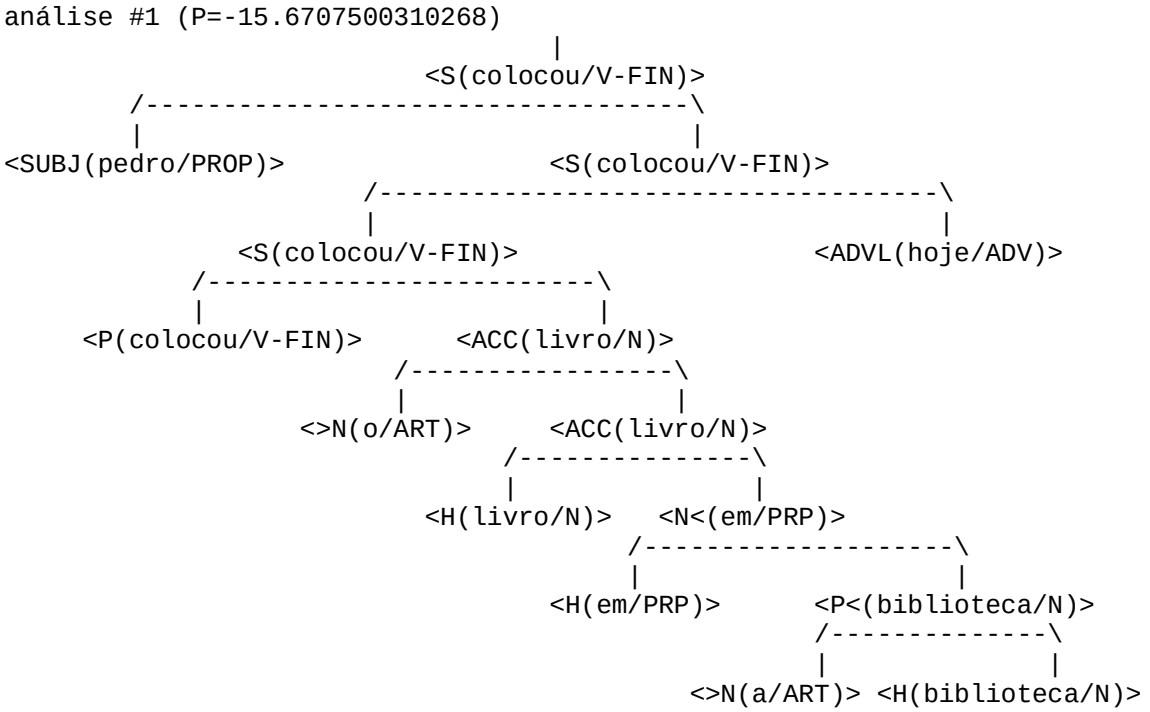

Figura 5.4: Análise do PAPO-II para a sentença #7 *Pedro colocou o livro na biblioteca hoje* (Oração transitiva indireta com locativo - complemento: advérbio de lugar)

```
SUBJ: np
  >N: pron*det('todo_o' <quant> M P)   Todos_os
  H: n('filho' M P)          filhos
P: v*fin('precisar' PR 3P IND)    precisam
P<: fcl
   PIV: pp
     H: prp('de')de
   SUB: conj*s('que')     que
   SUBJ: np
     >N: art('o' <artd> M P)       os
     H: n('pai' M P)       pais
   ACC: pron*pers('eles' M 3P ACC)      os
   P: v*fin('ajudar' PR 3P SUBJ)       ajudem
```

Figura 5.5: Análise do *Palavras* para a sentença #10 *Todos os filhos precisam de que os pais os ajudem* (Período composto por subordinação: substantiva objetiva indireta)

5.2.3.2 Falha por Ruído

Não se pode garantir que ruído na entrada do gerador de modelos seja a causa dos péssimo resultados a seguir, mas fato é que evidenciam a presença de regras absolutamente anômalas, provavelmente geradas no pré-processamento do corpus.

As sentenças #3, #6 e #20 manifestam a anomalia de um verbo auxiliar (AUX) ser modificado (diretamente) por um modificador pós-nominal (N<), por sua vez naturalmente gerado por sintagma preposicionado. PAPO-I acerta a sentença #3, sofre por insuficiência de treebank na sentença #6 e não responde para a sentença #20.

Na sentença #9, ocorre a anomalia de um modificador pós-nominal (N<) modificar diretamente um verbo (de ligação), tomando o lugar natural de um predicativo do sujeito (SC). Vale notar que PAPO-II, apesar disso, acerta a estrutura, notadamente complexa, e, só por isso, não acerta de todo. PAPO-I, no caso, manifesta a mesma anomalia, obtendo uma estrutura absurda, no entanto.

As análises para as sentenças #18 e #22 são anômalas por conter um conectivo subordinador (SUB) modificando um núcleo de sentença verbal (P), quando aquela classe somente poderia ser núcleo de sentenças subordinadas. PAPO-I não responde para a sentença #18, mas apresenta a mesma anomalia para sentença #22 e #21 (esta última analisada corretamente por PAPO-II).

PAPO-I, exclusivamente, manifesta uma anomalia inusitada nas sentenças #7, #10 (treebank insuficiente para PAPO-II em ambos os casos) e #19 (correta para PAPO-II):

Figura 5.6: Análise do PAPO-II para a sentença #10 *Todos os filhos precisam de que os pais os ajudem* (Período composto por subordinação: substantiva objetiva indireta)

```
SUBJ: np
   >N: pron*det('este' <dem> M S)       Este
   H: n('livro' M S)       livro
P: v*fin('ser' PR 3S IND)          é
SC:pp
   H:prp('de')    de
   P<:np
     >N:pron*det('muito' <quant> M S) muito
     H:n('valor' M S)     valor
```

Figura 5.7: Análise do *Palavras* para a sentença #3 *Este livro é de muito valor* (Oração de predicado nominal)

```
análise #1 (P=-7.92715372462626)
                                        |
                                   <S(é/V-FIN)>
                 /--------------------------------------\
                 |                                      |
          <P<(livro/N)>                           <P(é/V-FIN)>
          /---------------\                /------------------\
          |               |                |                  |
<>N(este/PRON-DET)> <H(livro/N)> <AUX(é/V-FIN)>      <N<(de/PRP)>
                                              /-----------------\
                                              |                 |
                                         <H(de/PRP)>     <P<(valor/N)>
                                                        /----------------\
                                                        |                |
                                          <>N(muito/PRON-DET)> <H(valor/N)>
```

Figura 5.8: Análise do PAPO-II para a sentença #3 *Este livro é de muito valor* (Oração de predicado nominal)

uma preposição (respectivamente "em", "de" e "para") assume, sem complemento, o papel de um sintagma completo (respectivamente N<, ADVL e ADVL).

5.2.3.3 Falha por Não-Aprendibilidade

Identificou-se um tipo de estrutura sintática que PAPO jamais conseguirá aprender. Trata-se das estruturas de subordinação em que uma palavra, a um só tempo, acumula a função de conectivo e outra função sintática qualquer dentro da oração subordinada. Isso acontece em todas as orações subordinadas adjetivas (sentenças #11, #12, #14 e #15) e substantivas introduzidas por advérbio (sentenças #13 e #16) e pronome subordinadores (sentença #17).

O problema se resume no fato de que, por terem função sintática nunca nuclear dentro da oração subordinada, esses elementos de conexão desaparecem dentro do sin-

```
SUBJ: prop('Carlos' M S)        Carlos
ADVL: adv('não')          não
P: v*fin('ir' PS 3S IND)  foi
ADVS: pp
   H: prp('a' <sam->)     a
   P<: np
      >N: art('o' <-sam> <artd> M S)    o
      H: n('colégio' M S)          colégio
ADVL: adv('ontem')        ontem
```

Figura 5.9: Análise do *Palavras* para a sentença #6 *Carlos não foi ao colégio ontem* (Oração transitiva indireta com locativo - complemento: advérbio de lugar)

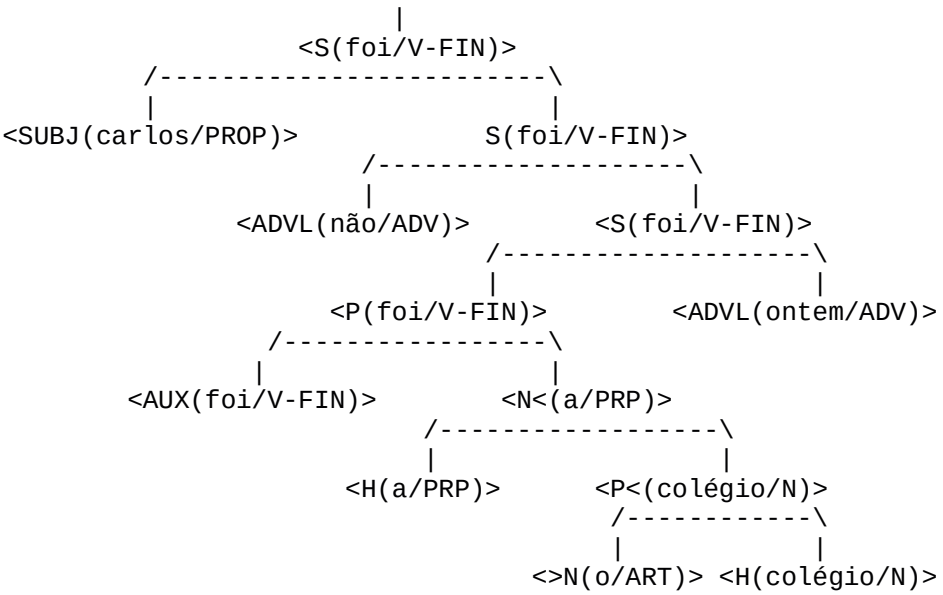

Figura 5.10: Análise do PAPO-II para a sentença #6 *Carlos não foi ao colégio ontem* (Oração transitiva indireta com locativo - complemento: advérbio de lugar)

```
CJT: fcl
  SUBJ: np
     >N: art('o' <artd> M S)      O
      H: n('pai' M S)     pai
  P: v*fin('falar' PR 3S IND)     fala
CO: conj*c('e' <co-vfin> <co-fmc>)       e
CJT: fcl
  SUBJ:np
     >N: art('o' <artd> M P)      os
     H: n('filho' M P)     filhos
  P: v*fin('escutar' PR 3P IND)  escutam
```

Figura 5.11: Análise do *Palavras* para a sentença #8 *O pai fala e os filhos escutam* (Período composto por coordenação)

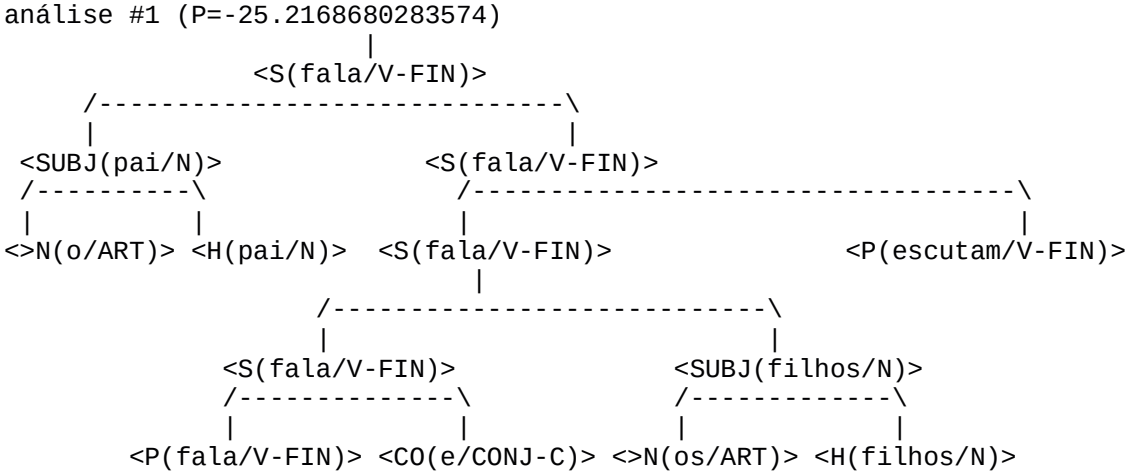

Figura 5.12: Análise do PAPO-II para a sentença #8 *O pai fala e os filhos escutam* (Período composto por coordenação)

```
SUBJ: pron*pers('eu' M/F 1S NOM)     Eu
P: v*fin('saber' PR 1S IND)       sei
ACC: fcl
     SUB: conj*s('que')   que
     SUBJ: np
        >N: art('o' <artd> F S)    a
        H: n('terra' <prop> F S)  Terra
     P: v*fin('ser' PR 3S IND)     é
     SC: adj('redondo' F S)        redonda
```

Figura 5.13: Análise do *Palavras* para a sentença #9 *Eu sei que a Terra é redonda* (Período composto por subordinação: substantiva)

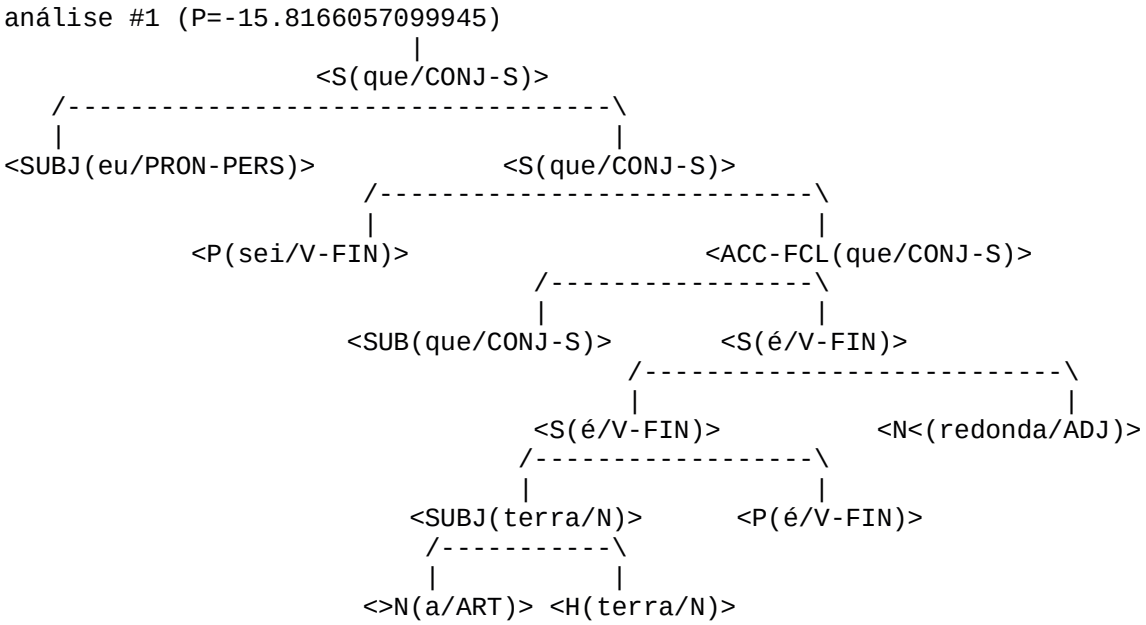

Figura 5.14: Análise do PAPO-II para a sentença #9 *Eu sei que a Terra é redonda* (Período composto por subordinação: substantiva)

SUBJ: prop('Maria' F S) Maria
P: v*fin('faltar' PS 3S IND) faltou
PIV: pp
H: prp('a' <sam->) a
P<: np
>N: art('o' <-sam> <artd> F S) a
H: n('aula' F S) aula
ADVL: fcl
SUB: conj*s('porque') porque
P: v*fin('estar' <ink> PS 3S IND) esteve
SC: adj('doente' M/F S) doente

Figura 5.15: Análise do *Palavras* para a sentença #18 *Maria faltou à aula porque esteve doente* (Subordinação adverbial: oração causal)

Figura 5.16: Análise do PAPO-II para a sentença #18 *Maria faltou à aula porque esteve doente* (Subordinação adverbial: oração causal)

```
ADVL: fcl
  SUB: conj*s('embora')   embora
  SUBJ: prop('Luís' M S)  Luís
  P : v*fin('estar' IMPF 3S SUBJ)   estivesse
  SC: adj('doente' M S)    doente
SUBJ: pron*pers('ele' M 3S NOM) ele
P:v*fin('ir' PS 3S IND)      foi
ADVS:pp
  H:prp('a' <sam->)        a
  P< :np
    >N:art('o' <-sam> <artd> F S)  a
    H:n('aula' F S)          aula
```

Figura 5.17: Análise do *Palavras* para a sentença #20 *Embora Luís estivesse doente, ele foi à aula* (Oração Concessiva).

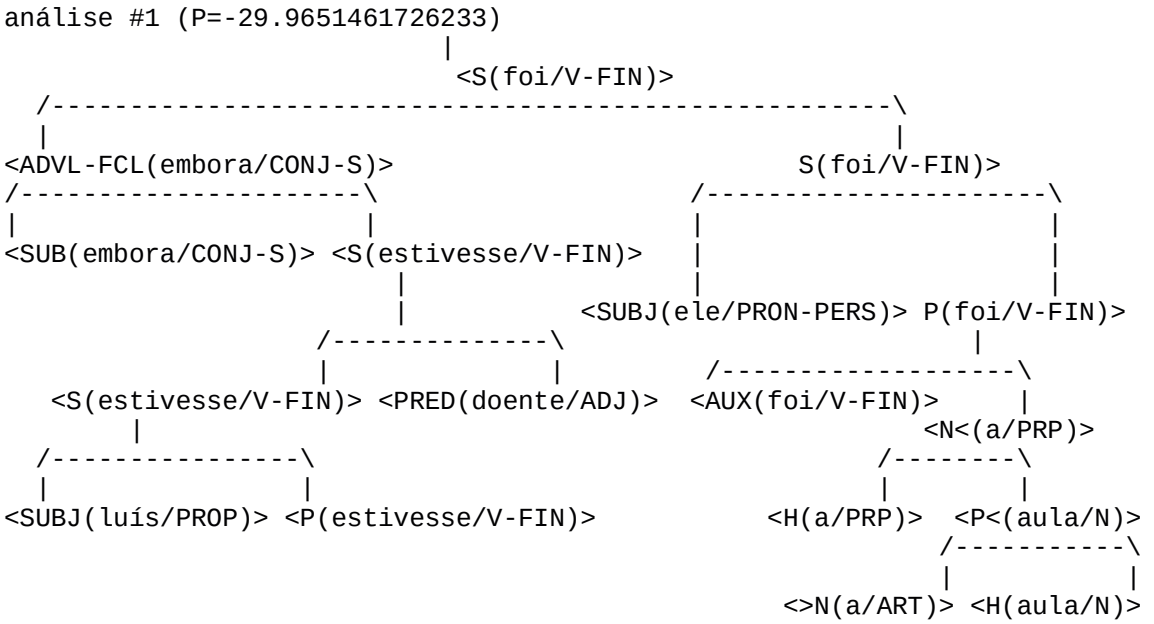

Figura 5.18: Análise do PAPO-II para a sentença #20 *Embora Luís estivesse doente, ele foi à aula* (Oração Concessiva).

SUBJ: np
  >N: pron*det('esse' <dem> M S) Esse
  H: n('rapaz' M S) rapaz
P: v*fin('estudar' PS 3S IND) estudou
ACC: pron*det('tanto' <quant> M S) tanto
  A<: fcl
    SUB: conj*s('que') que
    P: v*fin('adoecer' <ink> PS 3S IND) adoeceu

Figura 5.19: Análise do *Palavras* para a sentença #22 *Este rapaz estudou tanto que adoeceu* (Oração Consecutiva).

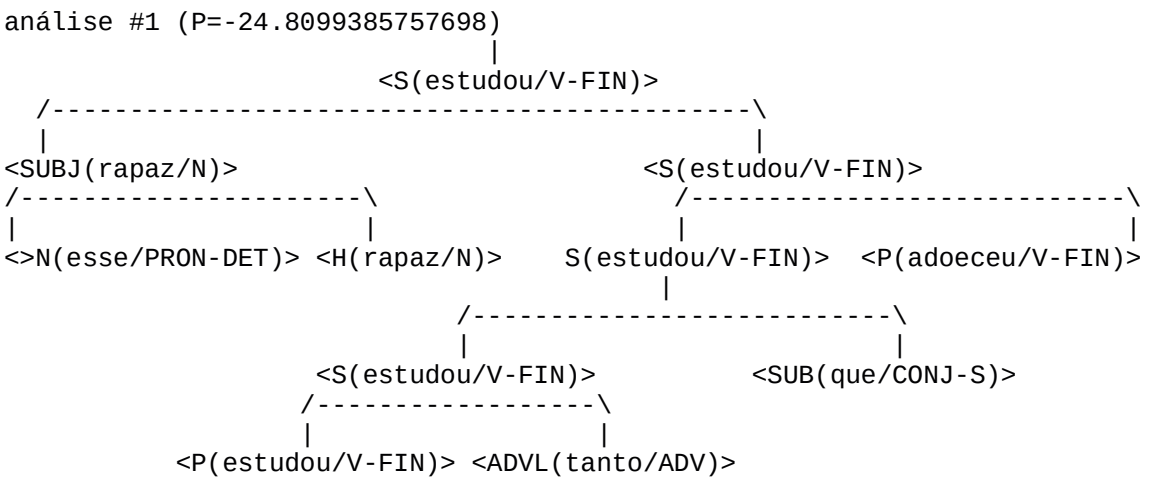

Figura 5.20: Análise do PAPO-II para a sentença #22 *Este rapaz estudou tanto que adoeceu* (Oração Consecutiva).

tagma S de que fazem parte, que só expressa exteriormente seu núcleo, verbal. Assim, não haverá pista externa que permita o encaixe deste S como modificador em uma outra cláusula. Algumas estruturas recebem correntemente um tratamento diverso e não de todo absurdo, como observado na Figura 5.24. Nesse caso, o pronome relativo é considerado núcleo da oração adjetiva, modificado por um sintagma S. Infelizmente, como aí bem exemplificado, essa também não é uma solução aceitável, já que é quase impossível receber, nessas condições, qualquer pista de qual deva ser o rótulo sintático para o pronome relativo (que, no caso, deveria ser rotulado como objeto direto - ACC).

Uma solução possível seria não permitir o surgimento do símbolo S nesse tipo de estrutura, mas substituí-lo diretamente pelas etiquetas N<-FCL (oração adjetiva) e ACC/SUBJ/P<-FCL (oração substantiva). Essa solução, entretanto, não é desejável por dois motivos: (i) praticamente quadruplica o branching factor de P (núcleo de S) e, principalmente, (ii) evita que as generalizações obtidas para S se apliquem em todas as orações subordinadas, ou seja, dificulta a aquisição da recursividade da língua.

Talvez o melhor tratamento para esse caso seja que, num passo de pré-processamento, tanto da treebank quanto das sentenças a serem analisadas, a dualidade desses pronomes/advérbios conectivos seja desdobrada, gerando uma palavra artificial à esquerda cuja função seja meramente conectiva e um pronome/advérbio, com uma etiqueta especial (PRO/ADV-KS - pronome/advérbio subordinador, PRO/ADV-KS-REL - pronome/advérbio subordinador relativo), à direita. Tal palavra artificial, ao ser *modificada* por um S gerará N</ACC/SUBJ/P<-FCL, dependendo da palavra original que a gerou (informação que ficaria codificada em sua tag morfossintática).

```
SUBJ: prop('Paulo' M S)          Paulo
P: v*fin('conhecer' PR 3S IND)   conhece
ACC: np
  >N: art('o' <artd> M S)        o
  H: n('homem' M S)    homem
  N<: fcl
      SUBJ: pron*indp('que' <rel> M S) que
      P: v*fin('falar' PS 3S IND)  falou
```

Figura 5.21: Análise do *Palavras* para a sentença #11 *Paulo conhece o homem que falou* (Período composto por subordinação: adjetiva restritiva)

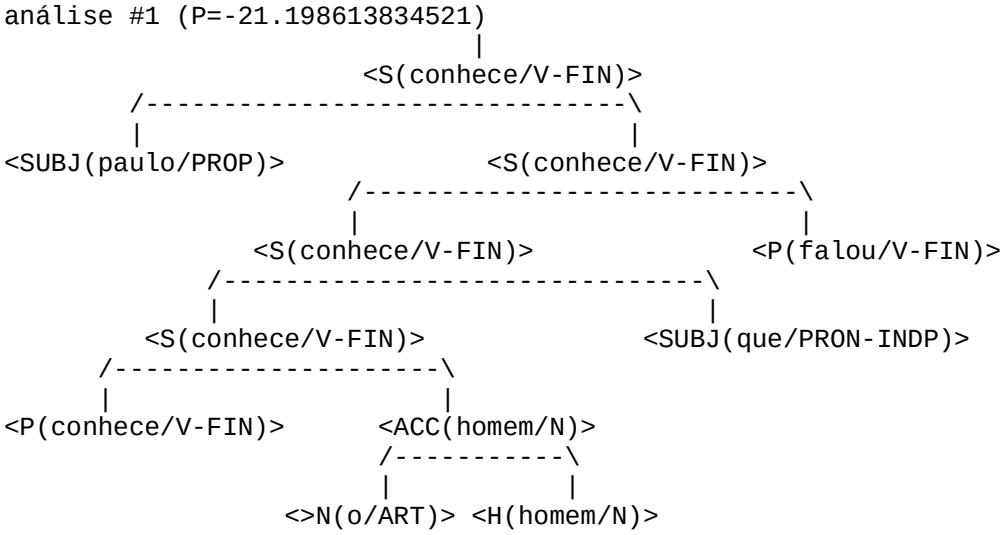

Figura 5.22: Análise do PAPO-II para a sentença #11 *Paulo conhece o homem que falou* (Período composto por subordinação: adjetiva restritiva)

A análise da sentença #13 é ainda mais prejudicada pela etiquetação morfossintática de "onde" como pronome (no caso, relativo). Essa é uma das inconsistências da gramática tradicional que aqui mostra claramente seus frutos: por considerá-lo um pronome, em vez de advérbio, PAPO permite, coerentemente, que "onde" se torne regularmente objeto (ACC) de uma oração não-subordinada. Fica assim defendida a classe dos advérbios relativos/subordinadores.

A maioria das sentenças aqui agrupadas apresenta ainda outros problemas. Há, por exemplo, anomalias na sentença #17 (modificador pré-nominal modificando pronome pessoal) e sentença #12 ("quartos grandes" formando, por si só, um modificador pós-nominal e, para piorar, modificando o núcleo do sintagma verbal). Entretanto, trata-se de problemas já relatados.

```
SUBJ:np
  >N:art('o' <artd> F S) A
  H:n('casa' F S)          casa
  N<: fcl
     ACC: pron*indp('que' <rel> M S)    que
     SUBJ: pron*pers('eu' M/F 1S NOM)          eu
     P: v*fin('ver' PS 1S IND)     vi
P: v*fin('ter' PR 3S IND) tem
ACC: np
  >N: num('três' <card> M P)    três
  H: n('quarto' M P)      quartos
  N<: adj('grande' M P) grandes
```

Figura 5.23: Análise do *Palavras* para a sentença #12 *A casa que eu vi tem três quartos grandes* (Período composto por subordinação: adjetiva restritiva)

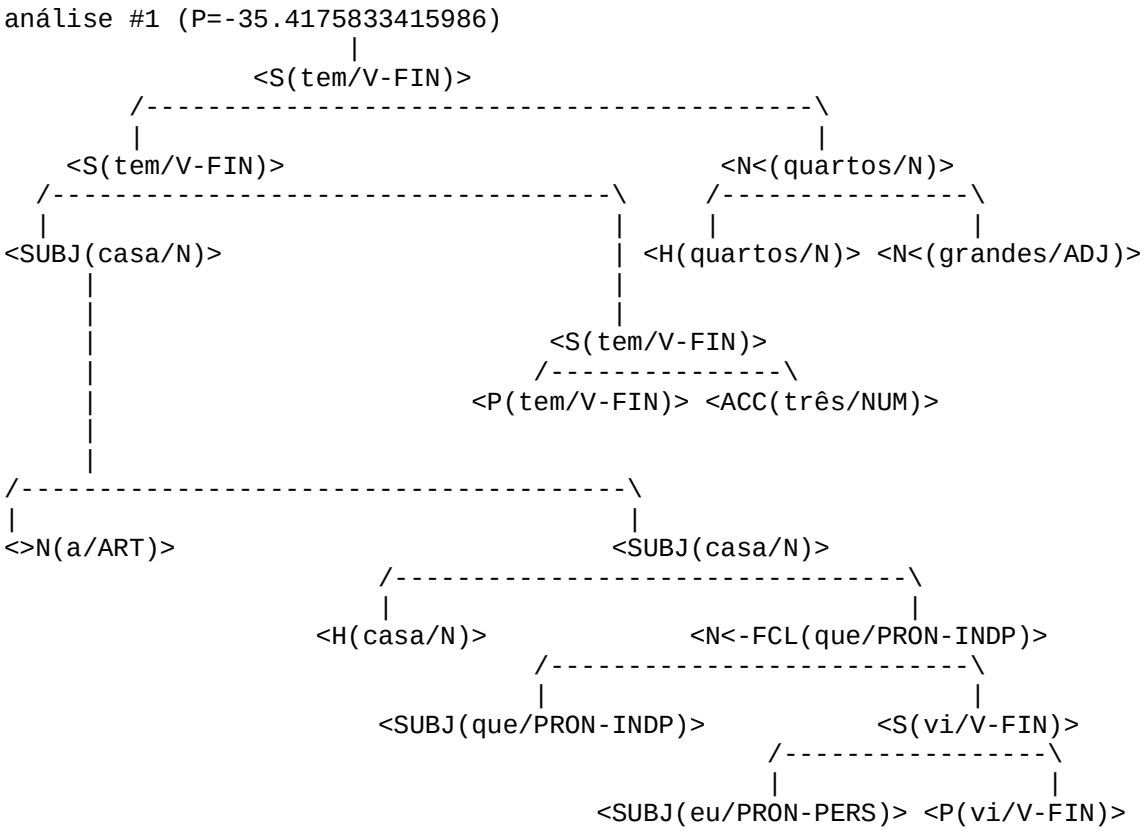

Figura 5.24: Análise do PAPO-II para a sentença #12 *A casa que eu vi tem três quartos grandes* (Período composto por subordinação: adjetiva restritiva)

```
P: v*fin('amar' <ink> PR 1S IND)          Amo
ADVL: pp
  H: prp('a')       a
  P<: np
    H: n('terra' F S)        terra
    N<: fcl
      ADVL: adv('onde' <rel> <ks>)          onde
      P: v*fin('nascer' <ink> PS 1S IND)    nasci
```

Figura 5.25: Análise do *Palavras* para a sentença #13 *Amo a terra onde nasci* (Período composto por subordinação: adjetiva restritiva)

```
análise #1 (P=-8.63999157414346)
                                    |
                                <S(amo/V-FIN)>
                              /-------------------------------------\
                              |                                     |
                          <S(amo/V-FIN)>                    <P(nasci/V-FIN)>
                       /------------------------------\
                       |                              |
              <S(amo/V-FIN)>                 <ACC(onde/PRON-INDP)>
        /-------------------\
        |                   |
<P(amo/V-FIN)>       <P<(terra/N)>
                     /-----------\
                     |           |
                <>N(a/ART)> <H(terra/N)>
```

Figura 5.26: Análise do PAPO-II para a sentença #13 *Amo a terra onde nasci* (Período composto por subordinação: adjetiva restritiva)

```
P: v*fin('ser' PR 3S IND)          É
SC: adj('francês' M S)    francês
SUBJ: np
  >N: art('o' <artd> M S)          o
  H: n('homem' M S)     homem
  N<: pp
    H: prp('de')   de
    P<: pron-indp('quem' <rel> M/F S/P)          quem
  SUBJ: pron*pers('eu' M/F 1S NOM)   eu
  ACC: pron*pers('tu' M/F 2S ACC)    te
  P: v*fin('falar' PS 1S IND)        falei
```

Figura 5.27: Análise do *Palavras* para a sentença #14 *É francês o homem de quem eu te falei* (Período composto por subordinação: adjetiva restritiva)

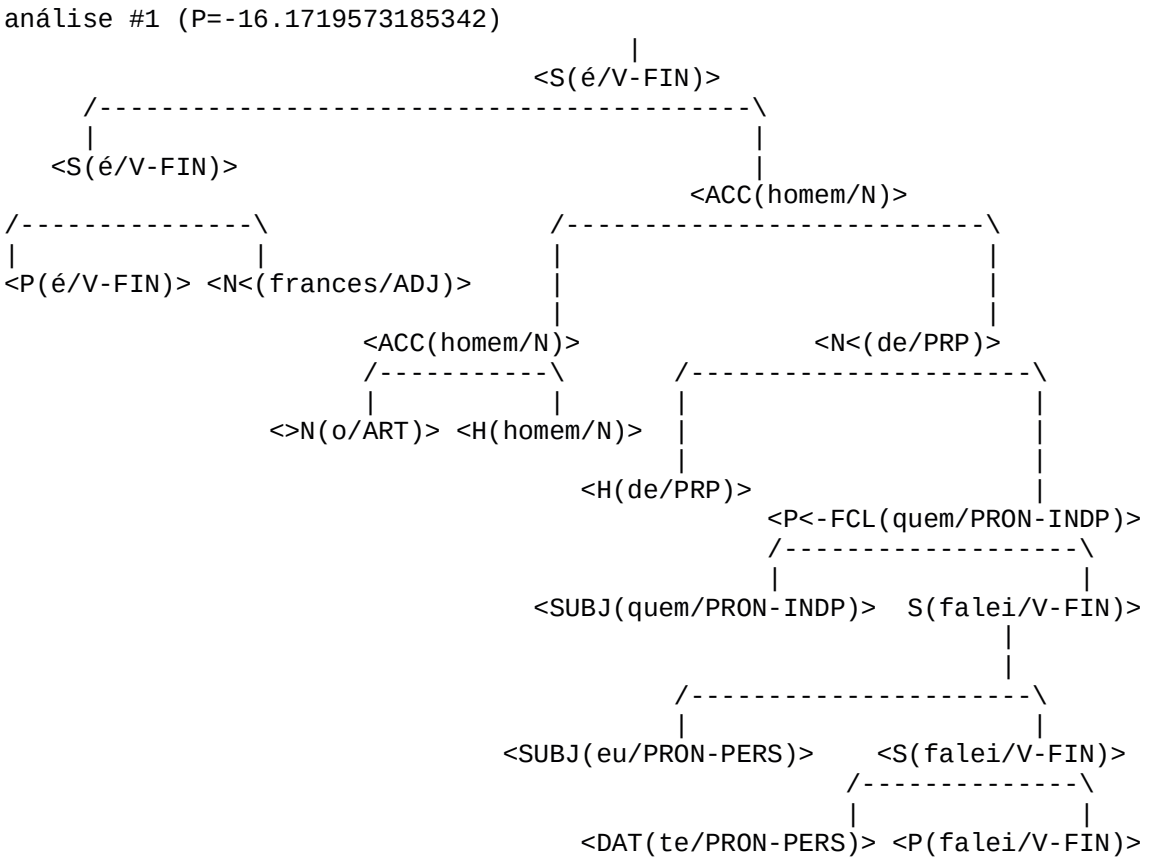

Figura 5.28: Análise do PAPO-II para a sentença #14 *É francês o homem de quem eu te falei* (Período composto por subordinação: adjetiva restritiva)

```
SUBJ:np
  >N: art('o' <artd> M S)          O
  H: n('poeta' M S)        poeta
     N<: fcl
     SUBJ: np
        >N: pron*det('cujo' <rel> F S)    cuja
        H: n('obra' F S) obra
        SUBJ: np
           >N: art('o' <artd> M S)        o
           H: n('professor' M S)          professor
           P: v*fin('citar' PS 3S IND)    citou
P: v*fin('ser' PR 3S IND)          é
SC: prop('Camões' M S)Camões
```

Figura 5.29: Análise do *Palavras* para a sentença #15 *O poeta cuja obra o professor citou é Camões* (Período composto por subordinação: adjetiva restritiva)

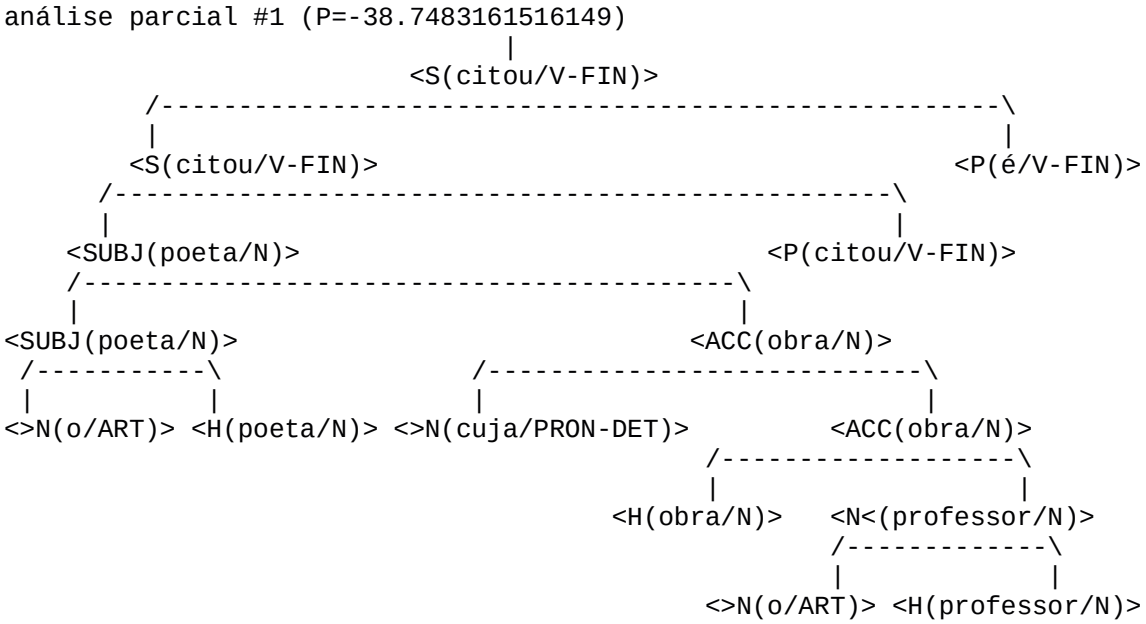

Figura 5.30: Análise do PAPO-II para a sentença #15 *O poeta cuja obra o professor citou é Camões* (Período composto por subordinação: adjetiva restritiva)

P: v*fin('ter' <ink> PR 1S IND) Tenho
ACC: icl
ADVL: adv('como' <rel>) como
P: v*inf('provar') provar
ACC:np
>N: pron*det('meu' <poss 1S> F S) minha
H: n('inocência' F S) inocência

Figura 5.31: Análise do *Palavras* para a sentença #16 *Tenho como provar minha inocência* (Período com oração substantiva derivada de oração adjetiva sem antecedente)

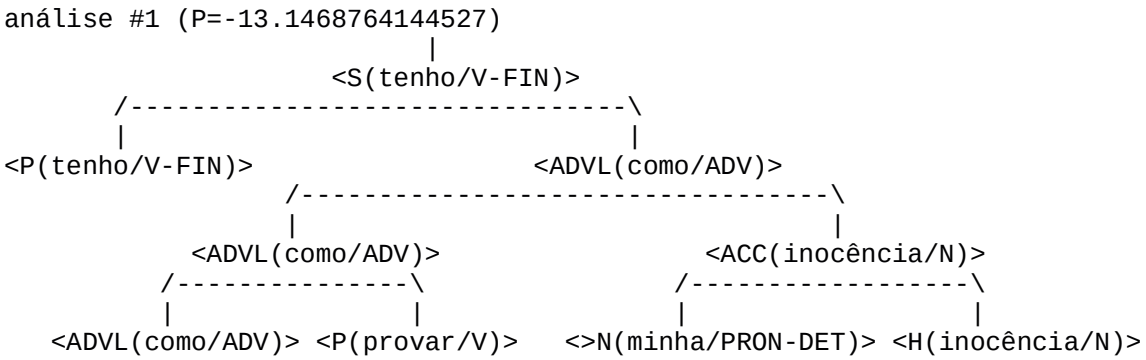

Figura 5.32: Análise do PAPO-II para a sentença #16 *Tenho como provar minha inocência*(Período com oração substantiva derivada de oração adjetiva sem antecedente)

```
SUBJ: np
   >N: pron*det('aquele' <dem> M S)     Aquele
   H: n('homem' M S)    homem
P: vp
  AUX: v*fin('ser' PR 3S IND)    é
  MV: v*pcp('admirar' M S)       admirado
P<: fcl
  PASS:pp
    H: prp('por')          por
    SUBJ: pron*det('quanto' <rel> M P)quantos
    ACC: pron*pers('ele' M 3S ACC)     o
    P: v*fin('conhecer' PR 3P IND)         conhecem
```

Figura 5.33: Análise do *Palavras* para a sentença #17 *Aquele homem é admirado por todos quantos o conhecem* (Período com oração substantiva derivada de oração adjetiva sem antecedente)

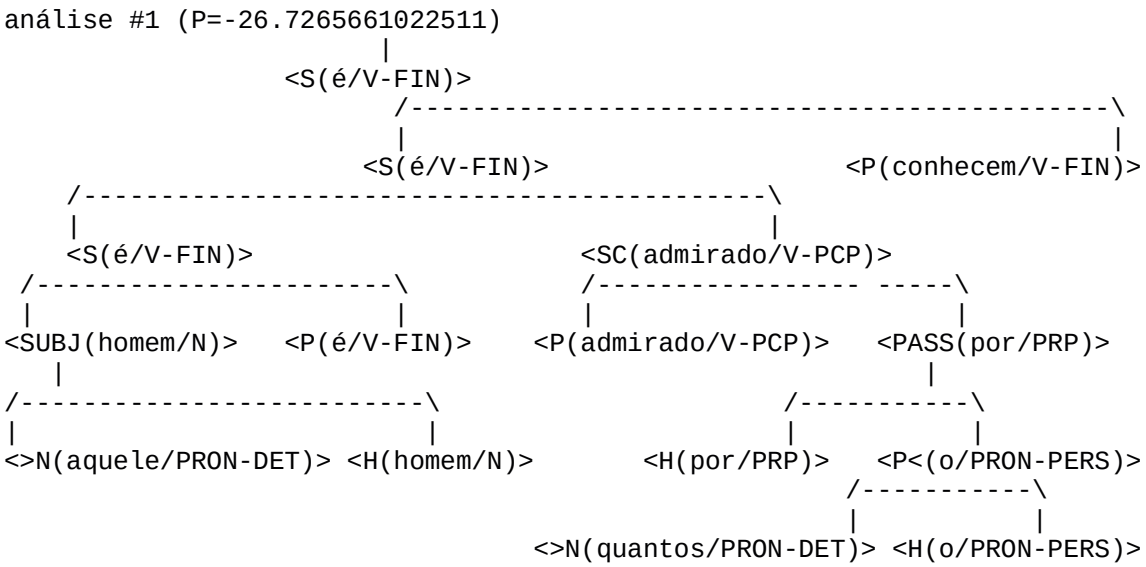

Figura 5.34: Análise do PAPO-II para a sentença #17 *Aquele homem é admirado por todos quantos o conhecem* (Período com oração substantiva derivada de oração adjetiva sem antecedente)

```
SUBJ: np
  >N: pron*det('nenhum' <quant> M S)nenhum
  H: n('aluno' M S)       aluno
P: v*fin('conhecer' PR 3S IND)   conhece
ACC: np
  >N: art('o' <artd> M S)         o
  H: n('livro' M S)       livro
```

Figura 5.35: Análise do *Palavras* para a sentença #1 *Nenhum aluno conhece o livro* (Oração transitiva)

```
análise #1 (P=-13.4725783697126)

                                |
                        <S(conhece/V-FIN)>
                /--------------------------------------\
                |                                      |
          <SUBJ(aluno/N)>                        <S(conhece/V-FIN)>
           /----------------\                 /---------------------\
           |                |                 |                     |
<>N(nenhum/PRON-DET)> <H(aluno/N)> <P(conhece/V-FIN)>    <ACC(livro/N)>
                                                         /-----------\
                                                         |           |
                                                  <>N(o/ART)> H(livro/N)>
```

Figura 5.36: Análise do PAPO-II para a sentença #1 *Nenhum aluno conhece o livro* (Oração transitiva)

5.2.3.4 Sucesso

Finalmente, sobraram os casos em que PAPO-II e, eventualmente, PAPO-I chegam a análises corretas logo na primeira análise proposta. Trata-se das sentenças #1, #2, #4, e #23 (PAPO-I e II) e sentenças #19 e #21 (anômalas para PAPO-I). Vale observar que as três últimas sentenças figuram entre as mais complexas dos casos de teste.

```
SUBJ: np
  >N: art('o' <artd> F S)a
  H: n('terra' <prop> F S)        Terra
P: v*fin('ser' PR 3S IND)         é
SC: np
  >N: art('um' <arti> M S)        um
  H: n('planeta' M S)     planeta
```

Figura 5.37: Análise do *Palavras* para a sentença #2 *A Terra é um planeta*

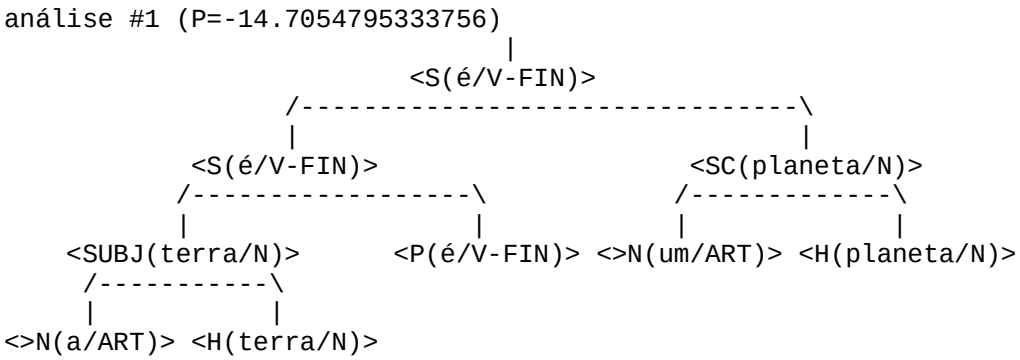

Figura 5.38: Análise do PAPO-II para a sentença #2 *A Terra é um planeta*

SUBJ: np
  >N: art('o' <artd> M S) O
  H: n('filho' M S) filho
P: v*fin('obedecer' PR 3S IND) obedece
  PIV: pp
    H: prp('a' <sam->) a
    P<: np
      >N: art('o' <-sam> <artd> M S) o
      H: n('pai' M S) pai

Figura 5.39: Análise do *Palavras* para a sentença #4 *O filho obedece ao pai* (Oração de verbo transitivo indireto)

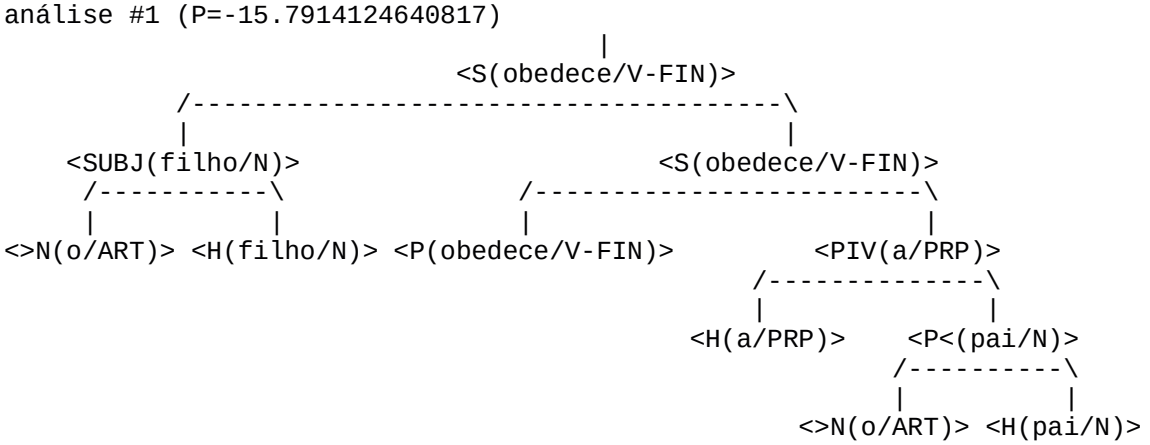

Figura 5.40: Análise do PAPO-II para a sentença #4 *O filho obedece ao pai* (Oração de verbo transitivo indireto)

```
SUBJ:np
   >N:art('o' <artd> M S)          O
   H:n('pai' M S)          pai
P: v*fin('trabalhar' PR 3S IND)  trabalha
P<: fcl
  ADVL:pp
     H:prp('para')         para
     SUB:conj*s('que')     que
     SUBJ:np
       >N:art('o' <artd> M P)     os
       H:n('filho' M P)    filhos
     P: vp
       AUX: v*fin('poder' PR 3P SUBJ)  possam
       MV: v*inf('estudar')        estudar
```

Figura 5.41: Análise do *Palavras* para a sentença #19 *O pai trabalha para que os filhos possam estudar* (Subordinação adverbial: oração final)

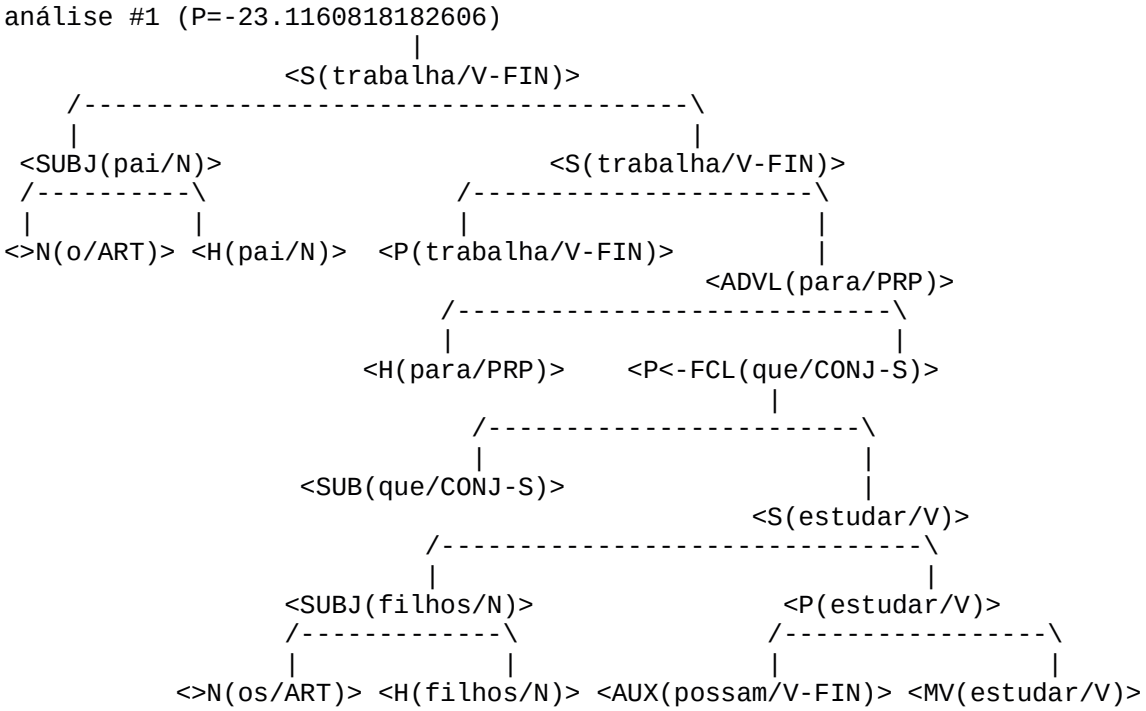

Figura 5.42: Análise do PAPO-II para a sentença #19 *O pai trabalha para que os filhos possam estudar* (Subordinação adverbial: oração final)

```
SUBJ: np
   >N: pron*det('meu' <poss 1S> M P)   Meus
   H: n('colega' M P)      colegas
P: v*fin('vir' FUT 3P IND)          virão
ADVL: fcl
  SUB: conj*s('se')        se
  P: v*fin('ter' <ink> FUT 3P SUBJ)       tiverem
  ACC: n('tempo' M S)    tempo
```

Figura 5.43: Análise do *Palavras* para a sentença #21 *Meus colegas virão se tiverem tempo* (Oração Condicional).

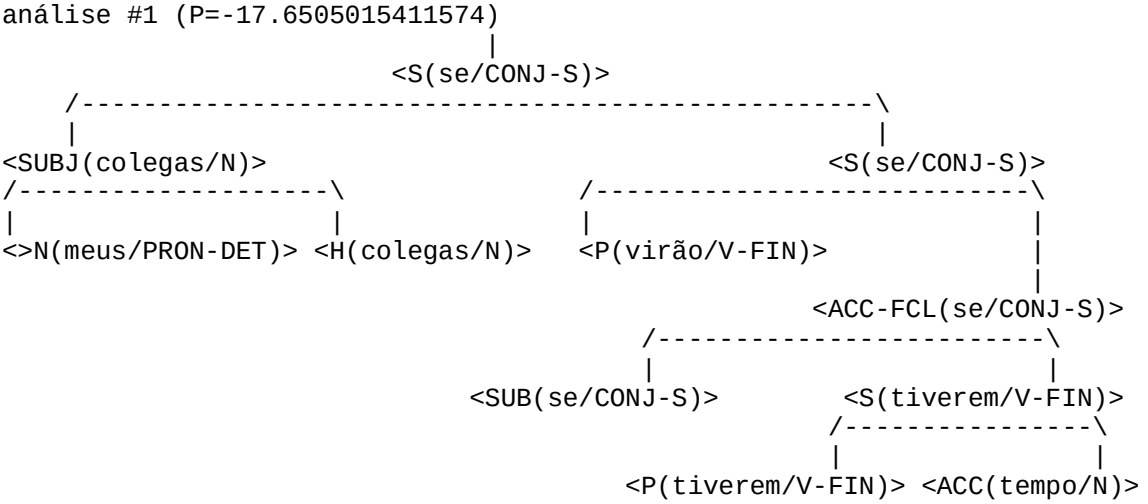

Figura 5.44: Análise do PAPO-II para a sentença #21 *Meus colegas virão se tiverem tempo* (Oração Condicional).

```
SUBJ: prop('Maria' F S)          Maria
P: v*fin('ser' PR 3S IND)        é
SC: ap
  >A: adv('muito' <quant>)       mais
  H: adj('inteligente' F S)      inteligente
  KOMP<: acl
    COM: conj*s('do_que')        do_que
    SUBJ: prop('Teresa' F S)     Teresa
```

Figura 5.45: Análise do *Palavras* para a sentença #23 *Maria é mais inteligente do que Teresa*(Oração Comparativa).

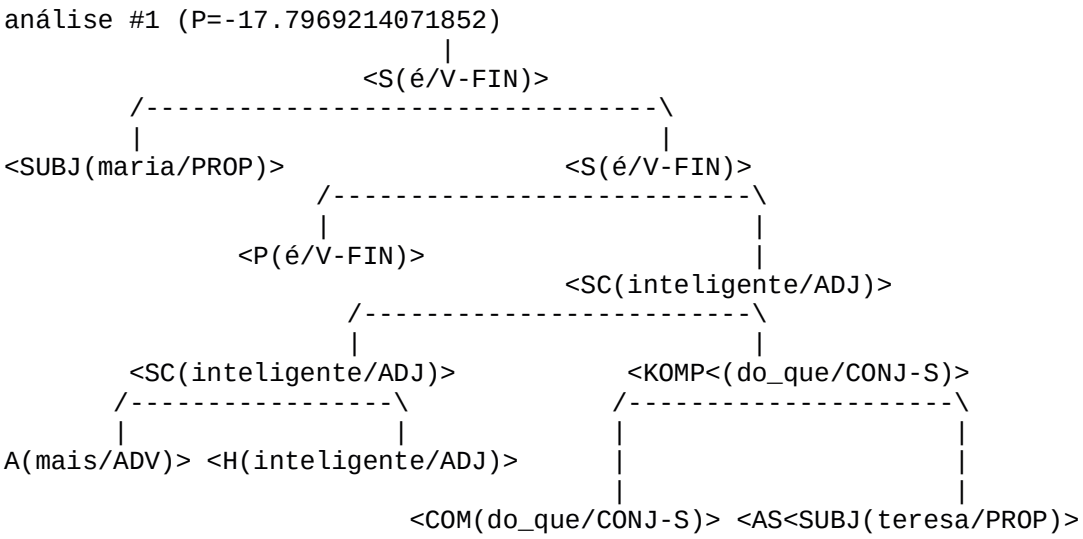

Figura 5.46: Análise do PAPO-II para a sentença #23 *Maria é mais inteligente do que Teresa*(Oração Comparativa).

Capítulo 6

Conclusões e Trabalhos Futuros

Esta tese consistiu em investigar, entender, implementar e analisar o comportamento e a viabilidade de uso de um método estatístico para analisar sintaticamente sentenças em Português do Brasil. Para tanto, foi implementado o sistema PAPO, um ambiente composto por vários módulos que incluem a preparação dos dados usados como treinamento do método de aprendizado, a geração de dois modelos probabilísticos (PAPO-I e PAPO-II) e a análise das sentenças com base nos modelos produzidos. As sentenças usadas como treinamento foram obtidas da treebank CETENFolha, subconjunto do corpus NILC, etiquetado automaticamente com o parser "Palavras", que produz as análises em uma notação não hierárquica ("flat"), baseada em gramáticas de restrições.

Apesar dos diversos problemas encontrados na avaliação preliminar do PAPO, e do volume reduzido dos casos de teste, é possível concluir que os modelos probabilísticos em questão parecem promissores. E, exatamente pela variedade e gravidade dos problemas encontrados, considerou-se bem-sucedida a avaliação realizada, mostrando-se adequada para esse momento do ciclo de vida do sistema. Foi possível sentir em profundidade a influência do modelo de anotação, tanto sintática quanto morfossintática, no comportamento de um parser empírico, cuja capacidade de aprendizado é claramente delimitada e dificilmente se dá com inconsistências.

Acima de tudo, vários e interessantes desafios foram identificados ao longo do desenvolvimento desta tese e merecem uma atenção especial, em trabalhos futuros, entre os quais destacam-se:

- eliminação do ruído presente na *treebank* e balanceamento dos exemplos;
- revisão do modelo de análise, para que se torne mais compatível com o modelo de

aprendizado do PAPO;

- desenvolvimento de heurísticas para facultar maior liberdade no comprimento das sentenças submetidas, incluindo o estabelecimento de condições de parada/desistência do mecanismo de busca;

- investigação de estimativas para o índice de confiança das headwords de uma *treebank*;

- realização de novas avaliações: (1) *intrínsecas*[1], quanto à precisão na atribuição de etiquetas sintáticas, consistência estrutural, similaridade de árvore e *Best First ranking*, e (2) *extrínsecas*, avaliando as estruturas de dependência, e também comparativamente a um outro parser (no caso, o Curupira).

Como contribuições desta tese destacam-se:

- O ineditismo: desconhece-se um trabalho de tal natureza, com modelagem (baseada em etiquetas funcionais) e análise sintática probabilística (totalmente desprovido de conhecimento lingüístico) para o Português do Brasil;

- A contribuição como material de pesquisa: o conteúdo desta tese pode ser usado como fonte de pesquisa para: (1) os métodos estatísticos para parsing existentes na literatura, (2) tendências dos trabalhos produzidos na comunidade científica, (3) evolução das métricas usadas para avaliar parsers empíricos, e (4) fundamentos da teoria do chart-parser;

- A geração de novas pesquisas: esta tese deixa em aberto várias questões que merecem investigação, desde a identificação da fonte do problema de ruído nos modelos, até a pesquisa por novas heurísticas e otimizações de tempo e memória para o parser;

- A identificação de alguns problemas a serem enfrentados por qualquer parser empírico na análise de sentenças problemáticas da língua portuguesa;

- A perspectiva real de se gerar um parser robusto e com bons índices de precisão, de código aberto e de livre acesso, que possa ser usado por toda a comunidade de PLN.

[1] Ver Apêndice B para as definições desses tipos de avaliação.

Apêndice A

Etiquetas de maior freqüência no CETENFolha

Este apêndice apresenta as principais etiquetas do sistema de anotação do parser "Palavras" (Bick 2000) encontradas nas sentenças da treebank CETENFolha.

A.1 Etiquetas Morfossintáticas

- N - Nomes ("cadeira")
- PROP - Nomes próprios ('José')
- PRON-PERS - Pronomes pessoais ("ele","se")
- PRON-DET - Pronomes determinantes ("sua","alguns")
- ART - Artigos ("a","uma")
- ADJ - Adjetivos ("histórica","normativa")
- ADV - Advérbios ("inevitavelmente")
- V - Verbos ("defender")
- NUM - Numerais ("1968","primeira")
- KS - Conjunções subordinativas ("que","como")

- KC - Conjunções coordenativas (“porém”, “ou_seja”, “mais”)
- PRON-INDP - Pronomes especificadores independentes (“o_que”, “o_qual”,“que”)
- IN - Interjeições (“ah”)

A.2 Etiquetas Sintáticas

- S - Sentença
  Exemplo: “**Esta tese trata do problema da análise sintática**”.
- SUBJ - Sujeito
  Exemplo: “**Estudo** revela ação do álcool sobre a memória”.
- P - Predicado(verbo)
  Exemplo: “A companhia deverá ***divulgar*** novos planos até amanhã”.
- MV - Verbo principal
  Exemplo: “Esta inquietação pode ***levar*** ao caos”.
- H - Núcleo de qualque sintagma
  Exemplo: “Esta ***inquietação*** pode levar ao caos”.
- ACC - Objeto Direto Acusativo
  Exemplo: “Sem a fragrância, teriam **um *odor* desagradável de gordura”.**
- DAT - Objeto indireto dativo(somente pronominal)
  Exemplo: “***lhe*** dou um presente”
- PIV - Objeto indireto preposicional
  Exemplo: “O presidente Fernando Henrique chega **à(*a*+a metade de seu mandato** em plena lua de mel com a população
- CJT
  Exemplo: “E depois, quando os vir **encarreirados** e **grandes homens**...”.
- ADV - Objeto adverbial
  Exemplo: “***onde*** mora?”
- ADVL - Adjunto adverbial
  Exemplo: “**Na(*Em* + a) maior parte do material**, o erro persiste”.

- SC - Predicativo do sujeito
  Exemplo: "Atenção: não é ***cheiro*** **de carro limpo**".

- OC - Predicativo do objeto
  Exemplo: "A filosofia é de mater o cofre ***fechadíssimo***".

- PRED - Predicativo do sujeito "livre"
  Exemplo: "Nadava ***nua***".
  "Mas a inflação baixa funciona como inestimável alavanca eleitoral, o que já se comprovara antes, na Argentina e no Peru, **com a *reeleição* de Menem e Fujimore**, respectivamente.

- NPHR - Frases Nominais
  Exemplo: "**Um *professor* um tanto inoclasta**".

- VOC - Vocativo
  Exemplo: "Note-**se** que ela é abastada".

- S< - Aposição de sentença
  Exemplo: "Não venceu o que muito o contrariou".

- CO - Conjunção coordenativa
  Exemplo: "***Mas*** a comparação com Menem e Fujimore deve funcionar como um sinal amarelo."

- SUB - Conjunção subordinativa
  Exemplo: "Minimizava, ***Embora*** **tivesse que admitir que não**".

- >N - Modificador pré-nominal
  Exemplo: "***O*** azul tão claro do céu".

- N< - Modificador pós-nominal
  Exemplo: "O azul tão ***claro*** do céu".

- >A - Pré adjunto adverbial
  Exemplo: "O azul ***tão*** claro do céu".

- A< - Pós adjunto adverbial
  Exemplo: "A árvore estava cheia ***de*** flor".

- APP - Aposição (sempre depois de sintagma nominal + vírgula
  Exemplo: "..que já ultrapassou Plutão, **o último *planeta* do sistema solar**".

- N<PRED - Predicativo em pequenas cláusulas introduzidas por "com/sem"
  Exemplo: "Sem o pai ***ajudando***, não conseguiu".

- P< - Argumento de preposição
  Exemplo: "Uma píngua de ***água***".

- >P - Modificador de preposição
  Exemplo: "..***até*** no Brasil".

- AUX - auxiliar
  Exemplo: "Uma repetidora ***deve*** ser instalada nos próximos dias".

- AS<ADVL
  Exemplo: "... experiência que os homens aparentemente precisam viver, ainda que só na *imaginação*".

- PONT - Pontuação (",","**:**",";", etc.)

- AS<KOMP - Comparativo em pequenas cláusulas
  Exemplo: "Um homem **como um forte, qual um touro**".

- SUBJ-FCL - Subcláusula finita
  Exemplo: "***Quem*** **cedo madruga**...".

- ACC-FCL - Subcláusula finita
  Exemplo: "A noiva não acreditava ***que*** **ele a amasse**".

- SC-FCL - Subcláusula finita
  Exemplo: "Seja ***quem*** **for**".

- P<-FCL - Subcláusula finita
  Exemplo: "Mostrava a pedra a ***quem*** **quisesse ver**".

- A<-FCL – Subcláusula finita
  Exemplo: "Só foi avisado depois ***que*** **o seu jatinho levantou vôo**".

- N<-FCL - - Subcláusula finita
  Exemplo: "A amiga **com *a qual* apareceu na festa**".

- ADVL-FCL - Subcláusula finita
  Exemplo: "Entraram na vila ***quando*** **amanheceu**".

- N<PRED-FCL - Subcláusula finita
  Exemplo: "Na nossa época, os behavioristas aderiram ao empirismo, enquanto os cognitivistas ***tendem*** **a adotar alguma forma de racionalismo**".

- S<-FCL - Subcláusula finita
  Exemplo: "O pai não veio para o aniversário dele, ***o que*** **não o surpreendeu**."

- ACC-ACL - Subcláusula averbal
  Exemplo: "Não consigo saber ***qual*** **a diferença de qualidade**".

- SUBJ-ICL - Subcláusula não finita
  Exemplo: "***Retomar*** **o controle** foi difícil".

- ACC-ICL - Subcláusula não finita
  Exemplo: "Manda **o filho** ***comprar*** **leite**".

- SC-ICL - Subcláusula não finita
  Exemplo: "O problema era ***acabar*** **com os bandidos**".

- N<PRED-ICL - Subcláusula não finita
  Exemplo: "Tem **a mão** ***machucada***".

- P<-ICL - Subcláusula não finita
  Exemplo: "Para o amigo **lhe** ***ajudar***, propôs outra solução".

- N<-ICL - Subcláusula não finita
  Exemplo: "Tem muito **que** ***estudar***".

- ADVL-ICL - Subcláusula não finita
  Exemplo: "Foi à televisão ***recitar*** **o documento**".

Apêndice B

Metodologia de Avaliação

B.1 Introdução

Este Apêndice tem por objetivo descrever o estado da arte a avaliação de parsers para a língua inglesa, com o intuito de se determinar uma metodologia para a avaliação de parsers para a língua portuguesa.

Visando-se medir a performance dos sistemas de parsing, diversos critérios de avaliação têm sido usados: as primeiras avaliações eram empíricas, cujo critério consistia somente na tomada do tempo que levava o sistema para carregar a gramática e analisar as sentenças de teste. Essas medidas eram também usadas para comparação entre sistemas. No entanto, como o tempo é bastante relativo, novas medidas foram sendo incorporadas, como número de passos usados para realizar as análises, taxa de erro, etc.

Muito se tem discutido sobre quais métricas usar na avaliação dos parsers. Algumas métricas têm sido propostas ao longo dos anos, seja como avaliação individual ou comparativa. Black e et al. (1991) propuseram o que talvez seja a primeira tentativa em se avaliar parsers de domínio irrestrito. Trata-se de um sistema de avaliação denominado PARSEVAL, que se tornou largamente usado, apesar de várias limitações (apresentado na seção B.2). Um dos grandes congressos na área, o COLING2000, teve como objetivo discutir as possíveis abordagens para se comparar a eficiência dos parsers. Moore (2000) propõe uma uniformização, com gramática e dados iguais, juntamente com implementações padrão, uma espécie de parser de referência como base para todos os demais a serem analisados. Assim, a metodologia proposta para medir a eficiência seria: (1) providencia-se gramática(s), sentenças de teste e um parser de referência; (2) executa-se o

parser implementado com a gramática nas sentenças de teste; (3) executa-se o parser referência com a mesma gramática e nas mesmas sentenças na mesma máquina; (4) reporta-se tempo de execução do parser proposto como uma percentagem do tempo observado para o parser referência.

B.2 Avaliação de analisadores sintáticos

Os desenvolvedores de parsers reconhecem a necessidade de testes rigorosos para um desenvolvimento efetivo. Entre 1993 e 1996, o subgrupo de avaliação do EAGLES (1995), sugeriu quatro propostas de avaliação (Scarlett 2000):

1. Avaliação de diagnóstico - ajuda a localizar as deficiências do sistema, através de um conjunto de teste com fenômenos lingüísticos enumerados.
2. Avaliação de progresso - compara estágios sucessivos de desenvolvimento de um sistema no decorrer do tempo de sua implementação.
3. Avaliação de adequação - determina se um sistema alcança algum requisito especificado.
4. Avaliação de performance - mede quantitativamente o desempenho do sistema em uma ou mais áreas específicas. São distinguidos:
   (a) Critérios - consiste no que é interessante para avaliação - precisão, velocidade, taxa de erro, etc.
   (b) Medida - especifica o desempenho do sistema de acordo com algum critério escolhido - taxa de sucesso no total, tempo de processamento, percentagem incorreta.
   (c) Método - consiste em avaliar como determinar um valor apropriado de uma medida para um dado sistema.

Rashmi e Sarkar (2000) publicaram em seu artigo duas maneiras de se avaliar um parser. A primeira é com relação a sua cobertura gramatical, ou seja, quantos fenômenos lingüísticos sua gramática consegue cobrir. A segunda é com relação à cobertura do próprio parser, ou seja, quantas sentenças de um corpus de teste ele consegue classificar corretamente. Surgem, a partir daí, dois tipos de avaliação: um chamado *test-suite based*,

que avalia a cobertura gramatical; e um chamado *corpus-based*, que avalia a corretude do parser.

Carroll, Briscoe, e Sanlippo (1998) organizam os métodos de avaliação em três categorias: sem corpus, com corpus não anotado e com corpus anotado. Srinivas, Sarkar, Doran, e Hockey (1998) agrupam as propostas para avaliação em duas diferentes classes: intrínsecas e extrínsecas, distribuídas conforme a Tabela B.1.

Tabela B.1: Métodos de Avaliação Extrínsecos e Intrínsecos.

Intrínsecos	**Extrínsecos**
Baseado num conjunto de teste(sem corpus)	Comparativo
Quanto à cobertura(corpus não anotado)	PARSEVAL
Média das análises(corpus não anotado)	Estrutura de Dependência
Entropia/Perplexidade(corpus não anotado)	Árvore de Dependência
Precisão na atribuição de etiquetas(corpus anotado)	Relação
Consistência Estrutural(corpus anotado)	
Consistência Best-First/Ranked(corpus anotado)	
Similaridade da árvore(corpus anotado)	
Quantidade de arcos usados no chart-parser	

B.2.1 Categorias de Avaliação

B.2.1.1 Métodos Intrínsecos

A avaliação intrínseca mede a performance do sistema no contexto interno ao qual foi desenvolvido. Esse tipo de avaliação pode ser usado tanto para sistemas baseados em gramática quanto para os estatísticos, baseados em corpus, anotados ou não. Para sistemas baseados em gramática, a avaliação ajuda a identificar os pontos fracos da gramática. Uma vez que a avaliação é desempenhada dentro do próprio sistema, as métricas usadas para avaliação intrínseca podem ser sensíveis a representações de saída e a características do sistema.

É possível ainda fazer uma avaliação comparada com resultados de outros sistemas. Entretanto, diferentemente da avaliação comparativa, raramente há o caso em que os resultados serão diretamente comparáveis uns com os outros.

Alguns tipos de avaliação podem ser destacados com essa característica intrínseca, conforme pode ser observado abaixo: avaliação baseada num conjunto de teste, avaliação

quanto à cobertura, avaliação baseada na média das análises, entropia/perplexidade, precisão na atribuição de etiquetas morfossintáticas, consistência estrutural, avaliação da análise quanto à sua pontuação num *ranking* e similaridade da árvore.

Avaliação baseada num conjunto de teste (não usa corpus)

Esse tipo de avaliação consiste basicamente na listagem das construções que são cobertas e das que não são cobertas por uma dada gramática ou um parser (para exemplo veja (Grover, Carrol, e Briscoe 1993)). Um conjunto de testes com uma lista de sentenças é mantido numa base de dados. A avaliação é usada para melhoria e verificação de consistência entre gerações de desenvolvimento de gramáticas (avaliação de progresso). Pode ser usada também como apontador para falhas na gramática (avaliação de diagnóstico).

A vantagem de tal método é que não requer recursos como um corpus, bastando que o projetista da gramática supra o sistema com a informação necessária. Uma desvantagem que pode ser bastante perceptível é quanto à cobertura de casos raros ou casos de falta de informação do próprio projetista, uma vez que pode não haver um corpus como base, tendo o sistema que contar somente com a experiência do projetista.

Avaliação quanto à cobertura (corpus não anotado)

Gramáticas convencionais podem ser vistas como coleções de regras que precisamente especificam estruturas de constituintes possíveis. Ao definir um conjunto de estruturas legais, a gramática também define a classe de sentenças corretas.

Um passo útil para tal tipo de avaliação é determinar a percentagem de exemplos de um dado corpus para o qual o parser/gramática é capaz de atribuir pelo menos uma análise completa (não fragmentada) (Briscoe e Carrol 1995b).

Média das Análises (corpus não anotado)

É possível que uma sentença possa ter mais que uma análise correta gerada por uma gramática. Em 1993, um grupo da IBM (Black, Garside, e (Eds.) 1993) sugeriu o cálculo chamado de base de análise, PB (do inglês, *Parse Base*). Eles definiram uma média geométrica do número de análises (árvores sintáticas geradas) divididas pelo número de tokens de entrada (palavras) para cada sentença analisada. Assim, pode-se prever o número médio de árvores sintáticas para cada sentença através do número de palavras que ela possui. É assumido que o número de análises cresce linearmente com o tamanho da sentença, ao invés de exponencialmente, como observado na prática por Carroll et al., em (Carroll, Briscoe, e Sanlippo 1998). O que acaba acontecendo é que PB superestima

o número de análises para sentenças curtas e subestima o número para sentenças longas.

Para resolver esse problema, Briscoe e Carrol (1995b) propuseram uma nova medida chamada média das análises, APB (do inglês *Average Parse Base*). Essa medida é definida como a média geométrica sobre as sentenças no corpus de $\sqrt[n]{p}$, em que n é o número de palavras na sentença, e p, o número de árvores sintáticas para aquela sentença. A medida diz que uma sentença de tamanho n obtida de um corpus e analisada com um corpus/gramática teria um número esperado de APB^n análises (árvores). APB pode ser computado para um corpus grande, e ainda estimar o grau de ambigüidade na gramática.

Entropia/Perplexidade (corpus não anotado)

Alguns métodos estatísticos tentam resolver o problema da super geração descrito anteriormente, permitindo que o sistema de parsing atribua pontuações para as análises. Uma forma comum de atribuir pesos é usar uma gramática probabilística. A gramática probabilística (veja exemplo em (Charniak 1997)) atribui um conjunto de estruturas legais a cada seqüência de palavras da linguagem e dá uma probabilidade a cada estrutura. A probabilidade da seqüência de palavras da linguagem é a soma de probabilidades individuais de todas as suas estruturas.

Um modelo de linguagem probabilístico pode ser usado para calcular a entropia de um corpus (ou equivalentemente, a perplexidade do modelo). Por exemplo, uma gramática probabilística, geralmente induzida de um corpus anotado, pode ser usada para calcular a entropia de um corpus não anotado. Isto produz uma medida de quão bem a indução da gramática capturou regularidades no corpus, minimizando a ambigüidade e a não previsibilidade. A principal vantagem da medida da entropia é que ela permite uma comparação útil de diferentes gramáticas probabilísticas no mesmo corpus. As desvantagens são que essas medidas são aplicáveis somente a gramáticas probabilísticas, e, ainda que com melhores resultados, provêm uma medida fraca. Isto é, uma pontuação baixa de ambigüidade no corpus não significa que a gramática probabilística derivada dele será necessariamente precisa.

Embora seja em sua forma básica uma medida específica para um dado modelo ou corpus, a abordagem pode ser usada para comparar a eficiência de diferentes modelos de linguagem ou a eficiência de diferentes esquemas de treinamento (estimando probabilidades no mesmo conjunto de teste). Além disso, a abordagem pode ser generalizada para prover uma medida, independente de modelo, da complexidade inerente de um corpus através de sucessivas aproximações de modelos de linguagem baseados em ngramas.

Precisão na atribuição etiquetas (corpus anotado)

É um tipo de medida usada tanto para avaliar taggers (e parsers eventualmente), quando se trata de etiquetas morfossintáticas, como parsers, quando se trata de etiquetas sintáticas. As medidas mais satisfatórias para essa tarefa são precisão e cobertura.

Consistência Estrutural - (Zero Crossing Brackets) - corpus anotado

Numa tentativa de medir diretamente a eficiência do parsing, o grupo da IBM (Black, Garside, e (Eds.) 1993) definiu uma métrica chamada consistência estrutural. É a percentagem de sentenças para um corpus de teste que recebe pelo menos uma análise completa que não tem parentização conflitante quando comparada com a correta. É uma medida bastante simples e mais forte que medida de cobertura, mas requer um corpus anotado ainda que fazendo um uso mínimo dessas anotações. Sozinha, é uma medida inadequada e tenderá a favorecer sistemas com estruturas mínimas, uma vez que terão poucos parênteses conflitantes.

Best-first/Ranked Consistency (corpus anotado)

Brill (1993) mede a precisão de sistemas probabilísticos computando a percentagem das análises com ranking mais altos, produzidas pelo parser, que são idênticas a análises manuais proporcionadas no corpus anotado (treebank). A medida pode ser estendida de tal forma que as n principais análises sejam comparadas e a pontuação seja relativa a seu ranking, dando uma medida significativa de quão freqüente o parser deveria gerar uma análise apropriada e também algumas medidas indiretas do grau de ambigüidade na gramática.

Entretanto, essa medida é dependente de um corpus confiável e preciso, que seja totalmente compatível com a saída do parser. Aqui, novamente, medidas baseadas em ausência de erros de parentização não podem ser usadas se o sistema a ser analisado produz análises fragmentadas, com parentizações parciais.

Similaridade de árvores (corpus anotado)

Define medidas de similaridade de vários tipos como a taxa de regras na análise obtida sobre as regras do corpus anotado. Essas medidas são mais minuciosas do que uma identificação completa da análise correta, e são mais tolerantes a erros existentes no corpus de teste. Entretanto, para tais medidas mais fracas que a identificação completa é difícil ver como o mapeamento é feito em muitas tarefas de análise: por exemplo, com 80% da árvore correta, quão importante é o restante dos 20% para a interpretação correta da

sentença?

Número médio de arcos num chart-parser

Carabalho e Charniak (1998) medem a quantidade de trabalho feito por um chart parser estatístico em termos de número médio de arcos retirados da agenda antes de encontrar uma árvore sintática. Segundo o autor, o método tem a vantagem de ser independente de plataforma, e produz uma medida de perfeição, em que perfeição é encarada como o número mínimo de arcos que seria necessário retirar da agenda de modo a produzir a análise correta. Para uma gramática binarizada (lados direitos com apenas dois símbolos), que ele usa no trabalho, em que cada arco retirado é um constituinte completo, esse número é simplesmente o número de terminais mais o número de não-terminais na forma sentencial. Para cada sentença são armazenadas a quantidade de arcos retirados da pilha, necessários para alcançar a primeira análise, a as medidas de precisão e cobertura daquela análise.

B.2.1.2 Métodos Extrínsecos

A avaliação extrínseca é possível quando um sistema de análise está incorporado numa aplicação e a contribuição para a performance geral da aplicação pode ser medida. O objetivo no método extrínseco é avaliar a adequação de um parser. Essa avaliação também pode ser usada como um método indireto para comparar sistemas de parsing até mesmo se eles produzem representações diferentes para suas saídas, contanto que a saída possa ser convertida para uma forma na qual possa ser usada na aplicação. Exemplos de métodos extrínsecos são: métodos comparativos, PARSEVAL, métrica baseada em estrutura de dependência, entre outros:

Métodos Comparativos

O objetivo dos métodos comparativos de avaliação é comparar a performance de dois ou mais parsers que usam diferentes formalismos gramaticais ou diferentes modelos estatísticos. A avaliação comparativa ajuda a identificar pontos fortes e/ou pontos fracos relativos do sistema, e pode sugerir possibilidades de combinar diferentes abordagens. Entretanto, tal esquema de avaliação requer uma métrica que é insensível às diferenças representacionais das saídas produzidas pelos diferentes parsers. Para alcançar isso, tal métrica deve ter uma representação abstrata das representações individuais de forma a ser capaz de compará-los diretamente. Uma desvantagem dessa abordagem é que os pontos fortes das

representações de certos parsers podem se perder nesse processo de abstração.

PARSEVAL/Penn Treebank (corpus anotado)

Proposto por Harrison, Abney, Fleckenger, Gdaniec, Grishman, Hindle, Ingria, Marcus, Santorini, e Strzalkowski (1991) e Marcus e et al. (1993), o esquema de avaliação PARSEVAL tem sido largamente usado, apesar de várias limitações e contrapontos, possivelmente porque nada mais tenha sido proposto que supra todas as necessidades. O esquema PARSEVAL propõe uma métrica de avaliação que quantifica e compara a performance de diferentes parsers. O procedimento de avaliação compara a saída do parser com as análises anotadas da treebank; usa informação de parentização da representação da treebank de uma sentença e a análise produzida para computar três medidas (Srinivas, Sarkar, Doran, e Hockey 1998): crossing brackets, precisão e cobertura.

Tais métricas, chamadas estruturais, são baseadas na avaliação das fronteiras dos sintagmas. Os algoritmos de parsing têm por objetivo otimizar uma métrica em comum, que é a probabilidade de se ter uma árvore corretamente rotulada, ou seja, com uma marcação correta das fronteiras dos constituintes. Assim, dado um nó em uma árvore sintática, a seqüência de palavras dominadas por esse nó forma um sintagma, sendo a fronteira do sintagma representada por um intervalo inteiro $[i, j]$, em que i representa o índice da primeira palavra e j da última palavra no sintagma.

Já Black e et al. (1991) propunham três medidas estruturais para avaliar sistemas de parsing: Precisão Rotulada (Labeled Precision), Cobertura Rotulada (Labeled Recall) e número de Crossing-Brackets. Segundo Lin (1995), esse esquema de avaliação pode ser classificado como em nível de sintagma, ou nível de sentença. As três medidas são computadas da seguinte forma:

- Precisão e Cobertura: as fronteiras dos sintagmas na resposta (análise produzida pelo parser) e na chave (análise da treebank) são tratados como dois conjuntos (A e K), em que A é a análise obtida do parser proposto e K, a chave da treebank a ser usada na avaliação. A cobertura é definida como a percentagem na chave que também é encontrada na resposta ($\frac{|A \wedge K|}{|K|}$). A medida precisão é definida como a percentagem de fronteiras no sintagma da resposta que também é encontrada na chave ($\frac{|A \wedge K|}{|A|}$).
- Número de crossing-brackets: Uma fronteira de sintagma $[i, j]$ na resposta e outra na chave $[i', j']$ é um par de crossing brackets se $i < i' \leq j < j'$. Os parsers podem ser avaliados pelo número médio de crossing brackets por sentença.

As medidas propostas no PARSEVAL partem do pressuposto de que um constituinte está correto se corresponde ao mesmo conjunto de palavras (ignorando qualquer caractere de pontuação) e tem o mesmo rótulo que um constituinte na treebank.

Exemplo: Considere (1) chave e (2) a análise do parser:

1. [Eles [[chegaram] ontem]]
2. [[Eles [chegaram]][ontem]]

Temos:

- fronteiras para os sintagmas em (1): [0,2], [1,1] e [1,2].
- fronteira para os sintagmas em (2): [0,2], [1,1], [0,1] e [2,2].

Pontuações em 2: precisão = $2/4 = 50\%$, cobertura = $2/3 = 66.7\%$ e um par de crossing brackets: [0,1] na chave e [1,2] na resposta.

Essas pontuações têm de ser consideradas juntas para ter significado. Por exemplo, tratando a sentença como uma lista de palavras *[eles chegaram ontem]* teríamos 0-crossing-brackets e 100% de precisão. Entretanto, a cobertura seria baixa ($1/3 = 33\%$).

Goodman, em (Goodman 1996b), classifica as medidas descritas acima e algumas outras em níveis de rigidez, que podem ser seguidos ao se determinar se um constituinte está correto. O mais exigente é o chamado Labelled Match (LM). Nesse modelo de avaliação, um constituinte $(s, t, X) \in TG$, em que s é início, t fim, X rótulo e TG é a análise obtida do parser que se quer avaliar, é correto de acordo com a métrica Labelled Match se e somente se $(s, t, X) \in TC$ (análise correta). O próximo nível de rigidez é chamado Bracket Match. É parecido com o LM, exceto que o rótulo do não-terminal é ignorado. Formalmente, um constituinte $(s, t, X) \in TG$ é correto se existe um Y tal que $(s, t, Y) \in TC$. O menos restrito é o Consistent Brackets (também chamado Crossing Brackets). É parecido com Bracket Match no sentido de que o rótulo é ignorado. Além disso a tupla (s,t,X) não necessita estar em TC; ele deve simplesmente não estar sendo considerado por qualquer $(q, r, Y) \in TC$. Uma tripla particular (q,r,Y) é excluída de (s,t,X) se não há uma forma que (s,t,X) e (q,r,Y) possa estar na mesma árvore sintática. Em particular, se o intervalo (s,t) cruza o intervalo (q,r), então (s,t,X) é excluído e contado como um erro. Formalmente, dizemos que (s,t) cruza (q,r) se somente se $s < q \leq t < r$ ou $q < s \leq r < t$.

Sendo assim, considerando

- Ng - número de não terminais na árvore sintática produzida pelo parser (TG)
- Nc - número de não terminais na árvore sintática correta (TC).
- L - número de constituintes em TG que são corretos de acordo com o Labelled Match.
- B - número de constituintes em TG que são corretos de acordo com Bracket Match.
- C - número de constituintes em TG corretos de acordo com Consistent Brackets;

e seguindo essa definição, são seis as métricas para avaliação:

1. Labelled Recall Rate = $\frac{L}{Nc}$
2. Labelled Tree Rate = 1 se L = Nc[1].
3. Bracket Recall Rate = $\frac{B}{Nc}$
4. Bracket Tree Rate = 1 se B = Nc.
5. Consistent Brackets Recall Rate = $\frac{C}{Ng}$[2].
6. Consistent Brackets Tree Rate= 1 se C = Ng[3].

Dependendo da implementação, alguns critérios avaliam se a análise está correta. Para Black e et al. (1992), o critério de corretude é que o número de crossing-brackets deve ser 0. Já em Magerman, a sentença está analisada corretamente se ambos precisão e cobertura são 100%.

Alguns autores, como Charniak (1997) e Collins (1996), consideram algumas medidas de crossing-brackets, como número médio de crossing brackets por sentença (CB), percentagem de sentença com 0 crossing-brackets (OCB), percentagem de sentença com crossing brackets ≤ 2 (2CB).

Há vários pontos contra esse esquema de avaliação: a descrição original publicada e o software de avaliação são específicos para o inglês. Uma limitação significativa da métrica é que não foi projetada para avaliar parsers que produzem análises parciais. Além

[1] É chamado também Critério de Viterbi

[2] É frequentemente chamado de Crossing Brackets Rate.

[3] Essa métrica é fortemente relacionada a Bracketed Tree Rate. Esse critério é também chamado de Zero Crossing Brackets Rate.

disso, pode somente ser aplicada a árvores gramaticais geradas por parsers baseados em constituintes; não é minuciosa o suficiente para avaliar parsers com respeito a fenômenos sintáticos específicos. Carroll, Minnen, e Briscoe (1999) apontam que as estruturas da treebank têm sido criticadas como sendo "flat" (contendo relativamente poucos parênteses) e portanto, a pontuação do crossing-bracket tenderá a ser baixa. A medida de precisão penaliza umas partes por inserir parênteses extras, possivelmente corretos. Lin (1995) observou que a medida de crossing-brackets penaliza o erro de preferência de ligações entre constituintes feitas mais de uma vez. Uma análise mais superficial, contendo menos informação sintática, pontua melhor na métrica PARSEVAL. Esses métodos avaliam as análises somente em termos gerais. Um método mais flexível deveria ser capaz de avaliar as análises seletivamente com respeito a qualquer tipo de fenômeno sintático dado, por exemplo, quanto à manipulação de conjunções ou ligações dos sintagmas preposicionais. Respostas desse tipo teriam condição de avaliar um parser de propósito específico, e facilitariam o diagnóstico de análises incorretas.

Métrica baseada em Estrutura de Dependência

Para superar os problemas das medidas PARSEVAL na gramática do Penn Treebank, Lin (1995) propôs uma avaliação baseada na estrutura de dependência, na qual análises de estrutura de constituintes do parser e da treebank são convertidas num conjunto de relacionamentos de dependência. E esses relacionamentos de dependência é que são usados para a comparação. Lin defende que as vantagens da tal avaliação são: diferenças inconseqüentes (na parentização) são ignoradas; parsers podem ser seletivamente avaliados com respeito a qualquer tipo de fenômeno sintático; e erros na saída do parser podem ser apontados mais precisamente.

Parsers baseados em dependência produzem um conjunto de ligações dependentes entre as palavras e a sentença. A suposição fundamental da teoria da gramática de dependência é que a estrutura sintática consiste de nós lexicais (representando palavras) e relações binárias (dependências) ligando-os.

Em contraste ao esquema baseado em estrutura sintagmal, as representações de dependência não têm categorias de estrutura de constituintes. Uma dependência existe entre duas palavras se uma palavra é sensível ou condicionada a alguma propriedade da outra, como definido por alguma regra gramatical. Relações de dependência são binárias, diretas, e podem ser tipadas de forma a garantir os diversos tipos de dependência.

Carroll, Briscoe, e Sanlippo (1998) propõem um esquema de avaliação no qual são comparadas essas relações de dependência. Esse método também pode ser aplicado a

árvores de constituintes, desde que transformadas em árvores de dependência (com um algoritmo proposto por ele).

Árvore de Dependência

Em uma árvore de dependência, cada palavra na sentença é modificador de exatamente uma palavra (chamada núcleo). Uma lista de tuplas especifica uma árvore de dependência. A tupla:

$$(modificador \quad TAG \quad posicão \quad núcleo \quad [relacionamento])$$

em que modificador é a palavra; TAG é a categoria lexical; núcleo, que é modificada; relacionamento é uma especificação do tipo de dependência entre o núcleo e seu modificador, do tipo sujeito, adjunto, complemento, especificador, etc.; e posição indica a posição relativa do núcleo com relação ao modificador (valores como: $<$, $>$, $<<$, $>>$, etc.).

Avaliação para a Estrutura de Dependência

A proposta de avaliação segue o esquema: a cada dependência atribuída incorretamente, é computado um erro. O erro passa a ser o número de palavras que atribuem dependências diferentes daquelas existente na análise correta.

Dadas duas árvores de dependência, a chave e a análise proposta (resposta), a função *avalia* retorna o erro da resposta. Esse erro é a distância de Hamming entre a resposta e a chave, porque ela é o número de relacionamentos de dependência que devem ser alterados de modo a fazer com que a resposta seja idêntica à chave.

Comparada a pontuações com precisão, cobertura e o número de crossing-brackets, a pontuação de contagem de erro é intuitivamente mais significativa. Uma vez que as fronteiras dos sintagmas por si só não são usadas diretamente na interpretação semântica, é dificil prever o quanto a falta de tais fronteiras interfere.

Carroll, Briscoe, e Sanlippo (1998) (também em Carroll e Briscoe (2002) e Carroll, Minnen, e Briscoe (2002)), descrevem uma nova técnica baseada em dependência que supera as limitações das propostas anteriores. O esquema de relação gramatical (Carroll, Briscoe, Calzolari, Federici, Montemagni, Pirrelli, Grefenstette, Sanlippo, Carrol, e Rooth 1997) é somente superficialmente similar ao esquema proposto por Lin. É construído sobre modificações e refinamentos para o padrão EAGLES (Sanfilippo e et al. 1996). Uma relação gramatical (GR) especifica a dependência sintática que existe entre um núcleo e seu dependente. GRs são organizadas hierarquicamente. A avaliação de parsers usando

o esquema de análise gramatical é baseada nas medidas de precisão e cobertura, agora definidas como:

1. Cobertura - o número de GRs retornado pelo parser que são iguais às GRs na sentença anotada correspondente, dividido pelo número total de GRs na sentença anotada.

2. Precisão - o número de GRs retornado pelo parser que são iguais às GRs na sentença anotada correspondente, dividido pelo número total de GRs retornado pelo parser para aquela sentença.

3. F-score - uma medida combinando precisão e cobertura num único cálculo:
   F-score = $\frac{(2*cobertura*precisão)}{cobertura+precisão}$

Segundo Carroll, em (Carroll, Minnen, e Briscoe 1999), esse esquema foi usado para o projeto SPARKLE para analisar alguns de seus parsers.

Modelo de Relação

Srinivas, Doran, Hockey, e Joshi (1996) e Srinivas, Sarkar, Doran, e Hockey (1998), propõem um método alternativo de avaliação comparativa que supera as limitações do PARSEVAL, e pode ser usado tanto para avaliar análises completas quanto parciais. O objetivo do modelo de relação é comparar os parsers independentemente de suas representações de saída. Eles sugerem uma representação normalizada da saída do parser, usando *chunks* e ligações de dependência.

De acordo com a literatura (Carroll, Briscoe, Calzolari, Federici, Montemagni, Pirrelli, Grefenstette, Sanlippo, Carrol, e Rooth 1997), um *chunk* pode ser uma unidade de tokens adjacentes de textos que são ligados por dependências. Cada *chunk* identificado é etiquetado com sua categoria - adjetivo, advérbio, verbo finito, nominal, preposicional, particípio. As categorias *chunk* são mais granulares que as categorias dos constituintes normalmente usadas.

Srinivas, Sarkar, Doran, e Hockey (1998) propõem normalizar as saídas dos parsers que produzem análises altamente estruturadas, baseadas em constituintes, por exemplo, através do "achatamento" dos constituintes sintagmais para *chunks*. Então, a precisão da estrutura pode ser medida em termos de precisão de *chunks* e das relações entre *chunks* (expressos como ligações de dependência entre os núcleos). Análises fragmentadas são penalizadas porque os constituintes não atachados são ligações de dependência faltantes.

(|ncsubj| |describe:2_VV0| |We:1_PPIS2| _)
(|dobj| |describe:2_VV0| |approach:7_NN1| _)
(|xcomp| |to:8_II| |describe:2_VV0| |statistical:9_JJ|)
(|xcomp| |to:8_II| |describe:2_VV0| |available:23_JJ|)
(|ncsubj| |incorporate+ed:11_VVN| |parsing:10_NN1| _)
(|iobj| |into:12_II| |incorporate+ed:11_VVN| |release:15_NN1|)
(|ncmod| _ |approach:7_NN1| |domain-independent:6_NN1|)
(|ncmod| _ |approach:7_NN1| |accurate:5_JJ|)
(|ncmod| _ |approach:7_NN1| |robust:4_JJ|)
(|detmod| _ |approach:7_NN1| |a:3_AT1|)
(|detmod| _ |ANLT:18_NP1| |the:17_AT|)
(|ncmod| _ |toolkit:19_NN1| |ANLT:18_NP1|)
(|ncmod| |of:16_IO| |release:15_NN1| |toolkit:19_NN1|)
(|ncmod| _ |release:15_NN1| |new:14_JJ|)
(|detmod| _ |release:15_NN1| |the:13_AT|)
(|mod| _ |statistical:9_JJ| |incorporate+ed:11_VVN|)
(|ncmod| _ |tool:27_NN1| |research:26_NN1|)
(|detmod| _ |tool:27_NN1| |a:25_AT1|)
(|ncmod| |as:24_II| |available:23_JJ| |tool:27_NN1|)
(|mod| _ |available:23_JJ| |publicly:22_RR|)
(|conj| _ |statistical:9_JJ| |available:23_JJ|)

Figura B.1: Relações Gramaticais para a sentença "We describe a robust accurate domain-independent approach to statistical parsing incorporated into the new release of the ANLT toolkit, and publicly available as a research tool.".

B.2.2 Avaliação usada no projeto RASP

Na avaliação do parser no projeto RASP, os autores defendem, conforme observado em (Carroll, Briscoe, e Sanlippo 1998), o uso de medidas como relações gramaticais recuperadas (ou dependências, como as usadas em (Lin 1995), (Lin 1998) e (Collins 1999)) em contraposição ao uso de medidas de similaridade de árvore como critérios para avaliação. A Figura B.1 mostra um exemplo de relações gramaticais que podem ser produzidas para uma análise. Uma relação gramatical é um par núcleo-dependente, e alguns parâmetros extras, dependendo da relação. E, conforme pode ser visto na Figura B.2, o sistema também pode produzir as relações gramaticais para as n melhores análises da entrada. Os pesos são as probabilidades de cada relação em todas as análises.

B.3 Conclusão

Este Apêndice apresentou algumas medidas de avaliação para parsers/gramáticas, de domínio irrestrito ou não, propostos pela literatura. Cada uma dessas medidas têm pontos fortes e fracos, e devem ser usadas conforme a característica do sistema que se quer avaliar.

```
0.353004      (|ncmod| |to:8_II| |approach:7_NN1| |available:23_JJ|)
 0.858024     (|ncmod| _ |approach:7_NN1| |robust:4_JJ|)
 0.294078     (|ncmod| |into:12_II| |incorporate+ed:11_VVN| |release:15_NN1|)
 0.858024     (|detmod| _ |approach:7_NN1| |a:3_AT1|)
 2.464891e-2  (|ncmod| _ |incorporate+ed:11_VVN| |into:12_II|)
 1.0          (|ncmod| |of:16_IO| |release:15_NN1| |toolkit:19_NN1|)
 0.141976     (|obj2| |describe:2_VV0| |approach:7_NN1|)
 1.0          (|ncsubj| |incorporate+ed:11_VVN| |parsing:10_NN1| _)
 0.103040     (|ncmod| |as:24_II| |describe:2_VV0| |tool:27_NN1|)
 0.676592     (|iobj| |into:12_II| |incorporate+ed:11_VVN| |release:15_NN1|)
 1.0          (|ncmod| _ |tool:27_NN1| |research:26_NN1|)
 1.0          (|ncmod| _ |release:15_NN1| |new:14_JJ|)
 0.858024     (|ncmod| _ |approach:7_NN1| |accurate:5_JJ|)
 0.289025     (|xcomp| |to:8_II| |describe:2_VV0| |statistical:9_JJ|)
 0.141976     (|dobj| |describe:2_VV0| |domain-independent:6_NN1| _)
 0.141976     (|detmod| _ |domain-independent:6_NN1| |a:3_AT1|)
 0.289025     (|xcomp| |to:8_II| |describe:2_VV0| |available:23_JJ|)
 1.0          (|detmod| _ |ANLT:18_NP1| |the:17_AT|)
 1.0          (|ncsubj| |describe:2_VV0| |We:1_PPIS2| _)
 2.464891e-2  (|dobj| |incorporate+ed:11_VVN| |release:15_NN1| _)
 1.0          (|conj| _ |statistical:9_JJ| |available:23_JJ|)
 0.759199     (|ncmod| |as:24_II| |available:23_JJ| |tool:27_NN1|)
 2.324469e-2  (|iobj| |as:24_CSA| |describe:2_VV0| |tool:27_NN1|)
 1.0          (|detmod| _ |tool:27_NN1| |a:25_AT1|)
 0.318363     (|ncmod| |to:8_II| |describe:2_VV0| |available:23_JJ|)
 0.353004     (|ncmod| |to:8_II| |approach:7_NN1| |statistical:9_JJ|)
 1.0          (|mod| _ |available:23_JJ| |publicly:22_RR|)
 3.960703e-2  (|xcomp| _ |describe:2_VV0| |statistical:9_JJ|)
 3.960703e-2  (|xcomp| _ |describe:2_VV0| |available:23_JJ|)
 0.141976     (|ncmod| _ |domain-independent:6_NN1| |robust:4_JJ|)
 4.628510e-2  (|iobj| |as:24_II| |describe:2_VV0| |tool:27_NN1|)
 0.141976     (|ncmod| _ |domain-independent:6_NN1| |accurate:5_JJ|)
 1.0          (|detmod| _ |release:15_NN1| |the:13_AT|)
 0.858024     (|dobj| |describe:2_VV0| |approach:7_NN1| _)
 4.681227e-3  (|ncmod| |into:12_II| |statistical:9_JJ| |release:15_NN1|)
 1.0          (|ncmod| _ |toolkit:19_NN1| |ANLT:18_NP1|)
 1.508456e-2  (|ncmod| |as:24_II| |approach:7_NN1| |tool:27_NN1|)
 1.0          (|mod| _ |statistical:9_JJ| |incorporate+ed:11_VVN|)
 0.858024     (|ncmod| _ |approach:7_NN1| |domain-independent:6_NN1|)
 0.318363     (|ncmod| |to:8_II| |describe:2_VV0| |statistical:9_JJ|)
 5.314655e-2  (|ncmod| |as:24_CSA| |describe:2_VV0| |tool:27_NN1|)
```

Figura B.2: Relações Gramaticais com pesos associados para a sentença “We describe a robust accurate domain-independent approach to statistical parsing incorporated into the new release of the ANLT toolkit, and publicly available as a research tool.”.

Apêndice C

Fundamentos do chart parser

O processamento automático da sentença com o objetivo do reconhecimento de sua estrutura sintática recebe tradicionalmente o nome de *parsing*. Por extensão, a ferramenta que executa esse conjunto de procedimentos que assinala funções sintáticas a cada um dos itens lexicais que compõem a sentença é chamada de *parser*.

Com a ajuda de uma gramática e de um léxico, o parser realiza a análise de uma sentença, que consiste na recuperação das funções sintáticas desempenhadas pelos itens lexicais nessa sentença. Um tipo de parser bastante comum na literatura é o chart parser, cuja grande inovação, com relação aos parsers que fazem análise de cima para baixo *top-down* e os que fazem análise de baixo para cima *bottom-up*, é o acréscimo de uma espécie de memória. Essa mémoria possibilita o armazenamento de todo o "trabalho" desempenhado pelo parser durante a análise. A base do mecanismo de armazenamento vem das chamadas *Well-Formed Substring Tables (WFSTs)* ((Atwell e Pocock 1994b), (Gazdar e Mellish 1989)). Elas podem ser representadas como grafos acíclicos dirigidos, cujos nós são rotulados de *0* (primeiro nó) até n (último nó), em que n é o número de palavras na cadeia a ser analisada. Os arcos do grafo são rotulados com categorias sintáticas e palavras. Uma WFST pode ser representada como um *conjunto de arcos (edges)*, na qual um arco é uma estrutura com os seguintes atributos:

- <começo> = ... algum inteiro ...
- <fim> = ... algum inteiro ...
- <rótulo> = ... alguma categoria ...

Os arcos carregam os rótulos das categorias sintáticas, e os nós têm essencialmente

nomes arbitrários (como inteiros de 0 a n). A Figura C.1 mostra um exemplo desse tipo de análise para a sentença *Eu viajo*. A codificação é feita de forma direta, ou seja, um constituinte com um arco indo de i para j na tabela precede um constituinte com arco de m para n (em que $j < m$).

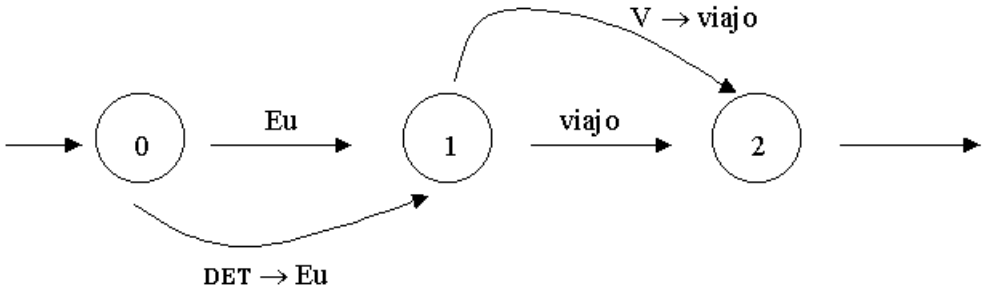

Figura C.1: Representação WFST para a sentença *Eu viajo*

As WFSTs, ao contrário das árvores, que codificam informações de dominância entre constituintes, não são capazes de tal representação. Por exemplo, se existe um arco representando a regra $S \rightarrow SN\ SV$, não há nada explícito conectando o arco S aos arcos SN e SV.

Algumas razões são observadas em favor do uso das WFSTs:

- Evitar a repetição do mesmo trabalho várias vezes.
  Por exemplo, armazenando as várias análises parciais de SN e SV na regra em que aparecem, de tal forma que não seja mais necessário que sejam descobertas em tentativas posteriores de analisar o mesmo texto.

- Representar estruturas sintáticas alternativas para a mesma cadeia de caracteres. Muitas sentenças são ambígüas sintaticamente e, sendo assim, têm mais de uma estrutura sintática possível. Cada análise pode ser representada por uma única árvore sintática, mais isso é ineficiente para sentenças longas e ambígüas.

- Armazenar análises parciais das sentenças que não podem ser analisadas inteiramente.

- Eficiência na análise. Os parsers usando WFST são capazes de encontrar a primeira análise de uma sentença em tempo de $O(n^3)$, enquanto os algoritmos mais simples são tipicamente exponenciais.

- Neutralidade. As WFSTs podem ser usadas por qualquer tipo de parser: os que produzem a análise de cima para baixo, os que produzem a análise de baixo para cima, etc.

Representação de hipóteses

O uso de WFSTs representa um ganho em tempo de processamento, uma vez que todas as análises com sucesso são armazenadas para um uso posterior. Um problema ocorre com as análises de subcadeias que não tiveram sucesso (hipóteses tentadas), que não são armazenadas, fazendo com que o parser gaste tempo reinvestido nelas. A única maneira de evitar duplicações de tentativas prévias é ter uma representação explícita prévia sobre as diferentes hipóteses e metas que o parser tem em cada passo da análise. As WFSTs não são capazes de representar análises parciais juntamente com as hipóteses que já tentaram e que falharam. Por exemplo, considere um objeto que tem como intenção representar o estado de análise em que uma cadeia de caracteres pode consistir de uma seqüência SN V SN[1]. É explorada a hipótese de que uma cadeia pode ser analisada como um S, consistindo de uma seqüência SN SV[2]. Há a necessidade de se estabelecer que a seqüência V SN pode ser coberta por SV (da regra S).

O exemplo apresentado é uma análise parcial com hipóteses. Para representar isso, algumas características extras necessitam ser adicionadas às WFSTs. Como resultado dessa mudança, surgem os chamados *chart ativos*.

Os chart ativos

Duas mudanças são requeridas na estrutura de dados WFST para se obter um chart ativo: arcos vazios e arcos rotulados com regras marcadas por pontos (*dotted rules*). A primeira extensão permite que os arcos sejam acíclicos, ou seja, voltam para o nó do qual partiram, com a restrição de que não deve haver nenhum nó intermediário nesse trajeto. Na segunda extensão, os arcos que antes eram marcados com categorias sintáticas agora são marcados com regras gramaticais com pontos. Essas extensões na WFST formam estruturas conhecidas como *chart*, e a família dos parsers que exploram o chart como estrutura de dados é chamada de chart parser(Atwell e Pocock 1994a). Como acontece com a WFST, um chart é completamente neutro com respeito à estratégia de parsing.

O chart consiste de uma coleção de vértices (nós), um para cada palavra da entrada, conectada por arcos (edges) rotulados com informação gramatical. O chart, ao contrário

[1]SN = sintagma nominal; V = verbo
[2]S = sentença; SV = sintagma verbal

da árvore, pode ter vários arcos apontando para o mesmo nó, e ainda permite que haja ciclos. As regras marcadas são usadas para representar estágios no processo de parsing. Elas se parecem com regras normais, com pontos em algum lugar do lado direito. Para uma regra S → SN SV, são três as opções de inserção de pontos:

- S → . SN SV
  Esse tipo de rótulo só ocorrerá no arco que volta à origem do nó que emergiu. Denota a hipótese de que um S pode ser encontrado como uma subcadeia que começa do nó em questão, tão logo uma cadeia seja coberta pela seqüência SN SV. Isso indica que tal hipótese pode ter sido feita, mas nenhuma parte foi verificada.
- S → SN . SV
  Indica que uma hipótese pode ser parcialmente confirmada. Ele encontrou um arco que cobre um SN, e assim indica que a primeira parte da hipótese foi confirmada.
- S → SN SV .
  Indica uma hipótese confirmada; SN e SV foram verificados, e assim, a regra S. Poderá ser encontrado um arco que cobre uma cadeia feita de uma subcadeia SN seguido por uma subcadeia SV.

Assim, o chart é composto por quatro elementos: vértices, que são nós no chart; edges, arcos no chart; arcos ativos, representando hipóteses; arcos inativos, representando hipóteses confirmadas inteiramente. Os campos do chart passam a ser:

- <começo> = ... algum inteiro ...
- <fim> = ... algum inteiro ...
- <rótulo> = ... alguma categoria ...
- <encontrado> = ... alguma seqüência de categorias ...
- <aSerEncontrado> = ... alguma seqüência de categorias ...

em que <rótulo> é o LHS (lado esquerdo) da regra apropriada, <encontrado> é a seqüência de categorias RHS (lado direito) à esquerda do ponto, e <aSerEncontrado> é a seqüência de categorias à direita do ponto. Se for vazio, será um arco inativo.

Inicializando o Chart

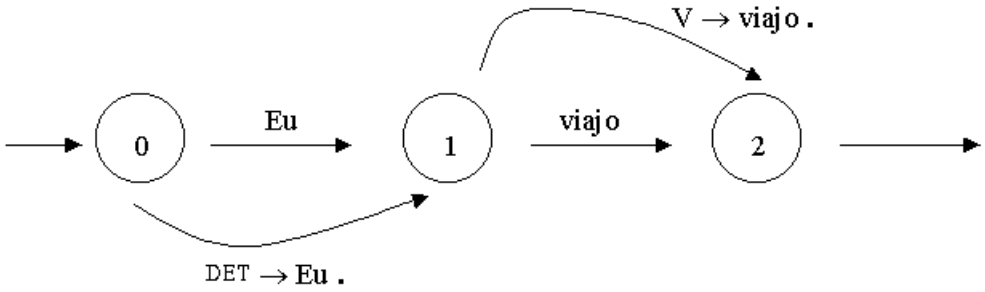

Figura C.2: Exemplo de chart parser inicial para a sentença *eu viajo*.

Antes de disparar o processo de análise, a estrutura de dados do chart tem que ser iniciada. Há tipicamente dois estágios para isso:

1. Se há n palavras na cadeia de entrada, então o chart conterá $n+1$ nós, rotulados de 0 a n. Para cada palavra em i na cadeia de entrada, crie um arco entre um par de nós adjacentes $i-1$ e i, e rotule-o com essa palavra.

2. Procure cada palavra no léxico para encontrar quais as categorias sintáticas às quais ela pertence. O resultado pode ser colocado diretamente no chart como arcos inativos.

Por exemplo, dada a cadeia *Eu viajo*, uma gramática e um léxico da língua, o chart seguinte pode ser inicializado como na Figura C.2.

Os nós inativos não são suficientes para o processo de análise. É necessário que sejam criados nós ativos. Um simples princípio para isso é: cada vez que é adicionado um arco inativo com rótulo C no chart, adicione um arco ativo vazio começando do mesmo nó, para cada regra na gramática que requer um constituinte C como seu filho mais à esquerda. Um arco vazio é um arco ativo com nenhum constituinte encontrado, e que começa e termina no mesmo nó. Isso resultará numa estratégia de parsing de baixo para cima, que pode ser formalmente especificado como segue:

- Se está sendo adicionado um arco $< i, j, C \rightarrow W1\ .\ >$ ao chart, então para cada regra na gramática na forma $B \rightarrow C\ W2$, adicione um arco $< i, i, B \rightarrow .\ C\ W2 >$ ao chart.

Se essa regra for aplicada no chart Figura C.2, o resultado obtido seria o mostrado na Figura C.3. Os arcos ativos adicionados ao chart são correspondentes a regras na

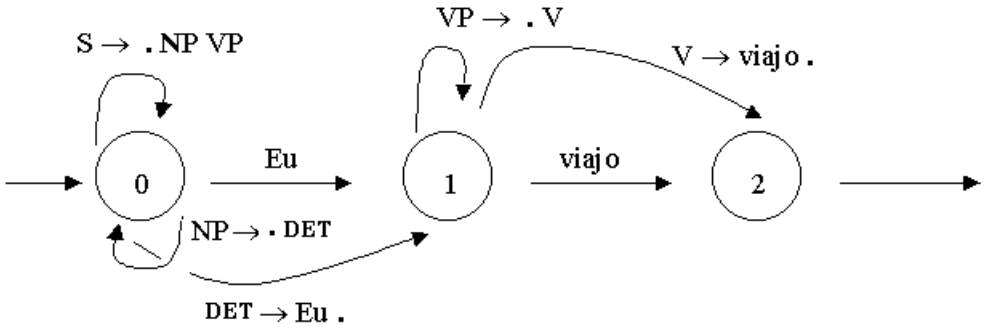

Figura C.3: Exemplo de chart parser para a sentença *eu viajo*.

gramática cujos filhos mais à esquerda já foram encontrados (isto é, arcos inativos $SN \rightarrow Eu$ e $V \rightarrow viajo$). Assim, a opção de baixo para cima produz uma estratégia de busca no sentido esquerda para direita. Uma nova regra é recuperada da gramática tão logo o elemento mais à esquerda do seu lado direito seja encontrado. Além das estratégias de análise de baixo para cima, outras estratégias de invocação também podem ser usadas.

Regra Fundamental do Chart Parsing

Essa regra de baixo para cima não é suficiente para o parsing, uma vez que somente adiciona arcos vazios ao chart. Um princípio extra é requerido, e é chamado regra fundamental do chart parsing. A regra fundamental nos diz como adicionar mais arcos ao chart, e pode levar à adição de arcos tanto ativos quanto inativos. Pode ser descrita como: se um arco ativo encontra um arco inativo de rótulo C, e se o constituinte mais à esquerda do campo <aSerEncontrado> do edge ativo é a categoria C, então ponha um novo arco no chart que se estende tanto por arcos ativos quanto por inativos.

Mais formalmente especificando:

- Se um chart contém arcos $<i, j, A \rightarrow W1\ .\ B\ W2>$ e $<j, k, B \rightarrow W3\ .>$, em que A e B são categorias e $W1$,$W2$ e $W3$ são seqüências de categorias ou palavras, então adicione ao chart, um novo arco $<i, k, A \rightarrow W1\ B\ .\ W2>$.

Exemplo: ao aplicar a regra fundamental ao chart anterior (em que a regra de baixo para cima foi aplicada) o resultado será um novo chart contendo novos arcos. Considere o arco ativo <0,0,SN → . DET>. A hipótese não está confirmada, sendo necessário que exista um arco inativo no chart começando pelo vértice 0. Existe tal arco <0,1,DET → *eu* . >, que confirma inteiramente a hipótese. Aplicando novamente a regra fundamental, um novo arco ativo <0,0,S → . SN SV> é inserido. Ele representa a hipótese sobre uma

sentença consistindo de um SN seguido por um SV, mas totalmente não confirmada. Para começar a satisfazer essa hipótese, é necessário que seja encontrado um arco inativo SN no chart que comece no vértice 0. O arco <0,1,SN → DET . > é desse tipo, significando que a hipótese pode ser parcialmente confirmada, representando-se isso pelo arco <0,1,S → SN . SV> a ser inserido no chart. Esse novo arco requer um SV para se confirmar totalmente, mas ele não existe.

O arco <1,1,SV → . V> representa a hipótese sobre um SV consistindo de um verbo. Como a hipótese da sentença acima, ela está totalmente não confirmada, sendo que a mesma requer um arco inativo no chart começando pelo vértice 1. Existe tal arco <1,2,V → *viajo* . >, que confirma inteiramente a hipótese SV, que passa a ser representada com a adição de um novo arco inativo <1,2,SV → V . > ao chart.

Retornando ao chart ativo <0,1,S → SN . SV>, pode-se ver que a hipótese é também totalmente confirmada, uma vez que há agora um arco inativo SN que começa no vértice 1. Assim, pode-se adicionar um arco inativo <0,2,S → SN SV . > ao chart, produzindo a análise completa para a sentença *eu viajo* por toda a extensão do chart.

O processo de análise

Quando o chart é inicializado, a regra de baixo para cima produzirá um número de arcos ativos. Esses arcos proverão trabalho para a regra fundamental, e, para cada arco inativo que é produzido, há também a possibilidade de que a regra de baixo para cima seja aplicada, produzindo ainda mais arcos para a regra fundamental. Lembrando sempre que é importante a checagem se o arco já existe quando se está inserindo, para evitar a duplicação dos mesmos. Esse processo continua até que as regras de baixo para cima e fundamental não produzam mais arcos. Se há um ou mais arcos inativos S expandindo por todo o chart, então a cadeia dada pode ser inteiramente analisada. Se não, a cadeia é agramatical de acordo com o fragmento da gramática usado.

A agenda

Cada arco ativo representa uma hipótese a ser explorada. Se está tendo sucesso, mesmo parcialmente, com alguns constituintes confirmados, é desejável que sejam geradas novas hipóteses, de acordo com a operação da regra fundamental e da estratégia de invocação adotadas. Se há mais de uma hipótese gerada por vez, torna-se necessário decidir em que ordem analisar as diferentes hipóteses. A base na qual essa decisão é tomada é chamada de *estratégia de busca*.

No chart parsing é uma prática usar uma estrutura de dados, uma *agenda*, para armazenar novas hipóteses ou arcos a serem adicionados ao chart. Ou seja, os novos arcos são imediatamente adicionados à agenda para, somente posteriormente, serem inseridos no chart.

A agenda pode ser vista simplesmente como um mecanismo de armazenamento em que os arcos esperam para serem adicionados ao chart: quando um novo arco é criado, ao invés de pô-lo imediatamente no chart, o mesmo fica aguardando na agenda, enquanto chega a sua vez de ser adicionado.

Para determinar a ordem na qual os arcos são adicionados, a agenda usa estratégias de busca conhecidas como *depth first* ou *breath first*, representadas respectivamente por estruturas como pilhas ou filas para armazenar os arcos.

Extraindo análises do chart parsing

Quando o algoritmo do chart parsing termina, ele retorna um chart inteiro, mas na verdade o que queremos é uma árvore sintática (ou árvores sintáticas). Para isso, é necessário que exista uma rotina que mantenha uma lista de todos os arcos utilizados para se chegar ao final da análise. Essa rotina deve ser capaz de combinar dois arcos filhos para produzir um arco pai de forma que seja armazenada no arco pai a lista de seus filhos. Assim, quando a análise acaba, só é necessário procurar no chart por um arco que começa com 0 e termine em n, e recursivamente reproduzir a árvore sintática a partir de seus filhos. A complicação surge quando o problema é decidir o que fazer com as análises ambígüas. O problema com tal tipo de implementação é resolvido através de um mecanismo chamado *packed forest*(Russel e Norvig 1995), que é equivalente a um conjunto de árvores, mas eficientemente reunidas em uma única estrutura.

Vantagens do chart parsing

O uso do chart oferece três vantagens:

- Evitar replicação de trabalho
  Quando as partes da entrada são analisadas com sucesso, elas são adicionadas ao chart. A idéia é que, antes de tentar analisar qualquer parte da entrada, procurar no chart para ver se já existe um constituinte igual analisado previamente, e, se sim, não fazê-lo de novo.

- Ambigüidade local
  O chart parser provê uma representação compacta da ambigüidade local. Trechos

ambígüos são todos colocados no chart. A ambigüidade passa a ser desfeita uma vez que os trechos podem diferir dos locais em que terminam. Um princípio básico do chart para evitar duplicação é representar tudo, mas somente uma vez cada informação.

- Análise parcial
  Mesmo se o parsing geral falha (isto é, não se pode obter um arco que se estende por toda a entrada, e conseqüentemente, não se pode construir uma representação da entrada completa), o chart ainda contém informações úteis. Pode-se, por exemplo, recuperar a menor seqüência de arcos suficientes para cobrir a entrada inteira.

Referências

Abney, S. (1996). Part-of-speech tagging and partial parsing. In *Church, K. et al.(ed.) Corpus-based Methods in Language and Speech.*

Allen, J. F. (1995). *Natural Language Understanding.* Menlo Park, California: Benjamim / Cummings.

Atwell, E. S. e R. J. Pocock (1994a). Chart parsing: Theory and implementation. In -, http://www.comp.leeds.ac.uk/nti-kbs/ai5/Chart/chart.html.

Atwell, E. S. e R. J. Pocock (1994b). Well-formed substring tables and charts. chart parsing: Theory and implementation. In -, http://www.comp.leeds.ac.uk/nti-kbs/ai5/Wfst/wfst.html.

Baker, J. K. (1975). Stochastic modeling for automatic speech understanding. In *Speech Recognition*, pp. 521–542.

Berger, A., S. D. Pietra, e V. D. Pietra (1996). A maximum entropy approach to natural language processing. In *Computional Linguistics*, Volume 22, pp. 39–71.

Bick, E. (2000). *The Parsing System "Palavras" - Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework.* Aarhus University Press.

Bikel, D. M. (2000). A statistical model for parsing and word-sense disambiguation. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora.*

Bikel, D. M. e D. Chiang (2000). Two statistical parsing models applied to the chinese treebank. In *Proceedings of the Second Chinese Language Processing Workshop*, Hong Kong, pp. 1–6.

Black, E. e et al. (1991). A procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop.*

Black, E. e et al. (1992). Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 5th DARPA Speech and Natural Language Workshop*, Harriman, NY.

Black, E., R. Garside, e G. L. (Eds.) (1993). Statistically-driven computer grammars of english: The ibm/lancaster approach. In -, Amsterdam, The Netherlands: Rodopi.

Black, E., J. Lafferty, e S. Roukos (1992). Development and evaluation of a broad-coverage probabilistic grammar of english-language computer manuals. In *Proceedings of the Association for Computational Linguistics*.

Bod, R. (1993). Using an annotated language corpus as a virtual stochastic grammar. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, AAAI Press/MIT Press, Menlo Park, pp. 778–783.

Bonfante, A. G. e M. G. V. Nunes (1997). Redes neurais e correção gramatical do português: um estudo de caso. In *III Congresso Brasileiro de Redes Neurais)*, Florianópolis, SC, pp. 262–267.

Bonfante, A. G. e M. G. V. Nunes (1999). Connectionist learning applied to a brazilian portuguese grammar checking. In *Proceedings of the First Workshop on Natural Language Processing and Neural Networks (NLPNN'99)*, Beijing, China, pp. 9–13.

Bonfante, A. G. e M. G. V. Nunes (2000). Um parser probabilístico para o português brasileiro: um estudo exploratório. In *Probabilistic Reasoning in Artificial Intelligence)*, Atibaia, SP, pp. 9–13.

Bonfante, A. G. e M. G. V. Nunes (2001). The implementation process of a statistical parser for brazilian portuguese. In *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT2001))*, Beijing, China, pp. 217–221.

Bonfante, A. G. e M. G. V. Nunes (2002). Parsing probabilístico para o português do brasil. In *I Workshop de Teses e Dissertações em Inteligência Artificial (I WTDIA))*, Porto de Galinhas, Recife, pp. 10–18.

Booth, T. e R. Thompson (1973). Applying probability measures to abstract languages. In *IEEE Transactions on Computers*, Number 22(5).

Breiman, L., J. H. Friedman, R. A. Olshen, e C. J. Stone (1984). *Classification and Regression Trees*. Wadsworth and Brooks, Pacific Grove, California.

Brill, E. (1993). Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, pp. 259–265.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. In *Computational Linguistics*, Number 21(4), pp. 543–565.

Brill, E. e R. J. Mooney (1997). An overview of empirical natural language processing. In *AI Magazine*, pp. 13–24.

Briscoe, E. e J. Carrol (1995a). Developing and evaluating a probabilistic lp parser of part-of-speech and punctuation labels. In *Proceedings of the 4th International Workshop on Parsing Technologies (IWPT'95)*, pp. 48–58.

Briscoe, E. e J. Carrol (1995b). Developing and evaluating a probabilistic lr parser of part-of-speech and punctuation labels. In *Proceedings of the 4th ACL/SIGPARSE Intenational Workshop on Parsing Technologies*, Prague, Czech Republic, pp. 48–48.

Briscoe, E. e J. Carrol (2002). Robust accurate statistical annotation of general text. In *In Proceedings of the 3rd International Conference on Language Resources and Evaluation*, Las Palmas, Gran Canaria, pp. 1499–1504.

Carabalho, S. A. e E. Charniak (1998). New figures of merit for best-first probabilistic chart parsing. In *Computational Linguistics*, Number 24, pp. 275–298.

Carrol, J. (1994). Relating complexity to practical performance in parsing with wide-coverage unification grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, NM, pp. 287–294.

Carroll, J. e E. Briscoe (2002). High precision extraction of grammatical relations. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, Taipei, Taiwan.

Carroll, J., E. Briscoe, N. Calzolari, S. Federici, S. Montemagni, V. Pirrelli, G. Grefenstette, A. Sanlippo, G. Carrol, e M. Rooth (1997). Sparkle wp1 specification of phrasal parsing.

Carroll, J., E. Briscoe, e A. Sanlippo (1998). Parser evaluation : A survey and a new proposal.

Carroll, J., G. Minnen, e E. Briscoe (2002). Parser evaluation using a grammatical relation annotation scheme.

Carroll, J., G. Minnen, e T. Briscoe (1999). Corpus annotation for parser evaluation.

Charniak, E. (1993). *Statistical Language Learning*. MIT Press.

Charniak, E. (1996). Tree-bank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, AAAI Press/ MIT Press, Menlo Park, pp. 1031–1036.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, Seattle, Washington, USA.

Chi, Z. (1998). *Probability models for complex systems*. Ph. D. thesis, Brown University.

Chiang, D. (2000). Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Meeting of the Association for Computational Linguistics (ACL)*, pp. 456–463.

Chomsky, N. (1965). *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.

Chomsky, N. (1995). *The Minimalist Program*. Cambridge, Mass,: The MIT Press.

Church, K. e R. L. Mercer (1993). Introduction to the special issue on computational linguistics using large corpora. In *Computational Linguistics*, Number 19(1), pp. 1–24.

Church, K. e R. Patil (1982). Coping with syntactic ambiguity or how to put the block in the box on the table. In *American Journal of Computational Linguistics*, Number 8(3-4), pp. 139–149.

Clark, S., J. Hockenmaier, e M. Steedman (2002). Building deep dependency structures with a wide-coverage ccg parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, PA.

Collins, M. J. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the* 34^{th} *Annual Meeting of the Association for Computatational Linguistics*, pp. 184–191.

Collins, M. J. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics.*

Collins, M. J. (1999). *Head-Driven Statistical Models for Natural Language Parsing.* Ph. D. thesis, University of Pennsylvania.

Collins, M. J. (2000). Discriminative raranking for natural language parsing. In *ICML.*

EAGLES (1995). Evaluation of natural language processing systems.

Elman, J. L. (1990). Finding structure in time. In *Cognitive Science*, Number 14, pp. 179–211.

Gazdar, G. e C. Mellish (1989). *Natural Language Processing in Prolog.* Addison-Wesley.

Goodman, J. (1996a). Parsing algorithms and metrics. In *Proceedings of the* 34^{th} *Annual Meeting of the ACL*, pp. 177–183.

Goodman, J. (1996b). Parsing algorithms and metrics. In A. Joshi e M. Palmer (Eds.), *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, San Francisco, pp. 177–183. Morgan Kaufmann Publishers.

Grover, C., J. Carrol, e T. Briscoe (1993). *The Alvey Natural Tools Grammar.* 4th. release edition.

Harrison, P., S. Abney, D. Fleckenger, C. Gdaniec, P. Grishman, D. Hindle, B. Ingria, M. Marcus, B. Santorini, e T. Strzalkowski (1991). Evaluating syntax performance of parser/grammars on english. In *Proceedings of the Workshop on Evaluating Natural Language Processing Systems, ACL.*

Hermjakob, U. e R. Mooney (1997). Learning parse and translation decisions from examples with rich context. In *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics*, Somerset, N. J., pp. 482–489. Association for Computational Linguistics.

Hockenmaier, J. e M. Steedman (2002). Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of the 40th Meeting of the ACL*, Philadelphia, PA.

Inui, K. e et al. (1997). A new formalization of probabilistic glr parsing.

Jelinek, F. (1990). Self-organized language modeling for speech recognition. In *Readings in Speech Recognition*, Edited by Waibel and Lee. Morgan Kaufmann Publishers.

Jelinek, F. e et al. (1994). Decision tree parsing using a hidden derivation model. In *Proceedings of the 1994 Human Language Technology Workshop*, pp. 272–277.

Jelinek, F. e J. D. Lafferty (1991). Computation of the probability of initial substring generation by stochastic context-free grammars. In *Computational Linguistics*, Number 17, pp. 315–324.

John, M. F. S. e J. L. McClelland (1990). Learning and applying contextual constraints in sentence comprehension. In *Artificial Intelligence*, Number 46, pp. 217–257.

Kaplan, R. e J. Bresnan (1982). Lexical-functional grammar: a formal system for gramatical representation. In *The Mental Representation of Grammatical Relations*, pp. 173–281.

Karlsson, F. (1990). Constraint grammar as a framework for parsing running text. In *Karlgren, Hans (ed.), COLING-90: Proceedings of the 13th International Conference on Computational Linguistics*, Volume 3, pp. 168–173.

Kupiec, J. M. (1991). A trellis-based algorithm for estimating the parameters of a hidden stochastic context-free grammar. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop*, pp. 241–246.

Kupiec, J. M. (1992). Robust part-of-speech tagging using a hidden markov model. In *Computer Speech and Language*, Volume 6, pp. 225–242.

Lane, P. e J. Henderson (1998). Simple synchrony networks: Learning to parse natural language with temporal synchrony variable binding. In *Proceedings of the International Conference on Artificial Neural Networks*, pp. 615–620.

Lawrence, S., S. Fong, e C. L. Giles (1996). On the applicability of neural network and machine learning methodologies to natural language processing. In *Workshop on New Approaches to Learning for Natural Language Processing, IJCAI-95*, Montreal, Canadá, pp. 1–8.

Lin, D. (1995). A dependency-based method for evaluating broad-coverage parsers. In *IJCAI*, pp. 1420–1427.

Lin, D. (1998). Dependency-based evaluation of minipar. In *Proceedings of the Workshop at LREC'98 on the Evaluation of Parsing Systems*, Granada, Spain.

Luft, C. P. (1994). *Moderna Gramática Brasileira*. São Paulo: Editora Globo S.A.

Magerman, D. M. (1994). *Natural Language Parsing as Statistical Pattern Recognition*. Ph. D. thesis, Stanford University.

Magerman, D. M. (2003). Everything you always wanted to know about probability theory but were afraid to ask. Technical report, IBM T. J. Watson Research Center, http://www-cs-students.stanford.edu/ magerman/pubs.html.

Magerman, D. M. (June, 1995). Statistical decision-tree models for parsing. In *33rd Annual Meeting of the Association for Computational Linguistics - ACL Conference*, http://www-cs-students.stanford.edu/ magerman/pubs.html, pp. 276–283.

Magerman, D. M. e M. P. Marcus (April, 1991). Pearl: A probabilistic chart parser. In *European ACL*, http://www-cs-students.stanford.edu/ magerman/pubs.html.

Manning, C. D. e H. Schütze (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

Marcus, M. P. e et al. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics 19*(2), 313–330.

Martins, R. T., R. Hasegawa, e M. G. V. Nunes (2002). Curupira: um parser de português brasileiro. NILC-TR-02-24. Technical Report (In portuguese).

Miikkulainen, R. (1993). *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory*. Cambridge, MA: MIT Press.

Mitchell, T. M. (1980). The need for biases in learning generalizations. In *Readings in Machine Learning*, San Mateo, CA. Morgan Kaufmann.

Mooney, R. (1996). Inductive logic programming for natural language processing. In *Proceedings of the Sixth International Inductive Logic Programming Workshop on Logic Programming*, Stockholm, Sweeden. Springer Verlag.

Moore, R. C. (2000). Improved left-corner chart parsing for large context-free grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies - IWPT*.

Pereira, F. e Y. Schabes (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the February 1999 DARPA Speech and Natural Language Workshop*, pp. 122–127.

Pinheiro, G. M. e S. M. Aluísio (2003). Corpus nilc: Descrição e análise crítica com vistas ao projeto lacio-web. Relatórios Técnicos do ICMC - USP,São Carlos no. 190.

Poritz, A. B. (1988). Hidden markov models: A guided tour. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*.

Rashmi, P. e A. Sarkar (2000). Comparing test-suite based evaluation and corpus-based evaluation of a wide-coverage grammar for english.

Ratnaparkhi, A. (1999). Learning to parse natural language with maximum entropy models. *Machine Learning 34*, 151–176.

Reilly, R. G. e N. E. S. Eds. (1992). *Connectionist Approaches to Natural Language Processing*. Hilldale, NJ: Lawrence Erlbaum and Associates.

Russel, S. e P. Norvig (1995). *Artificial Intelligence: a modern approach*. New Jersey, USA: Prentice-Hall.

Sanfilippo, A. e et al. (1996). Subcategorization standards.

Sarkar, A. (2001). Applying co-training methods to statistical parsing. In *Proceedings of NAACL*, Pittsburg, PA.

Scarlett, E. (March 2000). An evaluation of a rule-based parser of english sentences.

Shastri, L. e V. Ajjanagadde (1993). From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. In *Behavioral and Brain Sciences*.

Srinivas, B., C. Doran, B. Hockey, e A. Joshi (1996). An approach to robust partial parsing and evaluation metrics. In J. Carroll (Ed.), *Proceedings of the Workshop on Robust Parsing - 8th European Summer School in Logic, Language and Information*, pp. 70–82.

Srinivas, B., A. Sarkar, C. Doran, e B. Hockey (1998). Grammar and parser evaluation in the xtag project. In *Proceedings of the International Conference on Language Resources and Evaluation.*, Granada, Spain.

Steedman, M. (2000). *The Syntactic Process*. Cambridge, MA: MIT Press.

Stolcke, A. (1995). An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. In *Computational Linguistics*, Number 21, pp. 165–202.

Waltz, D. L. (1978). English language question-answering system for a large relational database. In *Communications of the Association for Computing Machinery*, Number 21(7), New York: Elsevier, pp. 526–539.

Wermter, S., E. Rillof, e G. Scheler (1996). *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*. New York: Springer.

Winograd, T. (1970). *Understanding Natural Language*. San Diego, California: Academic.

Woods, W. A. (1977). Lunar rocks in natural english: Explorations in natural language question answering. In *Linguistic Structures Processing, eds. A Zampoli*, New York: Elsevier.

Zelle, J. M. e R. J. Mooney (1996). Learning to parse data base queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Menlo Park, Calif., pp. 1050–1055. American Association for Artificial Intelligence.