Pontifícia Universidade Católica do
Rio Grande do Sul
Faculdade de Informática
Curso de Bacharelado em
Ciência da Computação

Parsing Probabilístico para a Língua Portuguesa

Trabalho de Conclusão II

Autores:

Rodrigo R M Kochenburger

Marlon Gomes Lopes

Orientador:

Carlos Augusto Prolo

Porto Alegre, Dezembro de 2009

Resumo

Este trabalho descreve a construção de um análisador sintático (também conhecido pelo termo em inglês *parser*) para o português utilizando os modelos probabilísticos desenvolvidos por Michael Collins para aprendizado supervisionado a partir de *corpus* anotado. Para este fim é utilizada a ferramenta de desenvolvimento construída por Dan Bikel para os modelos de Collins. Como *treebank* alimentador do processo utilizamos o *corpus* Floresta Sintática. São descritos aqui aspectos fundamentais do desenvolvimento do *parser* que envolve extenso trabalho de sintonia dos parâmetros da ferramenta de Bikel, algoritmos de pré- e pós-processamento do *corpus*, e outras estratégias utilizadas buscando melhorar a acuidade do *parser*. O processo é inerentemente empírico. Utilizou-se um método incremental na busca de melhores configurações baseada em rigorosa avaliação quantitativa a cada passo, seguindo os critérios de precisão e *recall* de constituintes rotulados do PARSEVAL. Nossa melhor configuração tem precisão 72.08%, *recall* 71,51% e *F-Score* 71,79%, que é superior aos resultados obtidos em trabalhos semelhantes da literatura, para a lingua portuguesa. Resultados melhores foram obtidos durante testes com lematização, porém não podem ser computados como testes válidos, pois utilizam os lemas das palavras do próprio *corpus*. Estes resultados apenas servem como *upper bound* para avaliar a utilização de lematização real.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

CFG *Context Free Grammar* - Gramática Livre de Contexto

HPSG *Head-driven phrase structure grammar*

PCFG *Probabilistic Context Free Grammar* - Gramática Livre de Contexto Probabilística

POS *Part-of-Speech* - Partes do discurso

MBHA Modelo baseado na história da análise

PTB Penn TreeBank

1 Introdução

A linguagem é um dos aspectos mais fundamentais do comportamento humano e um componente crucial de nossas vidas. A linguagem em sua modalidade escrita serve como um registro a longo prazo do conhecimento que é passado de geração em geração. Na forma falada, ela serve diariamente como meio primário de coordenação do nosso comportamento e interatividade entre as pessoas.

A linguagem é estudada em diferentes meios ou áreas acadêmicas. Cada disciplina define os seus próprios problemas e possui seus próprios métodos para resolvê-los. A linguística, por exemplo, estuda a estrutura da própria linguagem, considerando perguntas como: a) por que certas combinações de palavras compõem uma sentença e outras não; ou b) por que algumas sentenças possuem significado e outras não. A psicolinguística estuda o processo de produção e compreensão da língua pelos seres humanos, levando em consideração perguntas como: a) como as pessoas escolhem, ou identificam, a estrutura apropriada das frases; ou b) como elas decidem os significados para cada palavra.

O objetivo da linguística computacional é utilizar os conhecimentos desenvolvidos nas áreas citadas - e outras relacionadas - para desenvolver teorias e aplicações utilizando as noções de algoritmos e estrutura de dados para processar, entender e também produzir sintática e semanticamente a língua, escrita ou falada.

Nas últimas duas décadas, o desenvolvimento de métodos estatísticos para o processamento de linguagem natural (PLN) vem sendo impulsionado pela evolução no poder de processamento dos computadores [MAN99, JUR00]. Esses métodos utilizam grande quantidade de dados, em conjunto com cálculos probabilísticos, para tentar "entender" corretamente a estrutura e o significado da linguagem. Fundamental nesse processo foi o surgimento de extensos *corpora* sintaticamente anotados (*treebanks*) [MAR93, MAR94, ABE03, SAR04].

Neste trabalho focou-se o processo de *parsing* probabilístico baseado em *corpus*, em particular, utilizando como base os estudos e o *parser* desenvolvido por Michael Collins

[COL99, COL97], na sua versão posterior reimplementada por Dan Bikel [BIK04], sendo construído um *parser* para a língua portuguesa.

1.1 Motivação

A motivação para o desenvolvimento deste trabalho pode ser dividida em dois aspectos principais: científico e tecnológico.

A motivação científica é a obtenção do conhecimento e o melhor entendimento a respeito de como as linguagens funcionam. Nenhuma das disciplinas tradicionais possui, isoladamente, ferramentas necessárias para decifrar completamente a produção e a compreensão linguísticas que os seres humanos possuem. Porém, é possível utilizar programas de computadores para implementar essa complexa teoria, de modo que seja possível testá-la, verificá-la e incrementalmente melhorá-la. Ao aprofundar o estudo deste processo, podemos desenvolver um entendimento a respeito de como os seres humanos processam as línguas.

Quanto à natureza tecnológica, a maior parte do conhecimento humano está armazenada de forma linguística, escrita ou falada, e computadores que conseguissem "entender" linguagem natural poderiam acessar toda essa informação. Outro aspecto relevante é a possibilidade de melhorar a interação humano-computador, aumentando o nível de acessibilidade, o que tornaria mais simples a utilização de ferramentas computacionais por pessoas com necessidades especiais.

Atualmente, a evolução do poder computacional e a construção de grandes *treebanks* possibilitam a utilização de técnicas mais avançadas, que utilizam grande quantidade de informação e processamento para tentar resolver esses problemas. O processo de *parsing* probabilístico, que utiliza técnicas de aprendizado e cálculos estatísticos baseados em um banco de dados manualmente anotado - conhecido como *corpus* ou *treebank* - para identificar as informações sintáticas corretas, têm se mostrado bastante eficazes, na comparação com outras técnicas, e suficientemente satisfatórias, por conta dessa evolução computacional.

Muitas pesquisas e trabalhos vêm sendo realizados, com foco em vários idiomas, notadamente o inglês [PRO03, CHA97, COL97], entretanto verifica-se uma carência de pesquisas, ferramentas, recursos linguísticos e humanos para tratar computacionalmente a língua portuguesa. Existem alguns trabalhos [WIN06, BIC00, BON03] mas é fato reconhecido pelos pesquisadores que ainda não se atingiu um resultado de nível desejável.

Michael Collins, no final da década de 1990, desenvolveu três modelos de *parsing* probabilístico, sendo os últimos extensões aos anteriores [COL97, COL99]. Estes modelos e o seu *parser* são até hoje referência na área e continuam sendo utilizados. Posteriormente, em 2004, Dan Bikel reimplementou o *parser* de Collins, tornando-o mais parametrizável e extensível [BIK04]. Ambos os *parsers* foram amplamente testados para a língua inglesa apresentando bons resultados. Na atualidade, as tentativas de se construir um *parser* probabilístico para a língua portuguesa não têm sido, até o momento, satisfatórias.

1.2 Objetivos

Este trabalho de conclusão reporta o desenvolvimento de um *parser* para a língua portuguesa utilizando as técnicas estatísticas de processamento de linguagem natural desenvolvidas por Michael Collins, utilizando a ferramenta implementada por Dan Bikel, que tornou possível a parametrização e extensão das suas bibliotecas para possíveis adaptações do código. O trabalho desenvolveu-se e nas seguintes etapas:

- Estudar as técnicas envolvidas no desenvolvimento de um *parser*, em particular os modelos de *parsing* de Collins.
- Dominar o uso da ferramenta de *parsing* de Bikel.
- Fazer um estudo detalhado dos parâmetros de implementação de Bikel, pois a eficácia dos algoritmos depende fundamentalmente dos ajustes desses parâmetros.
- Utilizar a ferramenta de Bikel para construção de um *parser* para a língua portuguesa. O treinamento do *parser* foi feito utilizando-se o *corpus* anotado Floresta Sintática (Projeto Linguateca) [SIN09].
- Desenvolver módulos de pré- e pós-processamento do *corpus*.
- Avaliação dos resultados.

1.3 Organização do texto

Este trabalho está organizado em cinco capítulos. Inicialmente neste primeiro capítulo, introduzimos o tema do trabalho, apresentando a motivação e os objetivos. No segundo capítulo apresentam-se o referencial teórico envolvido e trabalhos anteriores que serviram de referência na comparação de resultados. Logo após no terceiro capítulo apresentamos um aprofundamento teórico das técnicas estudadas e conceitos matemáticos implementadas na ferramenta de *parser* utilizada neste trabalho. No capítulo quatro descrevemos a metodologia utilizada, os métodos de avaliação propriamente ditos, e de que maneira os experimentos foram conduzidos, descrevendo os testes realizados juntamente com seus respectivos resultados e finalmente no quinto capítulo são elaboradas algumas considerações finais e possíveis trabalhos futuros.

2 Referencial Teórico

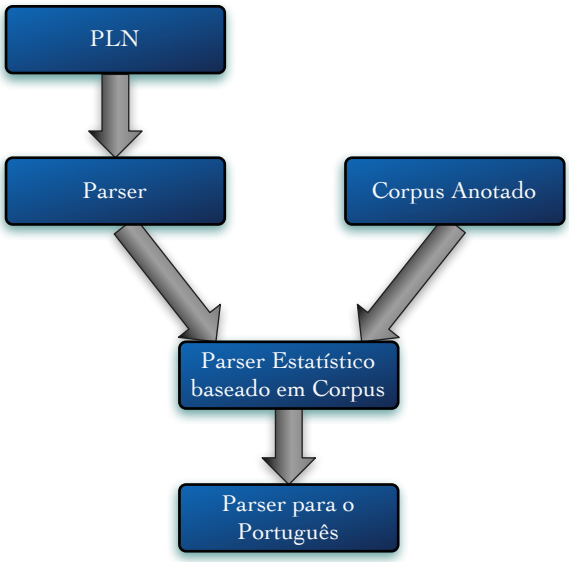

Figura 1: Inserção deste trabalho no contexto do processamento de linguagem natural proposto neste trabalho

A Figura 1 sumariza a inserção deste trabalho na área de processamento de linguagem natural. O *parser* desenvolvido para o português é um *parser* probabilístico que usa estatísticas derivadas de *corpora* anotados sintaticamente para estimar os parâmetros do *parser*. No texto que segue abordamos assuntos que fundamentam nosso trabalho.

2.1 Análise de Sentença

O processo de análise de sentença em linguagem natural é geralmente apresentado na literatura subdividido em vários níveis:

- Análise morfológica

- Análise sintática
- Análise semântica
- Análise pragmática

Este trabalho foca os dois primeiros níveis de análise acima citados. Uma abordagem mais completa de todos os níveis pode ser vista em [ALL95, LIM01] entre outros.

2.1.1 Análise morfológica ou *Part-of-Speech Tagging*

O analisador morfológico identifica palavras ou expressões isoladas em uma sentença, sendo este processo auxiliado por delimitadores (pontuação e espaços em branco). As palavras identificadas são classificadas de acordo com seu tipo de uso ou, em linguagem natural, de acordo com sua categoria gramatical.

Neste contexto, uma instância de uma palavra em uma sentença gramaticalmente válida pode ser substituída por outra do mesmo tipo (exemplo: substantivos, pronomes, verbos, etc.), configurando uma sentença ainda válida do ponto de vista gramatical. Para um mesmo tipo de palavra, existem grupos de regras que caracterizam o comportamento de um subconjunto de vocábulos da linguagem (exemplo: formação do plural de substantivos terminados em "ão", flexões dos verbos regulares terminados em "ar", etc.). Assim, a morfologia trata as palavras quanto à sua estrutura, forma, flexão e classificação, no que se refere a cada um dos tipos de palavras.

Esta fase é frequentemente chamada de *part-of-speech tagging*, pois seu principal resultado é a determinação da categoria sintática das palavras individuais como ocorrem na sentença, também conhecida como *part-of-speech* (POS). Entre essas categorias estão tipicamente as de nome (ou substantivo), verbo, preposição, etc. Outras características importantes podem ser obtidas nesta fase, como gênero (masculino ou feminino), número (singular ou plural), etc. Estas características secundárias, chamadas *features* ou traços, de certa forma estendem a POS. Cada POS tem um conjunto diferenciado de *features* apropriado que depende da aplicação do analisador e das concepções teóricas de quem a define.

Os algoritmos para etiquetagem fundamentam-se em dois modelos mais conhecidos: os baseados em regras e os estocásticos. Os algoritmos baseados em regras, como o nome diz, fazem uso de bases de regras para identificar a categoria de um certo item lexical. Neste caso, novas regras vão sendo integradas à base à medida que novas situações de uso do

item vão sendo encontradas. Os algoritmos baseados em métodos estocásticos costumam resolver as ambiguidades através de um *corpus* de treino, marcado corretamente (muitas vezes através de esforço manual), calculando a probabilidade que uma certa palavra ou item lexical terá de receber uma certa etiqueta em certo contexto. O etiquetador de Eric Brill [BRI95], bastante conhecido na literatura, faz uso de uma combinação desses modelos.

A escolha de um bom *tagset* é fundamental para o sucesso de um *parser*, embora não seja absolutamente claro como fazer este julgamento. Existem vários livros inteiros dedicados a este assunto [ABE03] [SAR04]. Em linhas gerais, um bom *tagset* para um *parser* é aquele que possui uma boa caraterística de "equivalência distribucional"em termos sintáticos; isto é, palavras que ocorrem tipicamente nas mesmas posições nas sentenças têm mesmo POS, enquanto que as que têm características de distribuição diferentes na mesma sentença têm POS diferente. Na abordagem de Collins [COL99] que usamos neste trabalho a etiquetagem é parte integrande no algoritmo de *parser*.

O conjunto de tags (*tagset*) utilizado nesse trabalho é definido em [SIN09].

2.1.2 Análise sintática ou *Parsing*

Através da gramática da linguagem a ser analisada e das informações do analisador morfológico, o analisador sintático procura construir árvores de derivação para cada sentença, mostrando como as palavras estão relacionadas entre si.

Durante a construção da árvore de derivação, é verificada a adequação das seqüências de palavras às regras de construção impostas pela linguagem, no processo de composição das sentenças. Dentre estas regras, pode-se citar a concordância e a regência nominal e/ou verbal, bem como o posicionamento de termos na frase.

A tarefa de um *parser* para a linguagem natural é construir a estrutura sintática da sentença, dividindo-a em subconstituintes de uma forma que reflita, segundo alguma teoria da linguagem, a estrutura composicional de análise da sentença. Esta estrutura é geralmente dada como uma árvore de constituintes, em que os nodos folhas são as POS, com as respectivas palavras, e os nodos internos os conhecidos como sintagmas[1] ou categorias sintáticas de mais alto nível.

Por exemplo, a frase "O João vendeu para Pedro o seu velho computador de mesa",

[1]Sintagma é uma unidade formada por uma ou várias palavras que, juntas, desempenham uma função na sentença

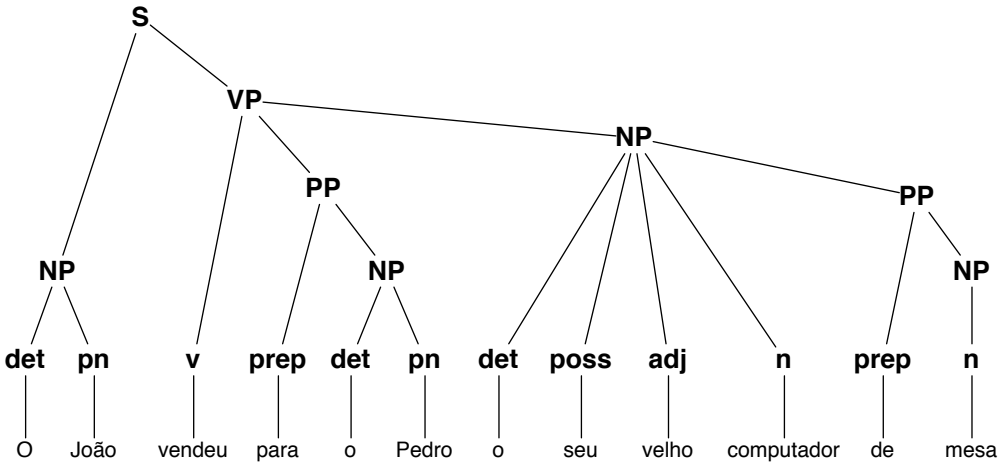

Figura 2: Árvore gramatical da frase *O João vendeu para Pedro o seu velho computador de mesa*

seria anotada gramaticalmente da seguinte forma:

```
(S
   (NP (DET O) (PN João))
   (VP
      (V vendeu)
      (PP (PREP para) (NP (DET o) (PN Pedro)))
      (NP
          (DET o)
          (POSS seu)
          (SDJ velho)
          (N computador)
          (PP (PREP de) (NP (N mesa))))))
```

A Figura 2 ilustra a mesma árvore em formato gráfico.

O conjunto de *TAGS* sintáticos utilizados nas árvores sintáticas do *parser* que construímos para o português está em [SIN09].

2.2 *Corpus* Anotado

Segundo [SAR04], *corpus* é um conjunto de dados linguísticos (pertencentes ao uso oral ou escrito da língua, ou a ambos), sistematizado segundo alguns critérios, suficientemente

extenso em amplitude e profundidade, de maneira que seja representativo da totalidade do uso linguístico ou de algum de seus âmbitos, disposto de tal modo que possa ser processado por computador, com a finalidade de propiciar vários, e úteis, resultados para a descrição e análise.

Corpora anotados sintaticamente, também conhecidos como *treebanks* [ABE03], são - simplificadamente - bancos de dados de sentenças anotadas com informações sintáticas e semânticas que servem como fonte de aprendizado para os sistemas estatísticos. A qualidade e o tamanho do *treebank* influenciam diretamente a qualidade do resultado obtido pelo *parser*. A criação do primeiro *corpus* anotado data do início dos anos 60, e foi desenvolvido inicialmente para o inglês. O objetivo era prover um esquema de anotação mais completo possível, para ser utilizado por esses métodos empíricos, isto tendo em vista processamento dos corpora linguísticos. Para outras finalidades há coisas bem mais antigas.

2.2.1 Extensão do *Corpus*

A extensão e diversidade dos *corpora* são definitivas na qualidade do aprendizado dos *parsers* estatísticos. Conforme [SAR04], pode-se definir três abordagens para a constituição de um *corpus*.

1. Impressionista: baseia-se em constatações derivadas da prática da criação e da exploração de *corpora*, em geral feitas por autoridades da área. Por exemplo, Aston [AST97] menciona patamares que caracterizariam um *corpus* pequeno (20 a 200 mil palavras) e um grande (100 milhões ou mais).

   Leech [LEE91] fala de 1 milhão de palavras com uma taxa usual (*going rate*), sugerindo o patamar mínimo. Outros são mais vagos, como Sinclair [SIN97], que postula que o *corpus* deva ser tão grande quanto a tecnologia permitir para a época, deixando subentender que a extensão de um *corpus* deva variar de acordo com o padrão corrente nos grandes centros de pesquisa, que possuem equipamentos de última geração.

2. Histórica: fundamenta-se na monitoração dos *corpora* eletivamente usados pela comunidade. Por exemplo, Berber Sardinha [SAR04] sugere uma classificação baseada na observação dos *corpora* utilizados, segundo 4 anos de conferências de *corpus*. Tabela de tamanho de *corporas*:

Tabela 2: Tamanho de um *Corpus*.

Tamanho	**Classificação**
Menos de 80 mil	Pequeno
80 mil a 250 mil	Pequeno-médio
250 mil a 1 milhão	Médio
1 milhão a 10 milhões	Médio-grande
10 milhões ou mais	Grande

3. Estatística: fundamenta-se na utilização de teorias estatísticas. Por exemplo, Biber [BIB93] emprega fórmulas matemáticas para identificar quantidades mínimas de palavras, gêneros e textos que se constituíram em uma amostra representativa. Algumas questões que norteiam essa abordagem são:

   (a) Dado um *corpus* preexistente que serve como amostra maior, qual o tamanho mínimo de uma amostra que mantém estáveis as características da amostra maior? Essa é uma perspectiva seguida por Biber [BIB90, BIB93].

   (b) Dada uma fonte externa de referência cuja dimensão é conhecida, qual o tamanho do *corpus* necessário para representar majoritariamente esta fonte? Essa vertente tem sido discutida pela comunidade de linguistas do *corpus*.

   (c) Quanto seria perdido se o *corpus* fosse de um tamanho x? Dados os recursos existentes, quais parâmetros utilizar para avalizar a decisão relativa ao tamanho de *corpus* que pode ser compilado? Uma proposta segundo essa perspectiva ainda não foi formalizada, mas está presente, por exemplo, em [SÁN97, GÓM97], que estima matematicamente a quantidade do vocabulário presente em *corpora* de diversos tamanhos hipotéticos.

2.2.2 *Corpus* para processamento de linguagem natural

2.2.2.1 Corpora da Língua Inglesa

A Tabela 3 lista alguns dos corpora anotados sintaticamente mais relevantes, que são da língua inglesa.

Tabela 3: Corpora da Língua Inglesa.

Corpus	**Lançamento, referência na literatura**	**Palavras**	**Composição**
Bank of English	1987 [1]	459 milhões	inglês britânico
Longman Writen American Corpus	1997	100 milhões	inglês americano, escrito (jornais e livros)
BNC (British National Corpus)	1995	100 milhões	inglês britânico escrito e falado.
LLEC (Longman-Lancaster English Language Corpus)	1988	30 milhões	inglês de vários tipos , escrito e falado.
CHILDES (Child Language Data Exchange)	1990	20 milhões	inglês infantil, falado.
The Penn TreeBank	1989	10 milhões	inglês americano, escrito e falado.
Brown Corpus (Brown University Standart Corpus of Present-day American English)	1964	1 milhão	ingles americano, escrito

[1]Data refere-se ao Birmingham *Corpus*, do qual o Bank of English derivou

Existem outros corpora além dos acima elencados, que possuem um número menor de palavras. Três corpora da lista servem como marcos de referência históricos:

Brown, BNC e Bank of English. O *corpus* Brown é um marco por razões óbvias: é o primeiro. O BNC é de destaque porque foi o primeiro a conter 100 milhões de palavras. Enquanto o Brown e o BNC são *corpus* de amostragem, planejados e fechados.

2.2.2.2 *Penn TreeBank*

O *Penn TreeBank* é um dos principais *corpus* disponíveis, contem aproximadamente 7 milhões de palavras com anotação de POS, 3 milhões de palavras com "esqueleto de *parsing*", mais de 2 milhões de palavras de texto anotado com informação predicado-argumento (*predicate-argument structure*), e 1.6 milhões de palavras de transcrição de conversas. O material anotado possui diferentes origens e gêneros como manuais de computadores da IBM, anotações de enfermeiras, artigos do Wall Street Journal e transcrições de conversas telefônicas, entre outras.

A maioria das anotações do *Penn Treebank* consiste em anotação de POS, estrutura sintática e estrutura predicado-argumento dos textos escritos como os artigos do Wall Street Journal.

O conjunto de *TAGS* sintáticos e de POS (*tagset*) usado no *Penn Treebank*, como muitos outros *corpus*, foi baseado no *Brown Corpus* e é mostrado nas Tabelas 4 (*TAGS* de *Part-of-Speech*) e 5 (*TAGS* de Categorias Sintáticas), contendo 36 tags de POS, 9 tags para pontuação e 17 tags para anotação sintática. Uma descrição detalhada do *tagset* do *Penn Treebank* é encontrada no *website* do projeto *Penn Treebank* em http://www.cis.upenn.edu/ treebank.

Tabela 4: Tags de *Part-of-Speech* do *Penn Treebank*.

CC	Conjunção Coordenativa	PRP	Pronome Pessoal
CD	Numeral	PRP$	Pronome Possessivo
DT	Determinador	RB	Advérbio
EX	Pronome expletivo existencial “there”	RBR	Advérbio, comparativo
FW	Palavra estrangeira	RBS	Advérbio, superlativo
IN	Preposição ou conjunção subordinada	RP	Partícula
JJ	Adjetivo	SYM	Símbolo
JJR	Adjetivo comparativo	TO	Qualquer ocorrência da palavra “TO”
JJS	Adjetivo superlativo	UH	Interjeição
LS	Marcador de item em listas	VB	Verbo infinitivo
MD	Verbo auxiliar ou modal	VBD	Verbo passado
NN	Nome, singular	VBG	Verbo gerúndio ou particípio presente
NNS	Nome, plural	VBN	Verbo particípio passado
NP	Nome próprio singular	VBP	Verbo presente, exceto na terceira pessoa singular
NPS	Nome próprio plural	VBZ	Verbo presente, terceira pessoa singular
PDT	Predeterminador	WDT	Determinador interrogativo
POS	Terminador possessivo	WP	Pronome interrogativo
		WP$	Pronome interrogativo possessivo
		WRB	Adverbio interrogativo
#	Libra sinal	$	Caractere$
.	final	,	vírgula
:	dois pontos	(	Abre parênteses
)	Fecha parênteses	”	aspas dupla abre/fecha
’	aspas simples		

Tabela 5: Tags de Categorias Sintáticas do *Penn Treebank*.

ADJP	Sintagma Adjetivo	ADVP	Sintagma Adverbial
NP	Sintagma nominal	PP	Sintagma preposicional
S	Sintagma de clausula declarativa simples	SBAR	Sintagma de sentença subordinada
SBARQ	Sintagma de sentença interrogativa	SINV	Sintagma declarativo com inversão de sujeito
SQ	Sintagma de questão sim/não e subconstituintes de SBARQ excluindo elemento interrogativo	VP	Sintagma verbal
WHADVP	Sintagma adverbial interrogativo	WHNP	Sintagma nominal interrogativo
WHPP	Sintagma preposicional interrogativo	X	Sintagma de constituinte desconhecido

2.2.2.3 Corpora da Língua Portuguesa

Na língua portuguesa, há alguns corpora eletrônicos de destaque. A Tabela 6 apresenta um pequeno resumo dos *corpus* existentes para o português.

Tabela 6: Corpora da Língua Portuguesa.

Corpus	**Palavras**	**Composição**	**Localização**
Banco de Português	233 milhões	português brasileiro, escrito e falado	PUC/SP
CETEMPublico (*Corpus* de extração de Textos eletrônicos MCT), (Floresta Sintática, Bosque)	220 milhões	jornal português, "público"	Projeto Linguateca
Corpus UNESP/Araraquara/ Usos do português	200 milhões	português brasileiro, escrito	UNESP / Araraquara
CRPC (*Corpus* de referencia do português contemporâneo)	152 milhões	português dos vários países lusófonos, com predominância da variedade européia	CLUL - Centro de lingüística da Universidade de Lisboa.
NILC	35 milhões	português brasileiro escrito	NILC (USP, UFSCAR, UNESP Araraquara)

2.2.2.4 Floresta Sintática (Projeto Linguateca)

Um dos objetivos da Linguateca é melhorar significativamente as condições para o processamento do português, e prover recursos para pesquisa como os repositórios do Floresta Sintática , CETEMPublico e o CETEMFolha .

O CETEMPúblico (*Corpus de Extractos de Textos Eletrônicos MCT/Público*) é um *corpus* de aproximadamente 180 milhões de palavras em português de Portugal, criado por um projeto de processamento computacional do português após a assinatura de um protocolo entre o Ministério da Ciência e Tecnologia português (MCT) e o jornal O Público.

O CETENFolha (*Corpus de Extractos de Textos Eletrônicos NILC/Folha de São Paulo*) é um *corpus* de cerca de 24 milhões de palavras em português brasileiro, criado por um projeto de processamento computacional do português com base nos textos do jornal A Folha de São Paulo que fazem parte do *corpus* NILC/São Carlos, compilado pelo Núcleo Interinstitucional de Lingüística Computacional (NILC).

A Floresta Sintática é um subconjunto dos corpora CETEM Público e CETEM Folha cujas sentanças foram analisadas (morfo)sintaticamente possuindo também indicação das funções sintáticas, explicitando hierarquicamente a informação relativa à estrutura de constituintes, enfim um *treebank*. Foi construído como uma colaboração entre a Linguateca e o projeto VISL. Os textos foram inicialmente anotados (analisados) automaticamente pelo analisador sintático PALAVRAS (Bick 2000) e revisados manualmente por linguistas.

Atualmente, o *corpus* da Floresta Sintá(c)tica tem 4 partes, que diferem quanto ao gênero textual, quanto ao modo (escrito ou falado) e quanto ao grau de revisão lingüística: o Bosque, totalmente revisado por lingüistas; a Selva, parcialmente revisado, a Floresta Virgem e a Amazônia, não revisados. Junto, todo esse material soma cerca de 261 mil frases (6.7 milhões de palavras) sintaticamente analisadas.

Para nosso estudos de desenvolvimento de um *parser* probabilístico para a língua portuguesa e treino da ferramenta desenvolvida por Bikel , será utilizado o Bosque, parte da floresta sintática completamente revisada por linguistas.

O Bosque é composto por frases retiradas dos primeiros 1000 extratos (aproximadamente) dos corpora CETENFolha e CETEMPúblico. Desde 2007, este *corpus* vem passando por um contínuo processo de revisão, em que foram corrigidas algumas pequenas inconsistências e acrescentadas novas etiquetas.

Uma descrição detalhada do tagset da Floresta é encontrado no website do projeto Floresta Sintática em [FLO09]. A versão atual do Bosque é 8.0, de 13 de Outubro de 2008, com 9.437 árvores revistas, correspondendo a 1.962 extratos, 215.420 unidades e aproximadamente 183.619 palavras.

Este é o *corpus* mais correto da Floresta, e por isso o mais aconselhado para pesquisas em que não se prioriza tanto a quantidade, mas sim a precisão dos resultados.

Uma quantificação das etiquetas usadas no Bosque pode ser encontrada no Anexo 4 da Bíblia Florestal [SIN09], e uma extensa documentação das opções lingüísticas tomadas durante o projeto.

2.2.2.4.1 Esquema de anotação

A Floresta Sintática foi o *corpus* utilizado para desenvolver o *parser* aqui descrito, e portanto as árvores geradas pelo *parser* seguem o esquema de anotação da Floresta.

Na Floresta, a cada palavra são associadas etiquetas (ou $TAGS$) principais (de função

e de forma) e secundárias. Estas etiquetas aparecem como FUNÇÃO:forma, onde forma corresponde ao conceito de POS. Em "a menina gulosa", por exemplo, temos:

```
>N:artd        a
H:n            menina
N<:adj         gulosa
```

A anotação ">N" para a palavra "a" de função, indica que a palavra em questão é dependente à esquerda (por isso o sinal ">") de um núcleo nominal (N). Já a forma de "a" é artigo definido. "Menina" é o núcleo do sintagma nominal, por isso a FUNÇÃO é H. Como a palavra em questão é um nome, a forma é n. Por fim, o adjetivo "gulosa" é um dependente (modificador) à direita do nome, e por isso recebe a etiqueta de FUNÇÃO (N<) e a etiqueta de forma adj.

A cada palavra também é associado o seu lema, e informações morfossintáticas (gênero, número, tempo, modo e pessoa para os verbos e, eventualmente, outras etiquetas indicativas de fenômenos como elipse, construções de foco etc.). As etiquetas de POS ou forma estão listadas na Tabela 7. O *TAGS* sintáticos são listados na Tabela 8.

Tabela 7: Tags de *Part-of-Speech* da Floresta Sintática.

Símbolo	**Categoria**
N	nome, substantivo
PROP	nome próprio
ADJ	Adjetivo
N-ADJ	flutuação entre substantivo e adjetivo
V-FIN	Verbo finito
V-INF	Infinitivo
V-PCP	Particípio passado
V-GER	Gerúndio
ART	Artigo
PRON-PERS	pronome pessoal
PRON-DET	pronome determinativo
PRON-INDP	pronome independente (com comportamento semelhante ao nome)
ADV	Advérbio
NUM	Numeral
PRP	Preposição
INTJ	Interjeição
CONJ-S	conjunção subordinativa
CONJ-C	conjunção coordenativa

Tabela 8: Tags Sintáticos da Floresta Sintática

Símbolo	**Categoria**
NP	Sintagma nominal (H: nome or pronome)
ADJP	Sintagma adjetival (H: Adjetivo ou determinante)
ADVP	Sintagma adverbial (H: advérbio)
VP	Sintagma verbal (contém sempre MV e poderá exibir AUX)
PP	Sintagma preposicional (H: preposição)
CU	Sintagma evidenciador de relação de coordenação
SQ	Sequência de funções discursivas; sequência de elementos identificadores do falante, tema, etc. e do discurso propriamente dito
FCL	Oração finita
ICL	Oração infinitiva
ACL	Oração adverbial

2.2.2.5 Projeto Semantic Share

O projeto SemanticShare [BRA09] tem como objetivo o desenvolvimento de corpora anotados para o português da mais recente geração, desenvolver um *PropBank* e um *LogicalFormBank*, dos quais uma parte é paralela a bancos de dados similares que estão a ser produzidos para outros idiomas, em outros projetos.

Estes *corpora* são diferentes materializações de um banco único de enunciados e correspondentes representações gramaticais. Contêm informação morfológica, sintática e semântica integradas, armazenadas internamente em HPSG [BRA08], que podem ser apresentadas em uma ou mais dentre várias visões:

1. Frases
2. Segmentos lexicais
3. Lemas
4. Traços de flexão
5. Etiquetas morfossintáticas (Tabela 10)
6. Entidades nomeadas e unidade multi-palavra
7. Árvores de constituintes (Tabela 9)
8. Árvores de funções e papéis semânticos
9. Formas lógicas

São apoiadas por ferramentas de desenvolvimento de corpora avançadas que asseguram uma extensão fácil das estruturas anotadas quando mais informação de mais dimensões lingüísticas possa ter de ser adicionada em extensões futuras (e.g. tempo, resolução de anáfora, etc), ou quando a cobertura da gramática seja aprofundada.

Estes corpora anotados representam recursos chave para o processamento do Português, incluindo:

- fornecimento de uma base empírica para o estudo lingüístico deste idioma e para o desenvolvimento de ferramentas elaboradas manualmente;

- treinamento de ferramentas de base estatística para o processamento superficial e profundo, incluindo parsers, etiquetadores de papéis semânticos, etc;
- avaliação de ferramentas de processamento;
- apoio à experimentação de abordagens inovadoras em PLN multilingue, incluindo tradução automática estatística ou meta-anotação automática para a web semântica, etc...

Em um dado momento este chegou a ser o *corpus* escolhido para o desenvolvimento do *parser*. No decorrer do desenvolvimento do trabalho mudou-se para o *corpus* Floresta Sintática, que se encontra em estágio mais consolidado.

2.2.2.5.1 Esquema de anotação

Tabela 9: Tags de *Part-of-Speech* do projeto Semantic Share.

Símbolo	**Categoria**
A	Adjetivo
ADV	Adverbio
C	Complementador (que)
CARD	Cardinal
CONJ	Conjunção
D	Determinador
DEM	Pronome demonstrativo
N	Nome
P	Preposição
PNT	Símbolo de pontuação
POSS	Pronome possessivo
PPA	Particípio passado
QNT	Quantificador
V	Verbo

Tabela 10: Tags sintáticos do projeto Semantic Share.

Símbolo	**Categoria**
ADVP	Sintagma adverbial
AP	Sintagma Adjetival
CONJP	Sintagma coordenativo
CP	Sintagma Complementizador
NP	Sintagma nominal
N'	Projeção intermediária entre N e NP
pp	Sintagma preposicional
PPA'	Projeção intermediária entre PPA e PPAP
PPAP	Sintagma de oração Passiva
S	Sintagma de sentença
SNS	Sintagma de sentença sem sujeito
VP	Sintagma verbal

Nota: Alguns sintagma são com informação de extração, por exemplo "S/NP" significa sintagma de sentença com extração de NP (sujeito), VP/NP significa sintagma verbal com extração de NP (objeto), e assim por diante. As ocorrências desse tipo encontradas no *corpus* foram essas: S/ADVP, S/AP, S/PP, SNS/ADVP, SNS/NP, VP/ADVP, VP/AP, VP/PP.

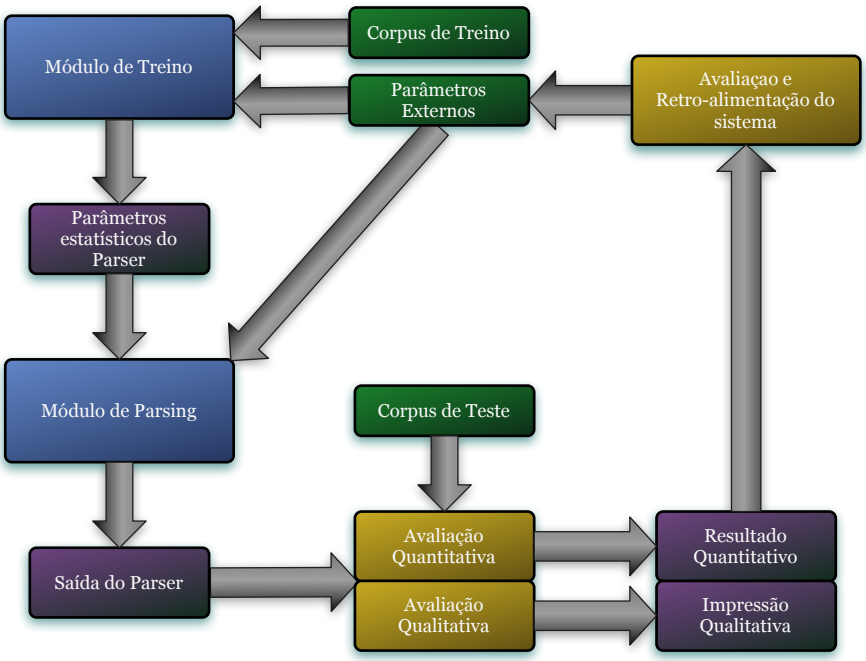

Figura 3: Estágios do processamento de linguagem natural proposto pelo trabalho

2.3 Diagrama geral do processo de desenvolvimento de um *parser* estatístico baseado em *corpus*

Uma visão geral do processo de *parsing* estatístico pode ser observada na Figura 3. No desenvolvimento de um *parser* estatístico baseado em *corpus*, o *corpus* anotado é dividido em 3 partes:

1. Treino: Composto por sentenças que o sistema usa para aprender.

2. Desenvolvimento (ou teste de desenvolvimento): Composto de sentenças utilizadas para avaliar a qualidade do *parser* obtidas a cada passo do desenvolvimento. Como o processo de sintonia do *parser* é incremental e baseado em realimentação do *corpus*, pode haver uma tendência de o *parser* ser ajustado para se adaptar ao conjunto de sentenças submetidas aos testes. Como a análise também é qualitativa, o processo de realimentação para correção do *parser* tem, fatalmente, um aspecto tendencioso. Ou seja, com o tempo, o *corpus* de desenvolvimento perde a isenção para representar resultados confiáveis, pois o desenvolvedor acaba adaptando o *parser* para corrigir especificamente os erros feitos naquelas sentenças.

Isto é conhecido como *overfitting*[2]. Para mitigar este efeito usa-se um terceiro conjunto de sentenças somente analisadas ao final do processo.

3. Teste final: É semelhante ao de desenvolvimento, porém não é usado para sintonia do *parser*. O objetivo é que a avaliação sobre este *corpus* seja isenta, sem efeito de *overfitting*.

O módulo de geração do *parser* tem como entrada os exemplos do *corpus* de treino e gera os parâmetros estatísticos que serão utilizados pelo *parser* para tomar as decisões. Este módulo é parametrizável com informações linguísticas fornecidas pelo desenvolvedor, que guiam a interpretação do *corpus* de treino. Por exemplo, a informação de que, quando um constituinte tem dois nomes seguidos, o núcleo é o da esquerda para o português e é o da direita para o inglês.

O *parser* gerado é composto pelo módulo de *parsing* que recebe as sentenças de entrada e toma as decisões de análise guiado pelos parâmetros estatísticos aprendidos, gerando a sentença analisada.

Cada vez que uma nova versão (ou seja, um novo conjunto de parâmetros estatísticos) do *parser* é gerada , ele é testado e os resultados do teste usados para realimentar o processo. Este teste é feito sobre o *corpus* de desenvolvimento. Os resultados são analisados qualitativa e quantitativamente. Com base nestes valores, pode-se avaliar, por exemplo se a nova versão é melhor ou pior que as anteriores e o que se pode fazer para melhorar.

2.4 *Parsers* para Português

Conforme mencionado anteriormente, existem alguns trabalhos de construção de *parsers* para o português. Dentre eles, o de Eckhard Bick [BIC00], baseado em regras, e portanto difícil de ser expandido ou adaptado; e os de Wing e Baldridge [WIN06] e Bonfante [BON03], que assim como o que aqui descrevemos são estatísticos, baseados no

[2]O conceito de *overfitting* [EVE02] é importante na aprendizagem de máquina. Geralmente, um algoritmo de aprendizagem é treinado utilizando algum conjunto de exemplos de treinamento, ou seja, situações exemplares para que a saída desejada seja acertada. Quem aprende assume como correto o que aprendeu para também ser capaz de acertar a saída correta para outros exemplos, generalizando-se a situações não apresentadas durante o treinamento (baseado em seu viés indutivo). No entanto, especialmente nos casos em que a aprendizagem foi realizada muito tempo ou quando são raros exemplos de treinamento, quem aprende pode adaptar-se a características muito específicas aleatórias dos dados de treinamento, que não têm nenhuma relação causal para a função de destino. Este processo de adaptação também ocorre com relação ao conjunto de sentenças de teste e desenvolvimento se submetidos exaustivamente à análise. Neste processo de *overfitting*, o desempenho nos exemplos de formação continua a aumentar, enquanto o desempenho em conjunto de dados invisível torna-se pior.

modelo de Collins. Entretanto, repetindo o que já mencionamos anteriormente, os resultados até agora obtidos ainda estão distantes dos desejados. A seguir descreve-se os dois trabalhos baseados em [COL99].

2.4.1 Trabalho de Benjamin Wing e Jason Baldridge

Wing e Baldridge apresentaram em [WIN06] seus resultados, do desenvolvimento de um *parser* para o português. Assim como neste trabalho, foi utilizado como *treebank* o Floresta Sintática com a ferramenta de Dan Bikel [BIK02]. Foi desenvolvido um trabalho de exploração de diversos parâmetros possíveis na utilização do *parser*, e em termos de composição do *treebank*, foram feitas alterações nas estruturas e *TAGS* do *treebank*.

Suas métricas de desempenho utilizadas foram o PARSEVAL padrão[3], que também utilizamos nesse trabalho, e análise de dependência não rotulada. Para a análise de dependência no entanto foi necessário um trabalho *ad-hoc* de transformação do *corpus* para criação de um *corpus* com as relações de dependência.

Fazendo mudanças simples nos dados e na parametrização do *parser* de Bikel, incluindo sensibilidade morfológica ao português, mostraram sensível melhora do desempenho atingindo 63,2% de *F-Score* em sua melhor configuração. Em capítulo posterior reportaremos nossos resultados sensivelmente superiores a este.

2.4.1.1 Preparando o material de treino, adaptações no *corpus*

Ao usar o Floresta Sintática para treino da ferramenta, Wing e Baldridge fizeram uma conversão do formato nativo para o formato PennTreebank (PTB). Para este trabalho também tivemos que fazer tal conversão uma vez que o *parser* de Dan Bikel espera como entrada arquivos nesse formato. Foram feitas modificações também quanto à pontuação para que esta seja melhor interpretada pelo parser em formato PTB, por exemplo '.', '?' e '!' foram marcadas como '.'.

Em princípio a informação de núcleo (*head*) dos sintagmas geralmente marcada explicitamente no *corpus* Floresta Sintática, seria de grande ajuda no processo. O PTB não contempla essa informação. Normalmente os analisadores baseados em núcleos dos sintagmas usam regras heurísticas para inferir os núcleos durante a análise. Wing e Baldridge utilizaram-se de marcação disponível no Floresta. No entanto, como nem todos os

[3]As métricas do PARSEVAL para precisão e *recall* de constituíntes rotulados também são descritas na seção 4.2.

sintagma possuem a informação de núcleo, Wing e Baldridge, ainda assim tiveram que utilizar regras heurísticas para resolver as omissões casos em que discordaram da informação constante no *corpus*. Em nosso trabalho optamos por ignorar a marcação fornecida no *corpus* e parametrizar o mecanismo disponibilizado pela ferramenta de Bikel para definir o núcleo dos sintagmas.

Outra mudança feita por Wing e Baldridge no *corpus* foi com relação às cláusulas conjuntivas. Cláusulas conjuntivas no Floresta são normalmente marcadas com a TAG 'CU' (*Coordinating Unit*)(Sintagma evidenciador de relação de coordenação), independente do tipo de constituintes coordenados. Isso faz com que no processo de treino de uma gramática, ocorram erros como confundir coordenação de sintagmas nominais com coordenações sentenciais com frequência. Neste sentido, foram alterados os *TAGS* para refletir mais especificamente as categorias dos sintagmas coordenados.

Em termos de modificação das árvores do *treebank* outras transformações foram feitas, como aumentar cláusulas em NPs para distinguir cláusulas relativas das cláusulas em outras circunstâncias.

Wing e Baldridge reportaram o uso de uma série de filtros para lidar com a riqueza morfológica do português. Por exemplo: Foi usada uma lista de 39 inflexões verbais ou nominais reconhecidas para o português, com cuidado para com falsos positivos e, ao mesmo tempo evitar a propagação de características das palavras. Deste modo temos únicos modos para lidar com várias terminações na terceira pessoa do plural do subjuntivo, mas separando 'ado' e 'ido' para evitar falsos positivos em substantivos como 'caldo' e 'Medo'. Além disso algumas terminações não são listadas como 'o' e 'a', porque elas são muito ambíguas e não são exatamente nominais ou verbais. Modificações no tratamento de plural 's' também foram necessárias pois no português quase sempre o plural é indicado por uma vogal seguida de 's', mas no inglês o plural pode ocorrer com 's' depois de várias consoantes diferentes. Este trabalho é bastante artesanal.

2.4.1.2 Experimentos

Wing e Baldridge trabalharam com três diferentes configurações de dados para experimentar o *parser* proposto, que variam quanto ao esforço na alteração do *treebank* Floresta que foi utilizado como base.

A primeira configuração de testes leva em consideração o *corpus* Floresta sem alteração e as configurações padrão da ferramenta de Bikel para o inglês. A segunda configuração

leva em consideração o *corpus* Floresta sem alteração mas utilizando o pacote de configuração para o português que eles construiram. O terceiro experimento utilizou o Floresta com alterações nas suas anotações e o pacote para o português.

O primeiro representa uma abordagem mais preguiçosa, ou seja, não faça nada que não seja garantir que as árvores geradas possam ser analisadas pelo *parser*. O segundo faz o analisador de reconhecimento da linguagem, aplicando as regras definidas para o português e as configurações ajustadas. O terceiro e último experimento envolve mudar as próprias árvores do *corpus* fornecendo para o *parser* mais informação para o processo de análise.

Para estes experimentos foi criado um conjunto de 7497 sentenças, dessas 1877 para testes e as restantes para treino.

O cálculo *F-score* para o primeiro experimento foi de 38.06%, para o segundo de 63.8% e para o terceiro de 67.1%.

Os desempenhos com relação à configuração básica tiveram grande melhora, simplesmente inserindo o pacote de parâmetros para português ao invés de utilizar o padrão inglês, em particular, as regras de inferência do núcleo dos sintagmas. Ao se adaptar um analisador como o de Bikel para uma nova linguagem vale claramente a pena colocar um mínimo de esforço para se definir um conjunto de regras de inferência reazoável do núcleo dos sintagmas (head-find rules).

2.4.2 *Parsing* probabilístico para o português do Brasil de Andréia Gentil Bonfante

Bonfante em sua tese [BON03], faz uma investigação de métodos estatísticos quando utilizados para analisar sentenças da língua portuguesa do Brasil, implementando o método de modelo gerativo de Michael Collins [COL99]. Como resultado apresenta uma ferramenta para processamento de linguagem natural, PAPO, formada por vários módulos que executam 3 funções básicas: o pré-processamento e a preparação dos dados do conjunto de sentenças usadas no treino; a geração de dois modelos probabilísticos de análise (PAPO I e PAPO II); e um *parser* propriamente dito que usa um dos modelos gerados e produz as árvores sintáticas mais prováveis para uma sentença.

Nesta tese Bonfante não chegou a realizar uma avaliação abrangente e robusta de sua ferramenta, em nenhum de seus modelos. Bonfante preferiu realizar uma investigação qualitativa do desempenho do sistema, com o intuito de identificar problemas mais apa-

rentes que surgissem na análise de um conjunto seleto de sentenças, realizando análise apenas nas sentenças consideradas mais dificeis.

Não é possível avaliar a qualidade dos resultados tanto da versao I quanto da versão II de sua ferramenta, uma vez que Bonfante usou um volume muito pequeno de casos de teste.

Em todos os experimentos realizados na tese de Bonfante, o sistema utiliza como fonte de exemplos de análise, o CENTENFolha, proveniente do Floresta Sintática.

Assim como nos trabalhos semelhantes, Bonfante teve que gerar um módulo de pré-processamento para as sentenças originais, um módulo de filtro de regras, que tem como entrada as sentenças do CENTENFolha e tem como saída regras, para que possam ser usadas na geração dos modelos de sua tese.

Além do filtro de regras foi preciso criar o filtro de núcleos de sentenças, os núcleos foram identificados para cada regra.

2.4.2.1 Experimentos

Para avaliar quantitativamente sua ferramenta Bonfante utilizou 23 sentenças absolutamente inéditas no sentido de não terem sido observadas no treebank utilizado para treino. Antes de serem processadas pelo seu *parser* de acordo com seus modelos de análise, estas sentenças foram anotadas morfossintaticamente com as TAGS do treebank.

Para cada sentença configurou-se a ferramenta para que obtivesse no máximo as dez análises mais prováveis encontradas. Caso sua ferramenta não terminar a análise de uma sentença no prazo máximo de cinco minutos, considera-o sem solução.

Os resultados são apresentados de forma qualitativa e quanto ao tempo de processamento, para processar as 23 sentenças a ferramenta levou em média 47 segundos, seus melhores resultados quanto a precisão são de 79% e *recall* 75%.

Não achamos que seus resultados sejam relevantes para avaliar a performace de sua ferramenta uma vez que apenas 23 sentenças de origem dúbia são um universo pequeno de casos para se avaliar uma ferramenta com tal propósito.

3 Modelos Probabilísticos de Michael Collins

3.1 Gramática Livre de Contexto Probabilística

Para descrever os modelos probabilísticos de *parsing* de Michael Collins antes precisamos entender um pouco de Gramática Livre de Contexto Probabilística (PCFG - *Probabilistic context-free grammar*).

PCFGs são uma extensão das gramáticas livres de contexto, só que existe uma probabilidade associada a cada regra de substituição.

Segundo Bonfante ([BON03], p. 20)

> "O uso de técnicas estatísticas para o aprendizado de gramáticas foi inspirado no sucesso dessas técnicas para o processamento de fala. O modelo proposto em PCFG faz uma suposição de independência que considera a probabilidade de cada regra de substituição independente de todas as outras regras usadas na derivação da sentença. A ordem de derivação não afeta o modelo. As probabilidades atribuídas às regras nas PGFGs, 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."

Segundo Bonfante ([BON03], p. 21)

> " As gramáticas probabilísticas têm muitas vantagens. Sendo elas extensões óbvias das gramáticas livres de contexto, os algoritmos usados para GLCs podem ser transportados para as PCFGs, permitindo que todas as possíveis análises possam ser encontradas num tempo de ordem n^3, em que n é o tamanho da sentença."

A ambiguidade é o maior problema na análise de sentenças. Uma gramática probabilística oferece solução para este problema, escolhendo a interpretação mais provável no momento da análise.

Vamos exemplificar: na frase "vi o homem no monte com os binóculos", supondo a gramática parcial,

1. $S \rightarrow S\ SP^1 \mid SV$
2. $SV \rightarrow SV\ SP^2 \mid V\ SN \mid V\ SN\ SP^3$
3. $SN \rightarrow SN\ SP^4 \mid N \mid N\ SP^5$
4. $SP \rightarrow P\ SN$

1. SP modifica a sentença
2. SP modifica o predicado
3. SP é argumento (objeto indireto) do verbo
4. SP modifica o sintagma nominal
5. SP é complemento nominal

Existe grande quantidade de árvores geradas, pois seria possível relacionar *os binóculos* com *no monte* com vários núcleos de sintagma como argumento ou modificador.

O caso anterior é genuinamente ambíguo (ambiguidade semântica), mas há muitos casos de ambiguidade que se devem apenas à gramática em si, ou seja, leitores percebem apenas uma interpretação.

Portanto, temos um problema quanto a descobrir qual a árvore de análise correta. Como solução poderíamos deixar que as situações de ambigüidade sejam resolvidas pela análise semântica, usar regras de desambiguação manuais ou usar modelos probabilísticos para atribuir probabilidades às diferentes árvores.

Uma gramática livre de contexto probabilística (PCFG) é uma quádrupla (N, T, S_0, R) onde:

- N: Conjunto de símbolos não-terminais
- T: Conjunto de símbolos terminais
- S_0: símbolo não-terminal, designado por símbolo inicial
- R: Conjunto de regras da forma $A \rightarrow \alpha[p]$, onde:
  - A é um símbolo não terminal;
  - α é uma cadeia de zero ou mais símbolos terminais e não terminais;
  - p é um número entre 0 e 1 que representa a probabilidade condicional $P(\alpha|A)$ (ou $P(A \rightarrow \alpha|A)$ ou de forma abreviada $P(A \rightarrow \alpha)$) de uma ocorrência de um dado não terminal em uma derivação ser expandida pela sequência α.

$p = P(\alpha|A)$ = probabilidade de um dado não terminal A ser expandido na expressão α.

Sendo P uma função probabilística, a somatória das probabilidades sobre o universo de eventos deve resultar 1:

$\sum_{\alpha} P(A \rightarrow \alpha) = 1$

Podemos usar uma PCFG para estimar a probabilidade associada a uma dada árvore, o que vai permitir arranjar uma solução para os casos de ambiguidade.

Considerando a hipótese assumida pelas PCFG de que a probabilidade de expansão de cada constituinte é independente do contexto em que aparece na árvore global de análise, a probabilidade associada a cada árvore é o produto das probabilidades das regras usadas na sua derivação. Nas folhas da árvore, usam-se as probabilidades POS $P(\alpha_i|w_i)$, onde α_i é a palavra e w_i é a POS atribuída a palavra.

A estimativa das probabilidades associadas a cada regra pode ser feita usando um *corpus* anotado de sentenças.

Supondo que haja n diferentes regras para expansão de A $A \rightarrow \alpha_i$, i de 1 a n.
Pode-se então estimar $P(A \rightarrow \alpha_i|A)$ como:

$P(A \rightarrow \alpha_i) = \frac{count(A \rightarrow \alpha_i)}{\sum_{j=1}^{n} count(A \rightarrow \alpha_j)} = \frac{count(A \rightarrow \alpha_i)}{count(A)}$, onde count$(A \rightarrow \alpha_i)$ é o número de vezes que uma ocorrência de A é expandida pela regra $A \rightarrow \alpha_i$ no *corpus* e count(A) é o número de vezes que uma ocorrência de A é expandida.

Consegue-se deste modo associar probabilidades às regras e construir uma gramática probabilística, por exemplo:

1. $SV \rightarrow Verbo$[.50]
2. $SV \rightarrow VerboSN$[.45]
3. $SV \rightarrow VerboSNSN$[.05]

3.2 Trabalhos Anteriores, histórico de *Parsing* Probabilístico para PLN

Segundo Bonfante ([BON03], p. 9)

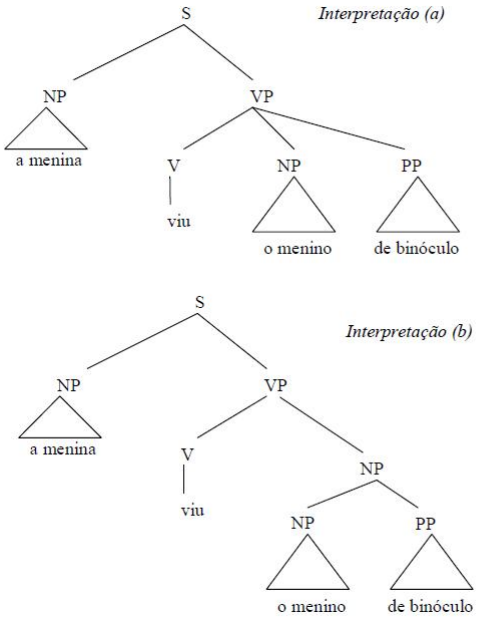

Figura 4: Imagem das árvores da frase *A menina viu o menino de binóculo*

" 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. 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. 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."

Segundo Bonfante ([BON03], p. 10)

"O maior obstáculo encontrado pelos sistemas automáticos de *parsing* (*parsers*) surge quando estes se deparam com sentenças que possuem algum tipo de ambiguidade sintática, ou seja, sentenças com duas ou mais árvores possíveis. Por exemplo, na sentença "A menina viu o menino de binóculo", a atribuição dos *TAGS* sintáticos pode mudar a interpretação que se faz da frase. Duas árvores de representação são 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 conforme ilustrado na Figura 4. 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 acham 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."

Um importante fator influenciador nas pesquisas no campo de PLN foi a disponibilização de grandes *corpora* (*treebanks*) anotados usados para treino dos analisadores probabilísticos.

3.3 Problemas encontrados na Gramática Livre de Contexto Probabilística

Segundo Bonfante [BON03], PCFGs foi o ponto de início natural para as pesquisas em desenvolvimento de *parsers* estatísticos para PLN. Suas propriedades formais foram bem compreendidas, algoritmos eficientes para *parsing* eram bem conhecidos e descritos. Mas após algumas pesquisas descobriu-se que o uso apenas de PCFGs era insuficiente para PLN, e percepção de algumas deficiências na utilização de PCFGs em PLN gerou estudos profundos em três áreas na tentativa se achar métodos apropriados e eficientes para PLN estatística.

1. Desenvolvimento de modelos mais sensíveis quanto as estruturas de linguagem.
2. Desenvolvimento de modelos contendo parâmetros correspondentes às dependências léxicas.
3. Desenvolvimento dos chamados "*history-based models*" ou modelos baseados em históricos.

3.4 Métodos Probabilísticos com aumento de sensibilidade estrutural ou ao contexto

Segundo Collins [COL99], muitas pesquisas foram voltadas ao objetivo de aumentar a sensibilidade ao contexto em uma PCFG, tendo resultados encorajadores. Considerando um modelo baseado em regras, mais uma vez, com mais sensibilidade ao contexto que PCFG.

3.5 Formalismo incluindo dependência lexical

Segundo Collins [COL99], existem pelo menos duas razões para o desenvolvimento de modelos que incluem dependência de parâmetros. Primeiro, pesquisas interessadas em modelagem para reconhecimento da fala imaginavam que enquanto modelos "*trigram*" teriam modelos sintáticos pobres, as probabilidades associadas aos pares ou triplas de palavras era muito úteis quando tinham probabilidades associadas às sentenças na linguagem. Segundo, o mais importante apontado por Michael Collins para seus estudos, pesquisas sugeriram que a dependência de probabilidade seria poderosa para abordar o problema da ambiguidade.

3.6 Modelos de Michael Collins

Segundo Bonfante ([BON03], p. 45)

> "Michael Collins propõe um modelo baseado em dependências lexicais entre bigramas. Este modelo usa informações lexicais para modelar relações núcleo-modificador. Também 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.
>
> Após isso, Collins [COL97] propõe três novos modelos gerativos de *parsing*, que usam uma nova abordagem para melhorar o modelo de bigramas, 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.
>
> Collins define uma probabilidade conjunta P(AS; S) sobre pares árvore-sentença. Ele usa um modelo baseado no histórico de 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, entre outros parâmetros.

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

3.6.1 Modelo 1

Segundo Bonfante ([BON03], p. 53)

"Este modelo apresenta uma proposta de como estender uma Gramática Livre de Contexto Probabilística (PCFG) para uma gramática lexicalizada (que considera itens lexicais). 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.

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

$$Pai(nuc) = E_n(pe_n)...E_1(pe_1)NUC(nuc)D_1(pd_1)...D_m(pd_m)$$

Onde $NUC(nuc)$ representa o núcleo do sintagma, que recebe o item lexical nuc de seu pai Pai; $E_1...E_neD_1...D_m$ são seus sintagmas modificadores, à esquerda e à direita de dentro do núcleo para as extremidades, 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:

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

$$P_{nuc}(NUC|Pai(nuc))\times$$

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

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

Nota-se que a ordem de decomposição é: primeiro núcleo do sintagma, depois os modificadores de dentro para fora (núcleo para extremidades), sendo primeiro os modificadores a esquerda e depois os a direita.

Para um modelo ser Modelo Baseado na História da Análise (MBHA), cada modificador poderia depender de qualquer função Θ dos modificadores anteriores, categoria do núcleo/pai e núcleo.

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

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

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

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

Fazendo a suposição de independência de que os modificadores são gerados independentemente uns dos outros, ou seja, fazendo Θ 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).”

3.6.1.1 Adicionando Distância

Segundo Bonfante ([BON03], p. 55)

“Michael Collins também adiciona distância a esse modelo. Essa adição é importante para capturar preferências relacionadas a modificação à direita (por exemplo, *right pp attachment*) 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 associadas a modificadores à direita) e existência de um verbo entre eles (que permite aprender a preferência pela modificação do verbo mais recente).”

3.6.2 Modelo 2

Segundo Bonfante ([BON03], p. 56)

"Este modelo proposto por Collins, introduz a distinção entre complemento/adjunto. Os complementos 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
   $P_{esq}(E_i(pe_i)|NUC, Pai, nuc, distancia_{esq}(i-1), E-C)$ 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."

3.6.3 Modelo 3

Segundo Bonfante ([BON03], p. 57)

"O modelo 3 é estendido da gramática de estrutura de frase generalizada para possibilitar tratamento de *Wh-movement*[1]. Introduz parâmetros TRACES e Wh-Movement. Por exemplo, na frase "*The store that IBM bought last week*", o modelo usaria as regras para gerá-la:

1. $SN \rightarrow SNSBAR(+gap)$
2. $SBAR(+gap) \rightarrow Wh_{sn}S-C(+gap)$
3. $S(+gap) \rightarrow SN-CSV(+gap)$
4. $SV(+gap) \rightarrow VerboTraceSN$

SBAR é a representação para subcláusula; gap é a indicação de que falta algo naquele espaço."

[1] A modificação do modelo 3 não se mostra relevante e não é contemplada no *parser* do Bikel, nem neste trabalho.

4 Experimentos e resultados obtidos

4.1 Método

Este trabalho possui um forte componente experimental e exploratório. Assim, em termos metodológicos, a cada experiência realizada, os resultados obtidos foram analisados quantitativa e qualitativamente, para orientar as correções nos parâmetros do *parser* ou indicar a necessidade de alterações como: pré- ou pós-processamento dos casos. Nesse sentido, a avaliação quantitativa é um componente importante e foi feita de forma rigorosa. Foram utilizados os métodos de avaliação tradicionais de *precision/recall* [BLA91].

Para trabalhar com o *corpus* no formato de entrada para o *parser* de Bikel foi necessário pré-processamento para eliminar ruídos e criar dados de entrada no formato PTB, mesmos obstáculos encontrados por Baldridge [WIN06] e Bonfante[BON03] em seus trabalhos.

O *corpus* de desenvolvimento, teste e treino utilizado é o Bosque da Floresta Sintática que contém um total de 5.221 sentenças separadas em 522 para desenvolvimento, 522 para teste e 4.177 para treino, nesta ordem.

A presença de ruído nos dados do *corpus* é inevitável pois a maioria das sentenças são anotadas inicialmente de maneira automática, e mesmo após revisão manual algumas inconsistências ainda foram encontradas no decorrer do trabalho. Ruídos são possívelmente provenientes da complexidade nas construções da língua que podem gerar ambiguidade ou estruturas complexas, o que dificulta a análise (em alguns casos dificulta até a análise humana).

O conjunto de experimentos foi dividido em subgrupos que tentam avaliar a melhor configuração com respeito a algum fator relevante, como a utilização de subcategorias das *TAGS* de POS, regras para determinar o núcleo das sentenças e parâmetro de utilização da ferramenta. As configurações que apresentam melhor resultado são mantidas, e os

testes prosseguem tendo como ponto de partida tais configurações.

Para agilizar este processo foi desenvolvido um ambiente de testes em que os experimentos são programados e automatizados, em particular os aspectos de pré-processamento e criação das seleções do corpus quanto a filtros necessários para diferentes experimentos.

4.2 Método de avaliação

As medidas de avaliação do *parser* seguirão a proposta do GEIC/Parseval [BLA91], adaptado conforme [COL97] para ignorar pontuação e não considerar a marcação de POS na avaliação.

Em particular, serão usadas as medidas de *Labeled Precision* (LP) e *Labeled Recall* (LR) e sua média harmônica ($F_{\beta=1}$) ou *F-Score*, descritas abaixo:

$$LP = \frac{número\ de\ constituintes\ corretos\ na\ análise\ proposta}{número\ de\ constituintes\ da\ análise\ proposta}$$

$$LR = \frac{número\ de\ constituintes\ corretos\ na\ análise\ proposta}{número\ de\ constituintes\ do\ treebank\ analisado}$$

$$F_{\beta=1} = \frac{2 * LP * LR}{LP + LR}$$

O termo *Labeled* se refere ao fato de que um constituinte, para contar como corretamente recuperado, deve acertar a extensão correta do texto bem como o rótulo do constituinte.

O procedimento de avaliação compara a saída do *parser* com as análises anotadas no *treebank*; usa a informação de parentização da representação do *treebank* de uma sentença e a análise produzida pra computar três medidas: *crossing brackets, precision e recall*. Neste trabalho não utilizaremos a medida de *crossing brackets*.

Estas métricas são chamadas métricas estruturais, e são baseadas na avaliação dos limites 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 dos limites dos constituintes. Assim dado um nó em uma árvore sintática, a sequência de palavras dominadas por esse nó forma um sintagma, sendo o limite do sintagma representado por um intervalo inteiro *[i,j]*, em que i representa o

índice da primeira palavra e j o da última palavra do sintagma.

Black [BLA91] propõe três medidas estruturais para avaliar sistemas de *parsing*: Labeled Precision, Labeled Recall e Crossing-Brackets. Segundo Lin (1995), esse esquema de avaliação pode ser classificado como em nível de sintagma, ou nível de sentença. As medidas de Labeled Precision e Labeled Recall são computadas da seguinte forma:

Os limites dos sintagmas na resposta (análise produzida pelo *parser*) e no *gold standard* (análise do treebank) são tratados como dois conjuntos (A e K), em que A é a análise obtida do *parser* proposto e K, o *gold standard* do treebank a ser usado na avaliação. O Labeled Recall é definido como a percentagem no *gold standard* que também é encontrada na resposta ((A K)/K). A Labeled Precision é definida como a percentagem de limites no sintagma da resposta que também é encontrada no *gold standard* ((A K)/A).

As medidas propostas no PARSEVAL partem de um 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 no *treebank*.

Exemplo: Considere (1) *gold standard* e (2) análise do *parser*:

```
1.  (S
      (ACL
        (NP
          (ART Um)
          (N arquitecto))
        (PP
          (PRP para)
          (NP
            (NUM 800)
            (N Km2)))))

2.  (S
      (NP
        (ART Um)
        (N arquitecto)
        (PP
          (PRP para)
          (NP
            (NUM 800)
            (N Km2)))))
```

Temos os seguintes limites dos sintagmas:

1. (S 0..4) (ACL 0..4) (NP 0..1) (PP 2..4) (NP 3..4)
2. (S 0..4) (NP 0..4) (PP 2..4) (NP 3..4)

Pontuações em (2)[1]: Labeled Precision = 2/3 = 66%, Labeled Recall = 2/4 = 50%. Essas pontuações têm que ser consideradas juntas para ter significado. No exemplo abaixo, teriamos 100% de Labeled Precision, porém, o Labeled Recall seria apenas (2/5) = 40%:

```
(S
  (NP
    (ART Um)
    (N arquitecto))
  (PRP para)
  (NUM 800)
  (N Km2))
```

4.3 Descrição dos experimentos e resultados

Na execução dos experimentos inicialmente trabalhamos sobre a influência da escolha do núcleo *(Head)* dos sintagmas que é essencial na implementação dos algoritmos dos modelos propostos por Collins [COL99].

O corpus do Bosque já tem anotados muitos dos núcleos (*heads*) dos sintagmas, porém o modelo de Collins precisa de todos os núcleos, e verificou-se que nem todos os sintagmas possuem essa informação, tendo que ser analisada e construída de forma empírica.

A ferramenta de Bikel possibilita a utilização de um módulo para especificação de regras para determinar o núcleo de um sintagma e optou-se por se usar esse módulo e construir incrementalmente, regras para a definição dos núcleos para a língua portuguesa.

4.3.1 Experimentos iniciais

Os primeiros experimentos abordam as regras de definição do núcleo dos sintagmas. Foram experimentadas quatro configurações.

A primeira configuração utilizada foi a *default* para o inglês, com regras de definição do núcleo específicas para o PTB. A segunda configuração de definição do núcleo foi a utilização das regras básicas, ou seja, o núcleo do sintagma é a primeira palavra da direita

[1]No nosso caso, o S foi inserido artificialmente (não constava no treebank, mas é necessário para o formato PTB uma TAG mais externa que domine toda a sentença), não é considerado no cálculo.

para a esquerda. Para terceira configuração alteram-se as regras para que o núcleo do sintagma seja a primeira palavra da esquerda para a direita. Para a quarta e última configuração desta bateria de experimentos foram construídas regras de definição do núcleo dos sintagmas baseados na construção das sentenças para a língua portuguesa.

Tabela 11: *Experimentos para definição dos núcleos dos sintagmas.*

Experimento	***Tagging Accuracy***	**Precisão**	***Recall***	***F-Score***
Standard	92.26%	62.09%	61.94%	62.01%
Right most	92.09%	64.62%	62.53%	63.56%
Left most	92.87%	65.90%	66.75%	66.32%
Regras Português	92.78%	67.31%	66.81%	67.06%

O último conjunto de regras utilizado é o que estamos usando atualmente, bem melhor que o original conforme resultados apresentados.

A estratégia da segunda bateria de testes é avaliar a separação das POS de TAGS em subgrupos. A ideia básica é que tags devem ser distintos quando as categorias têm distribuições sintáticas diferentes. Por outro lado se duas classes têm mesma distribuição ou distribuição próxima, separá-las apenas levará a perda de qualidade quanto à informação sintática constante nas sentenças. Esse tipo de configuração também será observado na evolução dos experimentos.

Nos testes anteriores foram usadas apenas as categorias principais de TAGS do Floresta Sintática. Para avaliar o efeito da distinção de categoria verbal em diferentes distribuições, primeiro foi feita a distinção dos verbos e foram avaliadas as suas subcategorias VFIN, VINF, VPCP, VGER (respectivamente forma finita, infinitiva, particípio e gerúndio). Foi percebida uma melhora considerável nos resultados, portanto mantivemos os verbos com suas subcategorias.

Em seguida o sistema foram avaliados também, as subcategorias da TAG N que possuem distinção no corpus como N e N-ADJ, a TAG CONJ e duas subcategorias (CONJC e CONJS) e, finalmente, a TAG PRON (PRONPERS e PRONINDP). Por último, foi avaliado com todas as subcategorias existentes no corpus.

Tabela 12: *Experimentos para avaliar sub-categorias de POS.*

Experimento	***Tagging Accuracy***	**Precisão**	***Recall***	***F-Score***
Subcategorias de V	92.49%	68.95%	68.32%	68.63%
Subcategorias de V e N	92.45%	69.03%	68.35%	68.69%
Subcategorias de V e CONJ	92.51%	68.99%	68.50%	68.74%
Subcategorias de V e PRON	92.53%	69.53%	68.93%	69.23%
Subcategorias de V, PRON, CONJ e N	92.61%	69.71%	69.19%	69.45%

4.3.2 Experimentos com lematização das palavras

Como terceira bateria de testes utilizamos lematização[2] das palavras do corpus, substituindo os verbos, substantivos, adjetivos pela suas formas canônicas[3]. Essa redução faz com que entradas que significam a mesma coisa tenham a mesma representação de significado.

Para verificar o efeito do uso de lematização, primeiramente foram substituídas as palavras pelos seus lemas embutidos no prório *corpus*, obtendo assim, o *upper bound* de uma possível melhora nos resultados.

Primeiro lematizou-se todas as palavras das sentenças, simplesmente substituindo-as pelos respectivos lemas. Porém, ao lematizar todas as palavras, aumenta-se a ambiguidade léxica, então, como segundo experimento dessa bateria de testes, experimentou-se colocar as POS TAGs concatenadas ao lema, corrigindo assim as suas distribuições e mitigando o problema de ambiguidade.

Sabemos que esses experimentos utilizando lemas do corpus produzem um resultado irreal, pois não podemos depender somente dos lemas disponíveis para as sentenças contidas no corpus, e que ao introduzir um lematizador e POS tagger externo para identificar e concatenar as POS tags com os lemas, iria gerar diferenças entre as TAGS e os lemas do corpus, então resolvemos experimentar a concatenação somente para os verbos, cuja distribuição tem maior impacto na derivação sintática.

Finalmente, foi feito um teste utilizando um POS tagger e lematizador externo, lematizando-se todas as palavras e concatenando a categoria somente aos verbos.

[2]Em Linguística, lematização é o processo de agrupar as diferentes formas flexionadas de uma palavra para que possam ser analisadas como um único item.

[3]Redução à forma canônica consiste em reduzir os verbos para o infinitivo, e os substantivos e adjetivos para a forma masculina singular.

Tabela 13: *Experimentos para avaliar influência de lematização.*

Experimento	***Tagging Accuracy***	**Precisão**	***Recall***	***F-Score***
Lemas do corpus	92.27%	68.99%	68.85%	68.92%
Lemas do corpus com categorias	96.15%	72.10%	71.78%	71.94%
Lemas do corpus com categorias de V	94.40%	71.17%	70.74%	70.95%
Lemas com categorias de V usando TreeTagger	93.53%	70.60%	70.02%	70.31%

Para os experimentos realizados nesse trabalho, foi utilizado o lematizador de palavras TreeTagger[4] [SCH09] e os arquivos de configuração necessárias para o portugês criados por Pablo Gamallo Otero do Grupo de Procesamento de Linguagem Natural da Universidade de Santiago de Compostela [OTE09], que tem a função de receber uma palavra e retornar o lema e sua TAG.

A lematização dos verbos principalmente torna-se importante porque o português é uma língua morfologicamente rica, onde um verbo pode aparecer em dezenas de formas, diferentemente do inglês.

Para verificar que os ajustes feitos não sofreram influência de *overfitting*, efetuamos o último experimento sobre o *corpus* de teste que não inclui as sentenças de treino nem de desenvolvimento. Os resultados foram:

Tabela 14: *Experimentos final com* corpus *de teste.*

Experimento	***Tagging Accuracy***	**Precisão**	***Recall***	***F-Score***
Lemas com categoria de V usando TreeTagger sob corpus de testes	93.35%	69.92%	69.35%	69.63%

4.3.3 Avaliação Quantitativa

Para ajudar a avaliar a performance do *parser* e identificar erros, construímos um analisador que constrói uma matriz de confusão comparando o *egold standard* com o arquivo resultante do *parser*. Na matriz, as linhas representam o *gold standard* e as colunas os arquivos analisados, portanto os valores representam a porcentagem das vezes que uma *tag* do *gold standard* (linha) foi marcada como uma *tag* no arquivo resultante (coluna). Existe um marcador para quando a TAG não foi encontrada no arquivo analisado chamada

[4]O TreeTagger é uma ferramenta para anotação de texto com a POS e informação de lema. Ele foi desenvolvido por Helmut Schmid no *CT Project* do Instituto de Linguística Computacional da Universidade de Stuttgart. O TreeTagger tem sido usado com sucesso o Alemão, Inglês, Francês, Italiano, Holandês, Espanhol, Búlgaro, Russo, Grego, Português, Chinês e os antigos textos em francês e é adaptável a outros idiomas.

de #NF# (*not found*).

Podemos avaliar o experimento *Lemas com categorias de V usando TreeTagger* na Figura 5. Por exemplo, percebe-se que o *parser* teve uma quantidade relativamente baixa de acertos quanto à classificação de orações (ACL, ICL, FCL), e também com conjunções, e a *tag* X foi marcada 9.09% das vezes como FCL, e a *tag* PP não foi encontrada na sentença resultante 29.20% das vezes.

Essa avaliação nos permite identificar quais problemas o *parser* está tendo e tentar resolver problemas mais específicos.

	#NF#	ACL	ADJP	ADVP	CU	FCL	ICL	NP	PP	PRP	S	SQ	VP	X
#NF#	**0.00%**	0.48%	3.81%	2.54%	6.59%	22.31%	6.01%	34.15%	19.42%	0.07%	0.00%	0.17%	3.36%	1.10%
ACL	38.89%	**25.00%**	0.00%	0.00%	1.39%	5.56%	1.39%	12.50%	15.28%	0.00%	0.00%	0.00%	0.00%	0.00%
ADJP	17.42%	0.00%	**73.20%**	0.84%	0.17%	0.00%	0.34%	5.86%	0.34%	0.00%	0.00%	0.00%	1.84%	0.00%
ADVP	14.07%	0.00%	0.66%	**80.88%**	0.22%	0.00%	0.00%	3.74%	0.22%	0.00%	0.00%	0.00%	0.22%	0.00%
CU	76.92%	0.00%	0.00%	0.57%	**15.38%**	4.27%	0.28%	0.85%	1.71%	0.00%	0.00%	0.00%	0.00%	0.00%
FCL	69.08%	0.00%	0.00%	0.10%	1.03%	**26.89%**	0.10%	0.41%	0.72%	0.00%	0.00%	0.00%	0.00%	1.65%
ICL	46.70%	0.00%	0.53%	0.00%	0.53%	0.26%	**47.76%**	2.90%	0.53%	0.00%	0.00%	0.00%	0.26%	0.53%
NP	25.82%	0.03%	1.37%	0.21%	0.75%	0.40%	0.05%	**70.59%**	0.40%	0.00%	0.00%	0.00%	0.37%	0.00%
PP	29.20%	0.10%	0.00%	0.10%	0.41%	0.20%	0.10%	0.46%	**69.43%**	0.00%	0.00%	0.00%	0.00%	0.00%
PRP	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**0.00%**	0.00%	0.00%	0.00%	0.00%
S	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**100.00%**	0.00%	0.00%	0.00%
SQ	28.57%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**71.43%**	0.00%	0.00%
VP	7.93%	0.00%	0.58%	0.07%	0.00%	0.00%	0.00%	0.51%	0.00%	0.00%	0.00%	0.00%	**90.90%**	0.00%
X	63.64%	0.00%	1.82%	0.00%	1.82%	9.09%	1.82%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**21.82%**

Figura 5: Matriz de Confusão: Categorias Sintáticas

4.3.4 Dificuldades

Uma das grandes dificuldades encontradas neste trabalho, foi com relação aos verbos. No português as inflexões verbais são significativamente mais complexas. Os verbos são conjugados em seis pessoas e em dez tempos com representação morfológica diferente, além de em diversas formas não finitas. Além disso muitas terminações de verbos são idênticas aos sufixos flexionais ou derivacionais usados para formar substantivos, isso complica em muito a tarefa de análise morfológica. Dificuldade esta também relatada por Wing e Baldridge em seu trabalho [WIN06].

Apesar de considerarmos importante, não foi dada grande atenção quanto ao tratamento das palavras desconhecidas. Trabalhamos com uma configuração bem próxima da padrão definida por Bikel. Acreditamos que com um ajuste nesse ponto se pode alcançar desempenhos ainda melhores.

5 Considerações finais e trabalhos futuros

Segundo Bonfante [BON03], para que um *parser* tenha robustez e precisão satisfatória, é necessário ter como base uma gramática complexa que consiga abranger grande parte das variedades sintáticas existentes. Métodos de aprendizado de máquina podem ser utilizados para obtenção de gramática a partir de grandes conjuntos de sentenças analisadas (treebanks).

Acreditamos que o uso do Floresta Sintática como treebank alimentador do processo foi um grande facilitador, pois é um *corpus* já estabelecido e bastante revisado, com isso, abrangente o suficiente para o propósito de implementação de um *parser* probabilístico baseado em *corpus* anotado.

Também a escolha de um bom *tagset* é fundamental para o sucesso de um *parser*. Um bom *tagset* para um *parser* é aquele que possui uma boa caraterística de "equivalência distribucional"em termos sintáticos; isto é, palavras que ocorrem tipicamente nas mesmas posições nas sentenças têm mesmo POS, enquanto que as que têm características de distribuição diferentes na mesma sentença têm POS diferente.

Mostramos também que a ferramenta de Dan Bikel pode ser facilmente adaptada para o Português, e que a precisão do *parser* pode ser melhorada significativamente com algumas modificações relativamente simples na configuração dos seu parâmetros.

O *Parser* de Bikel, além de permitir emular os modelos de Collins, também possibilita diferentes parametrizações quanto a algoritmos utilizados, implementação de regras de *head-find* e geração de árvores sintáticas, por exemplo. Ajustes nos parâmetros influenciam fortemente na performance do parser no momento de análise das sentenças, o tempo no processo de *parsing* pode aumentar significativamente, uma vez que pode ser possível parametrizar o quão profunda a análise pode chegar no sentido de geração de árvores sintáticas possíveis geradas para cada sentença submetida para análise, já estes mesmos parâmetros de configurações permitem que o tempo de análise das sentenças seja

menor, ou seja, o processo acelera, menos árvores sintáticas de representação são geradas, e assim a qualidade do resultado pode diminuir.

Percebemos claramente a implementação do modelo 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, que se mostrou evidente durante o processo de ajuste das regras de *head-find* nos primeiros experimentos, à medida que eram incrementalmente melhoradas as regras de escolha do núcleo dos sintagmas, os resultados melhoravam.

Outro fator de grande influência no incremento dos resultados foi a utilização de lematização das palavras do corpus, mais especificamente dos verbos, experimento este não abordado pelos trabalhos anteriores na literatura, acreditamos que a lematização contribuiu no processo de aprendizado do *parser*, melhorando significativamente os resultados.

Muito mais pode ser feito para melhorar o *parser*. Como trabalho futuro, pensamos em uma análise de um ponto de vista estrutural das regras implementadas por Bikel em sua ferramenta, pois percebemos que existem algumas regras, provavelmente de otimização, que estão *hard coded* no código fonte que são dependentes do corpus utilizado originalmente (Penn TreeBank).

Também achamos que uma atenção quanto a ajustes nos parâmetros que o *parser* necessita para tratar as palavras desconhecidas podem levar a melhor ganho na performace.

Nos resultados apresentados em nossa melhor configuração, foi também gerada uma matriz de confusão, com isso, pode-se ainda tentar melhorar os resultados obtidos, identificando os maiores erros e possibilitando focar trabalho de análise especificamente nesses casos.

Referência Bibliográfica

[ABE03] (Abeillé, Anne,Eds.). **Treebanks**: building and using parsed corpora. Paris, France Kluwer Academic Publishers, 2003.

[ALL95] Allen, James. **Natural language understanding**. The Benjamin/Cummings Publishing Company, Inc., 1995.

[AST97] Aston, Guy. **Small and large corpora in language learning**. *PALC Conference*, 1997.

[BIB90] Biber, D. **Methodological issues regarding corpus-based analyses of linguistic variation**. *Literary and Linguistic Computing*, v.5, n.4, p.257, 1990.

[BIB93] Biber, D. **Representativeness in corpus design**. *Literary and linguistic computing*, v.8, n.4, p.243, 1993.

[BIC00] Bick, Eckhard. **The parsing system palavras, automatic grammatical analysis of portuguese in a contraint grammar framework**. Aarhus University Press, 2000.

[BIK02] Bikel, D.M. **Design of a multi-lingual, parallel-processing statistical parsing engine**. San Francisco, CA, USA, 2002. p.178–182.

[BIK04] Bikel, D.M. **Intricacies of Collins' parsing model**. *Computational Linguistics*, MIT Press, v.30, n.4, p.479–511, 2004.

[BLA91] Black, Ezra; Abney, Steven; Gdaniec, C.; Grishman, Ralph; Harrison, P.; Hindle, Don; Ingria, R.; Jelinek, Fred; Klavans, Judith; Liberman, Mark; Marcus, Mitchell; Roukos, Salim; Santorini, Beatrice; Strzalkowski, T. **A procedure for quantitatively comparing the syntactic coverage of english grammars**. In: *Proceedings of the DARPA Speech and Natural Language Workshop*, San Mateo, CA, USA., 1991.

[BON03] Bonfante, Andréia Gentil. **Parsing probabilístico para o português do brasil**. 2003. Tese de Doutorado.

[BRA08] Branco, Antonio; Costa, Francisco. **A computational grammar for deep linguistic processing of portuguese**: lxgram, version a.4.1. 2008.

[BRA09] Branco, António. **Semantic share project**. 2009. (http://semanticshare.di.fc.ul.pt/. (Último acesso em Abril 2009)).

[BRI95] Brill, Eric. **Transformation-based error-driven learning and natural language processing**: a case study in part-of-speech tagging. *Computational Linguistics*, 1995.

[CHA97] Charniak, E. **Statistical techniques for natural language parsing**. *AI magazine*, Menlo Park, California, v.18, n.4, p.33–44, 1997.

[COL97] Collins, M. **Three generative, lexicalised models for statistical parsing**. Morristown, NJ, USA, 1997. p.16–23.

[COL99] Collins, Michael. **Head-driven statistical models for natural language parsing**. 1999. Tese de Doutorado.

[EVE02] Everitt, B.S.; Everitt, BS. **The Cambridge dictionary of statistics**. Cambridge Cambridge University Press Cambridge, 2002.

[FLO09] Floresta Sintática, Bosque. **Projeto floresta sintática, bosque**. 2009. (http://linguateca.dei.uc.pt/Floresta/BibliaFlorestal/anexo1.html (Último acesso em Dezembro 2009)).

[GÓM97] Gómez, Pascual Cantos; Pérez, Aquilino Sánchez. **El ritmo incremental de palabras nuevas en los repertorios de textos**: estudio experimental y comparativo basado en dos corpus lingüísticos equivalentes de cuatro millones de palabras, de las lenguas inglesa y española y en cinco autores de ambas lenguas. Atlantis: Revista de la Asociación Española de Estudios Anglo-Norteamericanos, 1997.

[JUR00] Jurafsky, D.; Martin, J.H.; Kehler, A. **Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition**. MIT Press, 2000.

[LEE91] Leech, Geoffrey. **The state of art in corpus linguistics**. 1991.

[LIM01] Lima, Vera Lúcia Strube de. **Linguística computacional**: princípios e aplicações. *IX Escola da Informática da SBC-Sul*, 2001.

[MAN99] Manning, Christopher D.; Schütze, Hinrich. **Foundations of statistical natural language processing**. The MIT Press, Cambridge, MA, 1999.

[MAR94] Marcus, M.; Kim, G.; Marcinkiewicz, M.A.; MacIntyre, R.; Bies, A.; Ferguson, M.; Katz, K.; Schasberger, B. **The Penn Treebank: annotating predicate argument structure**. 1994. p.119.

[MAR93] Marcus, Mitchell P.; Santorini, Beatrice; Marcinkiewicz, Mary Ann. **Building a large annotated corpus of English**: the Penn Treebank. *Computational Linguistics*, v.19, n.2, p.313–330, 1993.

[OTE09] Otero, Pablo Gamallo. **Grupo de processamento de linguagem natural da universidade de santiago de compostela**. Universidade de Santiago de Compostela, 2009. (http://gramatica.usc.es/ gamallo/ (Último acesso em Dezembro 2009)).

[PRO03] Prolo, Carlos A. **LR parsing for Tree Adjoining Grammars and its application to corpus-based natural language parsing**. June, 2003. Tese de Doutorado.

[SÁN97] Sánchez, A.; Cantos, P. **Predictability of word forms(types) and lemmas in linguistic corpora. A case study based on the analysis of the CUMBRE Corpus: an 8-million-word corpus of contemporary spanish**. *International Journal of Corpus Linguistics*, v.2, n.2, p.259–280, 1997.

[SAR04] Sardinha, Tony Berber. **Linguística de corpus**. Brasil, São Paulo Manole, 2004.

[SCH09] Schmid, Helmut. **Treetagger, tool for annotating text with part-of-speech and lemma information**. University of Stuttgart, 2009. (http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/ (Último acesso em Dezembro 2009)).

[SIN97] Sinclair, J.M. **Corpus evidence in language description**. *Wichmann et al.(eds)*, p.27–39, 1997.

[SIN09] Sintática, Floresta. **Projeto floresta sintática**. 2009. (http://www.linguateca.pt/Floresta/ (Último acesso em Dezembro 2009)).

[WIN06] Wing, B.; Baldridge, J. **Adaptation of data and models for probabilistic parsing of portuguese**. *Lecture Notes in Computer Science*, Itatiaia, Brasil, PROPOR, v.3960, p.140, Maio, 2006.

APÊNDICE A – Ferramenta de *parsing* estatístico de Dan Bikel

Dan Bikel desenvolveu uma ferramenta de *parsing* estatístico extensível que permite diferentes tipos de configuração de modelos estatísticos e gerativos, incluindo emulação dos modelos de *parsing* de Michael Collins [COL99] com performance semelhante. Pode ser facilmente estendida para novos domínios e novas linguagens.

Baseada no modelo 2 de Collins [COL99], permite uma grande gama de parametrizações, implementa e estende o modelo de análise, que inclue análise lexicalizada orientada ao núcleo das sentenças, modelo que incorpora diferentes níveis de informações estruturais, que ja foram descritas anteriormente nesse trabalho.

O analisador permite extenções específicas. Além de usar o pacote em inglês para determinar uma linha de base para análise de precisão, foi criado um pacote par ao português. Este pacote fornece regras para descoberta dos núcleos dos sintágmas, tratamento especial para quando os núcleos são explicitamente marcados, características morfológicas, e algumas opções de ajuste do analisador ao Floresta.

Modelos baseados no núcleo dos sintágmas deve permitir saber quem é o filho do núcleo anterior durante o processo de treino, Esta informação não é codificada no formato PTB, para o inglês o pacote fornece uma série de heurísticas para inferir o núcleo do sintágma.

A ferramenta de Dan Bikel faz uso de parâmetros que definem particularidades do seu funcionamento, sendo alterados dependendo de sua aplicação. Os parâmetros não só permitem alterar alguns comportamentos como também "*plugar*" classes modificadas para serem usadas no lugar das classes padrões, para fazer tratamento especifico do processamento de linguagem definidos algoritmicamente.

As regras de identificação dos núcleos dos sintagmas também são configuráveis e utiliza-se para isto um arquivo de parâmetros de *head-rules*, necessário para a implementação dos modelos 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 como falado anteriormente.

Os parâmetros estão separados em dois arquivos, arquivo de parâmetros e arquivo de regras de *head-rules*, e são utilizados no momento de treinamento do *parser* e no momento de analisar as frases. Os principais parâmetros são:

A.1 Parâmetros de utilização do *parser* de Dan Bikel

O Conjunto de parâmetros abaixo permite com que o *parser* de Bikel Emule o Modelo 2 definito por Michael Collins.

parser.language=english
Especifica o idioma que será analisado.

parser.language.package=danbikel.parser.english
Especifica o pacote referente ao idioma analisado

parser.language.wordFeatures=danbikel.parser.english.SimpleWordFeatures
Especifica o nome da classe que estende WordFeatures no pacote da língua.

parser.downcaseWords=false
Especifica se as palavras devem ser convertidas para minúsculas durante o treino e decodificação.

parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
Especifica qual subclasse de SubcatFactory deve ser utilizada como instanciador.

parser.shifterClass=danbikel.parser.BaseNPAwareShifter
Especifica qual classe deve ser utilizada como *Shifter*.

parser.language.training=portuguese.NPArgThreadTraining
Especifica qual classe que implementa a interface Training deve ser utilizada para efetuar o treinamento.

Parâmetros para classe danbikel.parser.Model

parser.model.precomputeProbabilities=true
A propriedade especifica se deve ou não pré-calcular probabilidades no treino e utilização desses probabilidades pré-computada na decodificação.

parser.model.collinsDeficientEstimation=true
A propriedade especifica se deve ou não fazer estimativa deficiente de probabilidades, como o bug descrito na tese de Michael Collins.

parser.model.prevModMapperClass=danbikel.parser.Collins
Especifica qual classe que implementa a interface NonterminalMapper deve ser utilizada pelo NTMapper para mapear não-terminais que são modificadores previamente gerados de algum não-terminal *head*.

parser.model.doPruning=true
A propriedade especifica se os parâmetros redundantes de cada instancia da classe Model devem ser removidos.

parser.model.pruningThreshold=0.05

A propriedade especifica um fator de quando o *pruning* deve ser realizado.

Parâmetros para Modelos de Bikel, mas é ignorado quando o parâmetro danbikel.model.precomputeProbabilities é ajustado como true

parser.modelCollection.writeCanonicalEvents=true

A propriedade indica ou não se a classe ModelCollection deverá salvar o (grande) *HashMap* contendo as versões canônicas das instancias de *Event* quando é serializado em disco. Ao decodificar usando caches ao invés de probabilidades pré-computadas (ver precomputeProbs), a criação da tabela de eventos canônicos economiza tempo, deixando o decodificador salvar no cache os eventos observados durante o treino ao invés de sempre ter que criar os eventos canônicos, dinamicamente, durante a decodificação.

Parâmetros necessários para o treinamento do parser

parser.training.addGapInfo=false

Propriedade para especificar se Training.addGapInformation(Sexp) adiciona a informação de *gap* ou deixa a formação da árvores intocadas.

parser.training.collinsRelabelHeadChildrenAsArgs=true

A propriedade especifica se o Training.identifyArguments(Sexp) deve re-rotular as os nodo filhos que são *head* como argumentos. Essa rerotulação é desnecessária, uma vez que os *heads* já são inerentemente distintos dos outros filhos, mas é realizada (e possivelmente um bug) no *parser* de Collins e, por isso, está disponível como uma configuração aqui, a fim de simular o mesmo modelo.

parser.training.collinsRepairBaseNPs=true

A propriedade especifica se Training.repairBaseNPs(Sexp) altera a estrutura da árvore ou a deixa intacta.

Parâmetros para classe danbikel.parser.Trainer

parser.trainer.unknownWordThreshold=6

A propriedade especifica o limite de ocorrência em que abaixo as palavras são consideradas desconhecidos pelo treinador.

parser.trainer.countThreshold=1

A propriedade especifica o limite em que abaixo as instancias de TrainerEvent são descartadas pelo treinador.

parser.trainer.reportingInterval=1000

A propriedade especifica o intervalo (em número de períodos) em que o treinador emite relatórios para System.err enquanto treina.

parser.trainer.numPrevMods=1

A propriedade especifica quantos modificadores anteriores a instancia de *Trainer* deve gerar saída.

parser.trainer.numPrevWords=1

A propriedade especifica quantos núcleos (*heads*) dos modificadores anteriores a instancia de *Trainer* deve gerar saída.

parser.trainer.keepAllWords=true

A propriedade especifica se a instancia de *Trainer* deve manter registro de todas as palavras. Normalmente, as palavras abaixo de um limite de ocorrência são mapeados como desconhecidas.

parser.trainer.keepLowFreqTags=true
A propriedade especifica se a instancia de *Trainer* inclui palavras de baixa freqüência no seu mapa de POS.

parser.trainer.collinsSkipWSJSentences=true
A propriedade especifica se algumas frases são ignoradas durante o treino, a fim de emular o *parser* de Michael Collins. Essa opção só deve ser utilizada com o *Penn Treebank Wall Street Journal*.

Parâmetros para a classe danbikel.parser.Decoder

parser.decoder.useLowFreqTags=true
A propriedade especifica se deve utilizar *tags* coletadas de palavras de baixa frequência pelo treinador.

parser.decoder.useCellLimit=false
A propriedade especifica se o decodificador deve impor um limite no número de itens por célula na tabela.

parser.decoder.cellLimit=10
A propriedade especifica o limite para o número de itens que o decodificador terá por célula. Este tipo de poda só irá ocorrer se decoderUseCellLimit está ativado.

parser.decoder.usePruneFactor=true
A propriedade indica se o algoritmo deve podar ou não as arvores geradas dentro de um determinado fator.

parser.decoder.pruneFactor=4
A propriedade para especificar o fator pelo qual o decodificador deverá podar as arvores geradas.

parser.decoder.useCommaConstraint=true
A propriedade especifica se o decodificador deve empregar restrições sobre como vírgulas podem aparecer segundo uma regra de CFG. $Z - - >< ..XY.. >$

parser.decoder.useHeadToParentMap=true
A propriedade para especificar se o decodificador deve usar o mapeamento de nodos *heads* para seus pais, derivados durante o treino.

parser.decoder.useSimpleModNonterminalMap=true
Este é mecanismo pelo qual o decodificador tenta calcular a probabilidade de um não-terminal ser modificador no contexto de um nodo pai e um núcleo.

Exemplo:
Se existe um NP a esquerda de um VP, cujo nodo pai é um S durante o treinamento, então o modificador não-terminal iria conter o mapeamento S, VP, left –¿ NP.

Parâmetros de configuração do pacote, substitua ¡língua¿ pelo nome do pacote
parser.wordfeatures.¡língua¿.useUnderscores=true
Propriedade que define se o símbolo "-" será incluído ou não no vetor de caracteres.

parser.headtable.¡língua¿=data/head-rules.lisp
Define onde estão as regras de determinação dos núcleos.

A.2 Formato do arquivo de parâmetros

```
#     WordNet Parser
```

```
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=english
parser.language.package=danbikel.parser.english
parser.language.wordFeatures=danbikel.parser.english.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=true
parser.training.collinsRepairBaseNPs=true
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=6
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=true
parser.trainer.modNonterminalModelStructureNumber=2
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=true
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=4
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package danbikel.parser.english
#
```

```
parser.wordfeatures.english.useUnderscores=true
parser.headtable.english=data/head-rules.lisp
parser.training.metadata.english=data/training-metadata.lisp
```

A.3 Formato do arquivo de *head-find rules*

O arquivo de *head-rules* contém as regras que determinam a construção das árvores sintáticas necessárias, definindo as sentenças e seu núcleo sintático.

Abaixo segue um exemplo de definição das *head-rules* utilizadas em testes de utilização do *parser*.

Regras de *head-find* para anotação do Bosque

```
(
(NP (l N PROP PRONPERS PRONINDP NADJ NP))
(VP (l VFIN VINF VPCP VGER) (l VP))
(ADJP (l ADJ ADJP) (l PRONDET))
(ADVP (r ADV ADVP))
(CU (r CONJC CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

As regras definidas acima são essenciais para que no momento de treino o *parser* defina exatamente a qual classe pertence as palavras classificando-as de maneira correta sintaticamente.

Por exemplo, a regra (VP (l VFIN VINF VPCP VGER) (l VP)) define que o núcleo deve ser o primeiro nodo filho, da esquerda para direita, que seja um VFIN, VINF, VPCP ou VGER. Caso não exista, então o núcleo será o mesmo do primeiro elemento, da esquerda para a direita, que seja um VP.

Serão experimentadas algumas alterações nos valores de alguns parâmetros acima citados para avaliação dos seus efeitos, e analisaremos os resultados obtidos no capitulo posterior.

APÊNDICE B – Experimentos

B.1 Head-Find Standard

Filtros Aplicados:

- *RemoveConjSubcategories*
- *RemoveNounSubcategories*
- *RemovePronSubcategories*
- *RemoveVerbSubcategories*

Head-Find rules utilizada:

```
((ADJP (l NNS) (l QP) (l NN) (l $) (l ADVP) (l JJ) (l VBN) (l VBG) (l ADJP) (l JJR) (l NP) (l JJS) (l DT) (l FW) (l RBR)
 (ADVP (r RB) (r RBR) (r RBS) (r FW) (r ADVP) (r TO) (r CD) (r JJR) (r JJ) (r IN) (r NP) (r JJS) (r NN))
 (CONJP (r CC) (r RB) (r IN))
 (FRAG (r))
 (INTJ (l))
 (LST (r LS) (r :))
 (NAC (l NN) (l NNS) (l NNP) (l NNPS) (l NP) (l NAC) (l EX) (l $) (l CD) (l QP) (l PRP) (l VBG) (l JJ) (l JJS) (l JJR) (l
 (NP (r NN NNP NNPS NNS NX POS JJR)
     (l NP)
     (r $ ADJP PRN)
     (r CD)
     (r JJ JJS RB QP)
     (r))
 (PP (r IN) (r TO) (r VBG) (r VBN) (r RP) (r FW))
 (PRN (l))
 (PRT (r RP))
 (QP (l $) (l IN) (l NNS) (l NN) (l JJ) (l RB) (l DT) (l CD) (l NCD) (l QP) (l JJR) (l JJS))
 (RRC (r VP) (r NP) (r ADVP) (r ADJP) (r PP))
 (S (l TO) (l IN) (l VP) (l S) (l SBAR) (l ADJP) (l UCP) (l NP))
 (SBAR (l WHNP) (l WHPP) (l WHADVP) (l WHADJP) (l IN) (l DT) (l S) (l SQ) (l SINV) (l SBAR) (l FRAG))
 (SBARQ (l SQ) (l S) (l SINV) (l SBARQ) (l FRAG))
 (SINV (l VBZ) (l VBD) (l VBP) (l VB) (l MD) (l VP) (l S) (l SINV) (l ADJP) (l NP))
```

```
(SQ (l VBZ) (l VBD) (l VBP) (l VB) (l MD) (l VP) (l SQ))
(UCP (r))
(VP (l TO) (l VBD) (l VBN) (l MD) (l VBZ) (l VB) (l VBG) (l VBP) (l VP) (l ADJP) (l NN) (l NNS) (l NP))
(WHADJP (l CC) (l WRB) (l JJ) (l ADJP))
(WHADVP (r CC) (r WRB))
(WHNP (l WDT) (l WP) (l WP$) (l WHADJP) (l WHPP) (l WHNP))
(WHPP (r IN) (r TO) (r FW))
(* (l)))
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
```

```
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=4
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
Number of sentence        =    522
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  63.83
Bracketing Precision      =  63.98
Complete match            =   9.20
Average crossing          =   3.50
No crossing               =  34.48
2 or less crossing        =  58.05
Tagging accuracy          =  92.26

-- len<=40 --
Number of sentence        =    425
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  67.99
Bracketing Precision      =  68.41
Complete match            =  11.29
Average crossing          =   2.00
No crossing               =  42.35
2 or less crossing        =  70.12
Tagging accuracy          =  92.61
No. of matched brackets   = 4481
No. of gold brackets      = 6591
No. of test brackets      = 6550
```

B.2 Head-Find Left Most

Filtros Aplicados:

- *RemoveConjSubcategories*
- *RemoveNounSubcategories*
- *RemovePronSubcategories*
- *RemoveVerbSubcategories*

Head-Find rules utilizada:

```
(
(* (1))
)
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
```

```
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=4
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
Number of sentence        =    522
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  68.40
Bracketing Precision      =  67.58
Complete match            =  10.15
Average crossing          =   3.02
No crossing               =  36.21
2 or less crossing        =  61.11
Tagging accuracy          =  92.87

-- len<=40 --
Number of sentence        =    425
```

```
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  72.11
Bracketing Precision      =  71.51
Complete match            =  12.47
Average crossing          =   1.73
No crossing               =  44.00
2 or less crossing        =  72.94
Tagging accuracy          =  93.02
No. of matched brackets   = 4753
No. of gold brackets      = 6591
No. of test brackets      = 6647
```

B.3 Head-Find Right Most

Filtros Aplicados:

- *RemoveConjSubcategories*
- *RemoveNounSubcategories*
- *RemovePronSubcategories*
- *RemoveVerbSubcategories*

Head-Find rules utilizada:

```
(
(* (r))
)
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
```

```
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=4
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
```

```
Number of sentence        =    522
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  64.30
Bracketing Precision      =  66.44
Complete match            =  11.30
Average crossing          =   3.22
No crossing               =  35.44
2 or less crossing        =  56.51
Tagging accuracy          =  92.09

-- len<=40 --
Number of sentence        =    425
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  68.53
Bracketing Precision      =  70.44
Complete match            =  13.88
Average crossing          =   1.98
No crossing               =  43.53
2 or less crossing        =  68.24
Tagging accuracy          =  92.39
No. of matched brackets   = 4517
No. of gold brackets      = 6591
No. of test brackets      = 6413
```

B.4 Head-Find Português

Filtros Aplicados:

- *RemoveConjSubcategories*
- *RemoveNounSubcategories*
- *RemovePronSubcategories*
- *RemoveVerbSubcategories*

Head-Find rules utilizada:

```
(
(NP (l N PROP PRON_PERS PRON_INDP PRON NADJ NP))
(VP (l V_FIN V_INF V_PCP V_GER V) (l VP))
(ADJP (l ADJ ADJP) (l PRON_DET PRON))
(ADVP (r ADV ADVP))
(CU (r CONJ_C CONJ CU , ;))
```

```
(X (1 VP))
(PP (1 PRP PP))
(FCL (1 VP) (1 NP))
(ICL (1 VP) (1 NP))
(ACL (1 VP) (1 NP))
(* (1))
)
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
```

```
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=4
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
Number of sentence        =    522
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  68.46
Bracketing Precision      =  68.95
Complete match            =  13.60
Average crossing          =   3.00
No crossing               =  36.40
2 or less crossing        =  61.49
Tagging accuracy          =  92.78

-- len<=40 --
Number of sentence        =    425
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  72.95
Bracketing Precision      =  73.62
Complete match            =  16.71
Average crossing          =   1.77
No crossing               =  44.71
2 or less crossing        =  74.35
Tagging accuracy          =  92.94
No. of matched brackets   = 4808
No. of gold brackets      = 6591
No. of test brackets      = 6531
```

B.5 Com subcategorias de V

Filtros Aplicados:

- *RemoveConjSubcategories*
- *RemoveNounSubcategories*
- *RemovePronSubcategories*

Head-Find rules utilizada:

```
(
(NP (l N PROP PRON_PERS PRON_INDP PRON NADJ NP))
(VP (l V_FIN V_INF V_PCP V_GER V) (l VP))
(ADJP (l ADJ ADJP) (l PRON_DET PRON))
(ADVP (r ADV ADVP))
(CU (r CONJ_C CONJ CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
```

```
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=4
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
Number of sentence        =    522
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  69.90
Bracketing Precision      =  70.51
```

```
Complete match            =  14.37
Average crossing          =   2.83
No crossing               =  36.97
2 or less crossing        =  62.45
Tagging accuracy          =  92.49

-- len<=40 --
Number of sentence        =    425
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  74.28
Bracketing Precision      =  74.83
Complete match            =  17.65
Average crossing          =   1.67
No crossing               =  45.41
2 or less crossing        =  74.82
Tagging accuracy          =  92.71
No. of matched brackets   = 4896
No. of gold brackets      = 6591
No. of test brackets      = 6543
```

B.6 Com categorias de V, e N

Filtros Aplicados:

- *RemoveConjSubcategories*
- *RemovePronSubcategories*

Head-Find rules utilizada:

```
(
(NP (l N PROP PRON_PERS PRON_INDP PRON NADJ NP))
(VP (l V_FIN V_INF V_PCP V_GER V) (l VP))
(ADJP (l ADJ ADJP) (l PRON_DET PRON))
(ADVP (r ADV ADVP))
(CU (r CONJ_C CONJ CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=4
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
```

```
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
Number of sentence        =    522
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  69.93
Bracketing Precision      =  70.58
Complete match            =  14.56
Average crossing          =   2.82
No crossing               =  36.97
2 or less crossing        =  63.03
Tagging accuracy          =  92.45

-- len<=40 --
Number of sentence        =    425
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  74.39
Bracketing Precision      =  75.00
Complete match            =  17.88
Average crossing          =   1.65
No crossing               =  45.41
2 or less crossing        =  75.53
Tagging accuracy          =  92.72
No. of matched brackets   = 4903
No. of gold brackets      = 6591
No. of test brackets      = 6537
```

B.7 Com categorias de V e CONJ

Filtros Aplicados:

- *RemoveNounSubcategories*
- *RemovePronSubcategories*

Head-Find rules utilizada:

```
(
(NP (l N PROP PRON_PERS PRON_INDP PRON NADJ NP))
(VP (l V_FIN V_INF V_PCP V_GER V) (l VP))
(ADJP (l ADJ ADJP) (l PRON_DET PRON))
(ADVP (r ADV ADVP))
(CU (r CONJ_C CONJ CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
```

```
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=4
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
Number of sentence        =    522
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  70.07
Bracketing Precision      =  70.54
Complete match            =  14.37
Average crossing          =   2.87
No crossing               =  36.97
2 or less crossing        =  62.07
Tagging accuracy          =  92.51

-- len<=40 --
Number of sentence        =    425
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  74.43
Bracketing Precision      =  75.04
Complete match            =  17.65
Average crossing          =   1.65
No crossing               =  45.41
2 or less crossing        =  74.82
```

```
Tagging accuracy          =  92.71
No. of matched brackets   = 4906
No. of gold brackets      = 6591
No. of test brackets      = 6538
```

B.8 Com categorias de V e PRON

Filtros Aplicados:

- *RemoveConjSubcategories*
- *RemoveNounSubcategories*

Head-Find rules utilizada:

```
(
(NP (l N PROP PRON_PERS PRON_INDP PRON NADJ NP))
(VP (l V_FIN V_INF V_PCP V_GER V) (l VP))
(ADJP (l ADJ ADJP) (l PRON_DET PRON))
(ADVP (r ADV ADVP))
(CU (r CONJ_C CONJ CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
```

```
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=4
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
Number of sentence        =    522
Number of Error sentence  =      0
```

```
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  70.47
Bracketing Precision      =  71.06
Complete match            =  14.37
Average crossing          =   2.75
No crossing               =  37.36
2 or less crossing        =  62.64
Tagging accuracy          =  92.53

-- len<=40 --
Number of sentence        =    425
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  74.97
Bracketing Precision      =  75.61
Complete match            =  17.65
Average crossing          =   1.57
No crossing               =  45.88
2 or less crossing        =  75.53
Tagging accuracy          =  92.83
No. of matched brackets   = 4941
No. of gold brackets      = 6591
No. of test brackets      = 6535
```

B.9 Com subcategorias de V, PRON, CONJ e N

Filtros Aplicados:

Nenhum filtro foi aplicado.

Head-Find rules utilizada:

```
(
(NP (l N PROP PRON_PERS PRON_INDP PRON NADJ NP))
(VP (l V_FIN V_INF V_PCP V_GER V) (l VP))
(ADJP (l ADJ ADJP) (l PRON_DET PRON))
(ADVP (r ADV ADVP))
(CU (r CONJ_C CONJ CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=4
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
```

```
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
Number of sentence        =    522
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  70.72
Bracketing Precision      =  71.23
Complete match            =  14.94
Average crossing          =   2.76
No crossing               =  36.78
2 or less crossing        =  61.88
Tagging accuracy          =  92.61

-- len<=40 --
Number of sentence        =    425
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  75.03
Bracketing Precision      =  75.74
Complete match            =  18.35
Average crossing          =   1.58
No crossing               =  45.18
2 or less crossing        =  75.06
Tagging accuracy          =  92.88
No. of matched brackets   = 4945
No. of gold brackets      = 6591
No. of test brackets      = 6529
```

B.10 Com lemas do corpus

Filtros Aplicados:

- *LematizeAll*

Head-Find rules utilizada:

```
(
(NP (l N PROP PRON_PERS PRON_INDP PRON NADJ NP))
(VP (l V_FIN V_INF V_PCP V_GER V) (l VP))
(ADJP (l ADJ ADJP) (l PRON_DET PRON))
(ADVP (r ADV ADVP))
(CU (r CONJ_C CONJ CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
```

```
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=10
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
Number of sentence        =    522
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  70.40
Bracketing Precision      =  70.54
Complete match            =  15.52
Average crossing          =   3.00
No crossing               =  38.31
2 or less crossing        =  59.20
Tagging accuracy          =  92.27

-- len<=40 --
Number of sentence        =    425
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  74.92
Bracketing Precision      =  75.29
Complete match            =  19.06
Average crossing          =   1.71
No crossing               =  47.06
```

```
2 or less crossing        =  72.24
Tagging accuracy          =  92.40
No. of matched brackets   = 4938
No. of gold brackets      = 6591
No. of test brackets      = 6559
```

B.11 Com lemas do corpus com categorias concatenada

Filtros Aplicados:

- *LematizeAllAndAppendCategory*

Head-Find rules utilizada:

```
(
(NP (l N PROP PRON_PERS PRON_INDP PRON NADJ NP))
(VP (l V_FIN V_INF V_PCP V_GER V) (l VP))
(ADJP (l ADJ ADJP) (l PRON_DET PRON))
(ADVP (r ADV ADVP))
(CU (r CONJ_C CONJ CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
```

```
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=4
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
Number of sentence        =    522
Number of Error sentence  =      0
```

```
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  73.18
Bracketing Precision      =  73.50
Complete match            =  18.58
Average crossing          =   2.76
No crossing               =  40.80
2 or less crossing        =  62.84
Tagging accuracy          =  96.15

-- len<=40 --
Number of sentence        =    425
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  77.56
Bracketing Precision      =  78.19
Complete match            =  22.82
Average crossing          =   1.57
No crossing               =  49.88
2 or less crossing        =  75.29
Tagging accuracy          =  96.32
No. of matched brackets   = 5112
No. of gold brackets      = 6591
No. of test brackets      = 6538
```

B.12 Com lemas do corpus com categorias de V concatenadas

Filtros Aplicados:

- *LematizeAllAndAppendCategoryOnlyToVerb*

Head-Find rules utilizada:

```
(
(NP (l N PROP PRON_PERS PRON_INDP PRON NADJ NP))
(VP (l V_FIN V_INF V_PCP V_GER V) (l VP))
(ADJP (l ADJ ADJP) (l PRON_DET PRON))
(ADVP (r ADV ADVP))
(CU (r CONJ_C CONJ CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=4
```

```
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
Number of sentence        =    522
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  72.20
Bracketing Precision      =  72.62
Complete match            =  17.05
Average crossing          =   2.78
No crossing               =  40.23
2 or less crossing        =  61.69
Tagging accuracy          =  94.40

-- len<=40 --
Number of sentence        =    425
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  76.63
Bracketing Precision      =  77.22
Complete match            =  20.94
Average crossing          =   1.57
No crossing               =  49.18
2 or less crossing        =  74.82
Tagging accuracy          =  94.47
No. of matched brackets   = 5051
No. of gold brackets      = 6591
No. of test brackets      = 6541
```

B.13 Com lemas do TreeTagger e categorias de V concatenadas

Filtros Aplicados:

- *LematizeAllAndAppendCategoryOnlyToVerbUsingTreeTagger*

Head-Find rules utilizada:

```
(
(NP (l N PROP PRON_PERS PRON_INDP PRON NADJ NP))
(VP (l V_FIN V_INF V_PCP V_GER V) (l VP))
(ADJP (l ADJ ADJP) (l PRON_DET PRON))
(ADVP (r ADV ADVP))
(CU (r CONJ_C CONJ CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Configurações do parser:

```
#     WordNet Parser
#    Settings to emulate Mike Collins' 1997 Model 2
#
parser.language=portuguese
parser.language.package=portuguese
parser.language.wordFeatures=portuguese.SimpleWordFeatures
parser.downcaseWords=false
parser.subcatFactoryClass=danbikel.parser.SubcatBagFactory
parser.shifterClass=danbikel.parser.BaseNPAwareShifter
parser.language.training=portuguese.NPArgThreadTraining
#
# settings for danbikel.parser.Model
parser.model.precomputeProbabilities=true
parser.model.collinsDeficientEstimation=true
parser.model.doPruning=true
parser.model.pruningThreshold=0.05
parser.model.prevModMapperClass=danbikel.parser.Collins
#
# settings for danbikel.parser.ModelCollection
#    the following property is ignored when
#    danbikel.model.precomputeProbabilities is true
parser.modelCollection.writeCanonicalEvents=true
#
# settings for danbikel.parser.Training
parser.training.addGapInfo=false
parser.training.collinsRelabelHeadChildrenAsArgs=false
parser.training.collinsRepairBaseNPs=false
#
# settings for danbikel.parser.Trainer
parser.trainer.unknownWordThreshold=5
parser.trainer.countThreshold=1
parser.trainer.reportingInterval=1000
parser.trainer.numPrevMods=1
```

```
parser.trainer.numPrevWords=1
parser.trainer.keepAllWords=true
parser.trainer.keepLowFreqTags=true
parser.trainer.globalModelStructureNumber=1
parser.trainer.collinsSkipWSJSentences=false
parser.trainer.modNonterminalModelStructureNumber=3
parser.trainer.modWordModelStructureNumber=2
#
# settings for danbikel.parser.CKYChart
parser.chart.itemClass=danbikel.parser.CKYItem$MappedPrevModBaseNPAware
parser.chart.collinsNPPruneHack=false
#
# settings for danbikel.parser.Decoder
parser.decoder.useLowFreqTags=true
parser.decoder.useCellLimit=false
parser.decoder.cellLimit=10
parser.decoder.usePruneFactor=true
parser.decoder.pruneFactor=3
parser.decoder.maxPruneFactor=4
parser.decoder.useCommaConstraint=true
parser.decoder.useHeadToParentMap=true
parser.decoder.useSimpleModNonterminalMap=true
#
#
# settings specific to language package portuguese
#
parser.wordfeatures.portuguese.useUnderscores=true
parser.headtable.portuguese=head-rules.lisp
parser.training.metadata.portuguese=training-metadata.lisp
```

Resultados obtidos:

```
-- All --
Number of sentence        =    522
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    522
Bracketing Recall         =  71.51
Bracketing Precision      =  72.08
Complete match            =  16.48
Average crossing          =   2.83
No crossing               =  38.89
2 or less crossing        =  61.49
Tagging accuracy          =  93.53

-- len<=40 --
Number of sentence        =    425
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =    425
Bracketing Recall         =  75.69
Bracketing Precision      =  76.48
Complete match            =  20.24
```

```
Average crossing          =   1.62
No crossing               =  47.76
2 or less crossing        =  74.59
Tagging accuracy          =  93.70
No. of matched brackets   = 4989
No. of gold brackets      = 6591
No. of test brackets      = 6523
```

APÊNDICE C – Ferramenta Desenvolvida

C.1 Visão Geral

Durante o desenvolvimento desse trabalho, construímos uma ferramenta para auxiliar na construção e organização de experimentos que possibilita criar filtros plugáveis de pré-processamento e programar experimentos a serem executados em sequência, mantendo os resultados de cada experimento armazenado para a posteridade.

Nesse apêndice, vamos explicar o seu funcionamento para que outras pessoas possam utiliza-las, também.

A ferramenta foi desenvolvida utilizando a linguagem de programção Ruby na versão 1.9 e está disponível com o código aberto no seguinte repositório: `http://github.com/divoxx/porser`.

C.2 Instalação

Para poder utilizar a aplicação é necessário ter Ruby >= 1.9 instalado, e o rubygem Rake. A linguagem Ruby pode ser baixada em `http://ruby-lang.org/` e o Rake pode ser instalado pelo seguinte comando: `sudo gem install rake`.

C.3 Utilizando a ferramenta

A ferramenta possui uma organização de diretórios pré-definida, permitindo a automatização de sua execução, da seguinte forma:

```
+ corpus              # Dados relacionados a corpus.
|-+ experiments       # Os experimentos que vão sendo criados e documentados.
|-+ originals         # Corpus originais.
|-+ pre-processed     # Corpus pre-processados, por exemplo o Bosque transformato no formato PTB.
```

```
|-+ selection         # As seleções bases dos experimentos, por exemplo: desenvolvimento, teste e treino.
+ coverage            # Informações de cobertura de testes.
+ ext                 # Pacote java que extende o parser do Bikel.
|-+ build             # Pacote compilado que é automaticamente incluido no classpath durante execução do parser.
|-+ src               # Código fonte do pacote java
+ lib                 # Código em ruby da ferramente
|-+ porser            # Código principal da ferramenta
  |-+ cli             # Classes de auxilio em interação com usuário usando Command Line Interface.
  |-+ corpus          # Classes de parsing, e representação de informação de corpus, como sentenças.
  |-+ experiment.rb   # Representação de um experimento, possui lógica de execução de todas etapas do experimento.
  |-+ filters         # Filtros plugáveis de pré-processamento de corpus, tal como lematização.
  |-+ performance     # Classes de análise de performance, como construção de matriz de confusão.
|-+ porser.rb         # Arquivo de boot.
|-+ tasks             # Rake tasks para uso do parser.
|-+ templates         # Alguns templates para geração de informação, como doc latex dos experimentos.
Rakefile              # Arquivo rake (http://rake.rubyforge.org)
+ samples             # Arquivos exemplos de configuração do parser.
+ scripts             # Scripts de utilidade.
|-+ bosque_cleanup.rb # Script para transformar o Bosque no formato PTB.
+ spec                # Specs (testes) de desenvolvimento.
+ vendor              # Outras aplicações e bibliotecas utilizadas, incluindo o próprio parser do Bikel.
+ volume              # Volume, esse próprio trabalho.
```

Para facilitar a utilização da ferramenta, utilizamos o Rake, que nos permite definir tarefas que podem ser executas pelo usuário, semelhante a ferramenta make e ant. Para executar uma tarefa basta executar: `rake <nome_da_tarefa>`, sendo as tarefas disponíveis:

```
$ rake -T
rake build                          # Compile and build the java extension
rake clobber_spec                   # Remove rcov products for spec
rake experiments:create             # Create a new experiment
rake experiments:doc                # Generate the LaTeX documentation for the experiment
rake experiments:parse              # Run the parsing process for an experiment
rake experiments:pretty_print[what] # Prettyprint
rake experiments:rebuild            # Rebuild a corpus experiment
rake experiments:run                # Run the whole process for many experiments
rake experiments:score              # Run the scoring process for an experiment
rake experiments:score_confusion    # Run the quantitative scoring process for an experiment
rake experiments:train              # Run the training process for an experiment
rake spec                           # Run all examples
```

O fluxo e utilização da ferramenta é, basicamente, a criação de um ou vários experimentos, e posteriormente a execução. Por exemplo, ao criar um experimento o usuário poderá informar quais filtros quer aplicar no pré-processamento, sendo a ordem importante, e se desejar copiar os arquivos exemplos de configuração para a pasta do experimento:

```
$ rake experiments:create
```

```
(in /Users/rodrigo/Documents/Puc/TC/porser)
Available filters:
  [0] None
  [1] lematize_all.rb
  [2] lematize_all_and_append_category.rb
  [3] lematize_all_and_append_category_only_to_verb.rb
  [4] lematize_all_and_append_category_only_to_verb_using_tree_tagger.rb
  [5] lematize_only_verb.rb
  [6] lematize_only_verb_and_append_category.rb
  [7] remove_conj_subcategories.rb
  [8] remove_noun_subcategories.rb
  [9] remove_pron_subcategories.rb
  [10] remove_verb_subcategories.rb
Choose the filters to apply (Multiple options allowed, separated by comma, or * for all): 7,8,9,4

Experiment created at corpus/experiments/2009.11.20_16.28.36-remove_conj_subcategories-
  remove_noun_subcategories-remove_pron_subcategories-lematize_all_and_append_category_only_to_verb_using_tree_tagger

Copy config files from sample/ to the experiment directory? y
```

Então, uma nova pasta será criada dentro de `corpus/experiments` usando o corpus base definido em `corpus/selection`. Após isso, pode-se manualmente alterar os arquivos na pasta do experimento, como por exemplo a configuração do parser. Quando tudo estiver pronto, pode-se criar outro experimento, sucessivamente até que se deseje executa-los:

```
$ rake experiments:run
(in /Users/rodrigo/Documents/Puc/TC/porser)
Available experiments:
  [0] 2009.11.20_16.28.36-remove_conj_subcategories-remove_noun_subcategories-
    remove_pron_subcategories-lematize_all_and_append_category_only_to_verb_using_tree_tagger
Choose the experiment to train (Multiple options allowed, separated by comma, or * for all): *
```

Os experimentos vão ser executados em série, incluindo treino, *parsing* do corpus de desenvolvimento e análise qualitativa e quantitativa do mesmo. Por exemplo, esse experimento que acabamos de criar teria os seguinter arquivos:

```
corpus.dev.gold.txt               # Corpus de desenvolvimento anotado
corpus.dev.parseable.txt          # Corpus de desenvolvimento sem anotação
corpus.dev.parseable.txt.parsed   # Corpus de desenvolvimento análisado pelo parser
corpus.dev.parseable.txt.scorable # Corpus de desenvolvimento analisado corrigido para análise de performance
corpus.test.gold.txt              # Corpus de teste anotado
corpus.test.parseable.txt         # Corpus de teste sem anotação
corpus.train.gold.txt             # Corpus de treino anotado
corpus.train.parseable.txt        # Corpus de treino sem anotação
document.dev.tex                  # Documentação tex incluindo parametros e resultados
head-rules.lisp                   # Regras de head finding
log.parse.dev.txt                 # Log do processo de parsing
```

```
log.score.dev.txt                # Log de processo de scoring
log.train.train.txt              # Log do proceso de treino
objects.gz                       # Arquivo contendo as probabilidades e informações derivadas do treino
observed.gz                      # Arquivo de observações derivado do treino
score.dev.txt                    # Análise qualitativa precision/recall
score_confusion.dev.txt          # Matriz de confusão, analise quantitativa.
settings.properties              # Configurações e paramêtros pro parser
training-metadata.lisp           # Metadata de treino
```

O uso da ferramenta é simples e não iremos entrar nos detalhes de implementação, já que o mesmo não é o foco do trabalho e o código fonte encontra-se disponível.