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 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 Floresta Sintática. São descritos aqui aspectos fundamentais do desenvolvimento do parser que envolve extenso trabalho de sintonia dos parâmetros da ferramenta de Bikel, bem como algorítmos de pré e pós processamento do corpus e outras estratégias utilizadas na busca melhorar a acuidade do parser. O processo é inerentemente empírico. Utilizou-se uma metodologia incremental na busca de melhores configurações baseado em rigorosa avaliação quantitativa a cada passo seguindo os critérios do PARSEVAL. Nossa melhor configuração com Precision 00, Recall 00 e F-Score 00, que é superior aos resultados obtidos em trabalhos semelhantes da literatura.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

CFG *Context Free Grammar* - Gramática Livre de Contexto

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 língua - ou 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 e entender 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 estatísticos, para tentar "entender" corretamente a estrutura e o significado da linguagem. Fundamental nesse processo foi o surgimento de *corpora* anotados (*treebanks*) [MAR93, MAR94, ABE03, SAR04].

Este trabalho pretende estudar o processo estatístico de *parsing* baseado em *corpus* aplicado à língua portuguesa. Iremos focar o processo de *parsing*, mais especificamente as

abordagens estatísticas baseadas em *corpus*, para solucionar esse problema e desenvolver, especificamente, o processo de *parsing* probabilístico aplicado à língua portuguesa. Iremos utilizar como base os estudos e o *parser* desenvolvido por Michael Collins [COL99, COL97] na versão posterior reimplementada por Dan Bikel [BIK04].

1.1 Motivação

A motivação 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. Técnicas como *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 outros métodos, 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. 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. Ambos os *parsers* foram amplamente testados para a língua inglesa e, 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 tem como objetivos principais estudar e compreender as técnicas estatísticas de processamento de linguagem natural implementadas por Michael Collins, utilizar e analisar o *parser* reimplementado por Dan Bikel, que tornou possível a parametrização e extensão das suas bibliotecas para possíveis adaptações do código. Os objetivos específicos são os seguintes:

- Estudar as técnicas envolvidas no desenvolvimento de um *parser*.
- Estudar os modelos de *parsing* de Collins.
- Dominar o uso da ferramenta de *parsing* de Bikel e, se necessário, a original de Collins.
- 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* será feito utilizando-se o *corpus* anotado Floresta Sintática (Projeto Linguateca).
- Desenvolver módulos de pré e pós processamento do corpus, e as alterações de códigos que se mostrarem necessárias.

1.3 Organização do texto

Este trabalho está organizado em seis capítulos. Inicialmente neste primeiro capítulo, aborda-se uma introdução ao 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 e de que maneira os experimentos foram conduzidos, descrevendo os testes realizados juntamente com seus respectivos resultados. No quinto capítulo são abordadas algumas considerações finais e finalmente no sexto capítulo discutem-se alguns possíveis trabalhos futuros.

2 Referencial Teórico

O presente trabalho de conclusão situa-se na área de processamento de linguagem natural, conforme ilustrado na Figura 1.

No texto que segue, abordaremos alguns 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,

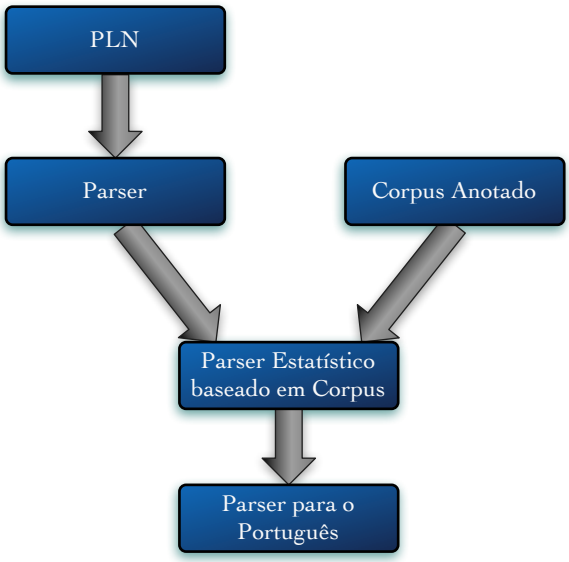

Figura 1: Estágios do processamento de linguagem natural propostos neste trabalho

verbos, etc.), configurando uma sentença ainda válida. 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. Na medida em que uma palavra é caracterizada pela sua categoria principal mais traços secundários, não é surpresa que haja uma razoável variabilidade na separação entre que características já devem estar embutidas na POS, e quais devem ser relegadas a *features*. Por exemplo, algumas propostas podem selecionar como POS nome e como *feature* número (singular ou plural). Outras podem atribuir POS *tags* (marcações de POS) separados para nome-singular e nome-plural.

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. 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.

O *corpus* que será usado no trabalho tem seu *tagset* definido em [BRA08].

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 ou categorias sintáticas de mais alto nível.

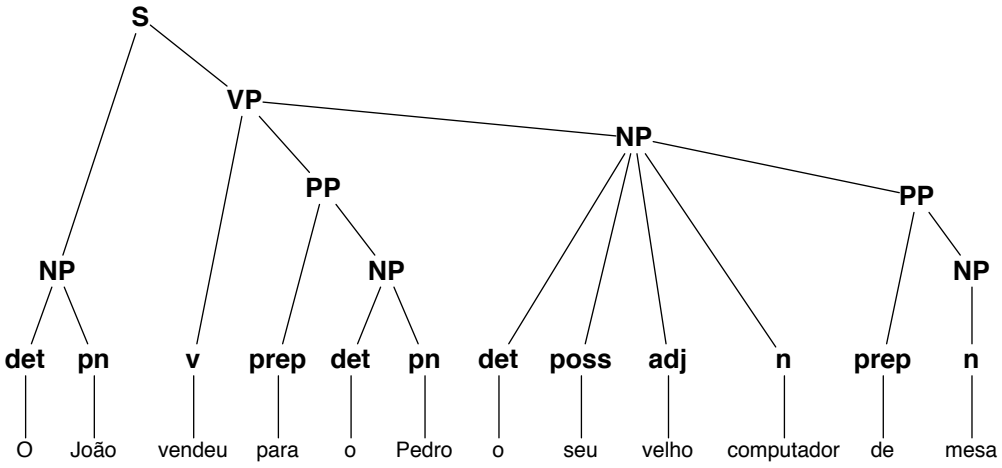

Figura 2: Árvore gramatical da frase *O João vendeu para Pedro o seu velho computador de mesa*

Por exemplo, a 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.

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*:

3. Estatística: fundamenta-se na utilização de teorias estatísticas. Por exemplo, Biber [BIB93] emprega fórmulas matemáticas para identificar quantidades mínimas

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

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 [GÓM97, GÓM97a], que estima matematicamente a quantidade do vocabulário presente em *corpora* de diversos tamanhos hipotéticos.

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)
DNC (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

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.

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, O Bank of English é um corpus monitor, orgânico e em crescente expansão.

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 rótulos sintáticos e de POS (*tagset*) usado no Penn Treebank, como muitos outros corpus, foi baseado no Brown Corpus e é mostrado nas tabelas 4 (Rótulos de Part-of-Speech) e 5 (Rótulos 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 encontrado no *website* do projeto Penn Treebank em http://www.cis.upenn.edu/ treebank.

2.2.2.3 Corpus 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.

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

Tabela 4: *Rótulos 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: *Rótulos 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

Tabela 6: *Corpus da Língua Portuguesa.*

Corpus	**Palavras**	**Composição**	**Localização**
Banco de Português	233 milhões	português brasileiro, escrito e falado	PUC/SP
CETEM (Corpus de extração de Textos eletrônicos MCT), publico	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)

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 revistos manualmente por linguistas.

Atualmente, o corpus da Floresta Sintá(c)tica tem 4 partes, que diferem quanto ao gênero textual, quanto ao modo (escrito vs falado) e quanto ao grau de revisão lingüística: o Bosque, totalmente revisto por lingüistas; a Selva, parcialmente revisto, a Floresta Virgem e a Amazônia, não revistos. 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 9.368 frases, retiradas os primeiros 1000 extratos (aproximadamente) dos corpora CETENFolha e CETEMPúblico. Desde 2007, o Bosque vem passando por um novo processo de revisão, em que foram corrigidas algumas pequenas inconsistências e acrescentadas novas etiquetas. A versão final, disponível para consulta e download, é o Bosque 8.0.

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, uma extensa documentação das opções lingüísticas tomadas durante o projeto.

2.2.2.4.1 Esquema de anotação

Na Floresta, a cada palavra são associadas etiquetas (ou rótulos) principais (de função e de forma) e secundárias. Estas etiquetas aparecem como FUNÇÃO:forma, aonde forma corresponde ao conceito de POS. Em "a menina gulosa", por exemplo, temos:

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

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

A anotação ">N" para a palavra "a" de função, e 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 rótulos sintáticos são listados na tabela 8.

Uma descrição detalhada do tagset da Floresta é encontrado no website do projeto Floresta Sintática em `http://linguateca.dei.uc.pt/Floresta/BibliaFlorestal/anexo1.html`. A versão atual do Bosque é 8.0, de 13 de Outubro de 2008, com 9.437 árvores revistas, correspondendo a 1962 extratos, 215.420 unidades e aprox. 183.619 palavras.

2.2.2.5 Projeto Semantic Share

Um objetivo principal do SemanticShare é o desenvolvimento para o português de corpora anotados da mais recente geração e da próxima geração - um PropBank e um

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

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 integrada.

Podem ser apresentadas em uma ou mais de entre várias vistas:

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

São arquivadas num formato de representação interno que é linguisticamente bem informado, seguindo um quadro gramatical de primeira linha para a lingüística computacional (HPSG);

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 objetivos estão ao alcance do projeto na medida em que tiram partido da maioria das ferramentas e recursos desenvolvidos pela equipa do SemanticShare em projetos anteriores bem sucedidos, nomeadamente o projeto TagShare, do qual o SemanticShare é uma continuação. Eles constituem uma coleção única de ferramentas de última geração para o Português – segmentador de frases e lexemas, etiquetador morfossintático, lematizador, analisador morfológico, reconhecedor de entidades nomeadas –, juntamente com o respectivo corpus de 1 milhão de ocorrências, anotado com precisão de acordo com as dimensões 1.-6. acima (serviços online em http://nlxgroup.di.fc.ul.pt).

Tudo isto será realizado com o apoio do consórcio DELPH-IN, uma iniciativa de nível mundial que visa dinamizar investigação de ponta em processamento lingüístico profundo através da partilha de ferramentas de desenvolvimento "open source", de recursos e de boas práticas (http://wiki.delph-in.net) entre os seus participantes convidados (vd. carta de convite em http://www.di.fc.ul.pt/ ahb/semanticshare.htm).

A sua plataforma tecnológica e a sua ferramenta de anotação sem rivais permitem avanços rápidos na concretização dos objetivos do projeto, dando assim continuidade à cooperação anterior, nomeadamente no quadro do projeto GramaXing, em que uma gramática para o processamento lingüístico profundo foi desenvolvida e está a ser mantida.

Para além disso, parte do banco lingüístico a ser desenvolvido é a componente portuguesa de bancos paralelos que estão a ser desenvolvidos para outros idiomas por outros membros do DELPH-IN, segundo requisitos similares.

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

- fornecer uma base empírica para o estudo lingüístico deste idioma e para o desenvolvimento de ferramentas elaboradas manualmente;
- treinar ferramentas de base estatística para o processamento superficial e profundo, incluindo parsers, etiquetadores de papéis semânticos, etc;
- avaliar ferramentas de processamento;
- apoiar a experimentação de abordagens inovadoras em PLN multilingue, incluindo

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

tradução automática estatística ou meta-anotação automática para a web semântica, etc...

2.2.2.5.1 Esquema de anotação

O esquema de anotação do projeto Semantic Share são visões baseadas na anotação base do projeto que utiliza HPSG [BRA08].

A partir desta anotação são extraídas “visões” no formato de árvores de constituintes. Estas árvores formam o corpus de pesquisa utilizado neste trabalho.

A tabela 9 mostra os rótulos de POS e a tabela 10, os rótulos sintáticos.

Nota: Alguns sintagma são com informação de extração, por exemplo “S/NP” significa sintágma de sentença com extração de NP (sujeito), VP/NP significa sintágma 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.

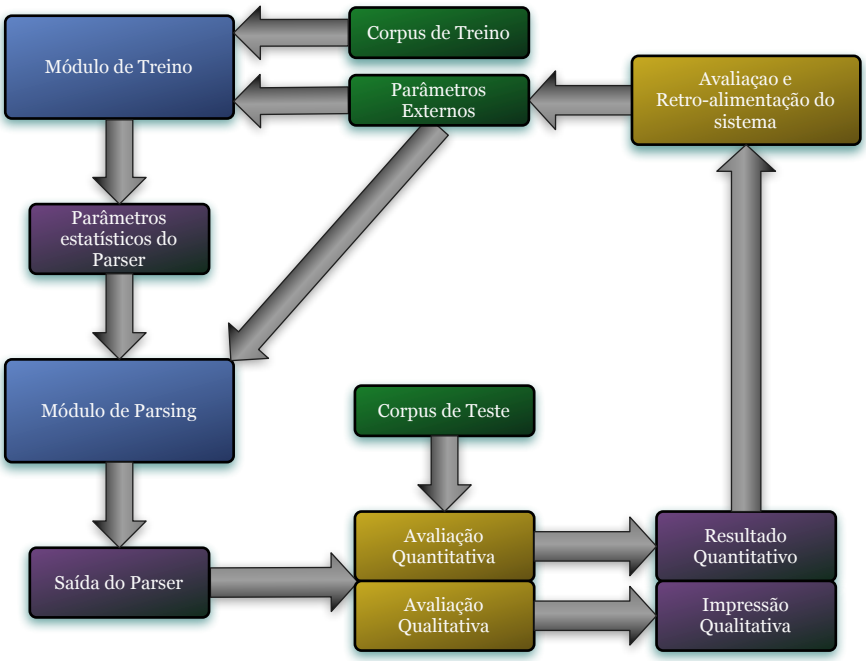

Figura 3: Estágios do processamento de linguagem natural proposto pelo trabalho

2.3 Diagrama geral do processo de *parsing* 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):

   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* [1]. 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 insenta, 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, os de Eckhard Bick [BIC00], baseado em regras, e portanto difícil de ser expandido ou adaptado; [WIN06] e [BON03], que também são estatísticos, baseados no modelo de Collins. Entretanto, revisando o que já referimos, os

[1] O conceito de *overfitting* 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 apresentados 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 muito específicas características 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 dados invisível torna-se pior.

resultados até agora obtidos ainda estão distantes dos desejados.

2.4.1 Trabalho de Benjamin Wing e Jason Baldridge

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

Suas métricas de desempenho utilizadas foram o PARSEVAL padrão, 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 tranformação do corpus para criação de um corpus com as relações de dependência.

Foi provado que fazendo mudanças simples nos dados e na parametrização do parser de Bikel, incluínddo sensibilidade morfológica do português, resulta em sensivel melhora do desempenho atingindo 63,2% de PARSEVAL 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 Daldridge fizeram uma conversão do formato nativo para o formato PennTreebank (PTB), para este trabalho tambem tivemos que fazer tal conversão uma vez que o parser de Dan Bikel espera como entrada arquivos nesse formato. Foram feitas modificações também quando a pontuação para que esta seja melhor interpretada pelo parser em formato PTB, por exemplo '.', '?' e '!' foram marcadas como '.'. O corpus do Floresta possui constituintes descontínuos, que na conversão foram mapeados em componentes separados.

Em principio a informação de núcleo (*Head*) das sentenças geralmente marcada explicitamente no corpus Florésta Sintática, seria de grande ajuda no processo. O PTB não contempla essa informação. Normalmente os analisadores baseados em heads das sentenças usam complexos conjuntos de heurísticas para definir os heads das sentenças durante a análise. Wing e Baldridge utilizaram-se desta marcação disponível no Floresta. No entanto, como nem todas as sentenças possuem a informação de head, Wong e Baldridge, ainda tiveram que utilizar um complexo conjunto de heurísticas para resolver as omissõese casos em que discordaram da informação constante no corpus. Neste

trabalhos optamos por ignorar a marcação fornecida no corpus e utilizar o mecanismo disponibilizado pela ferramenta de Bikel para definir o núcleo das sentenças.

Outra mudança feita por Wing e Baldridge no corpus foi com relação as cláusulas conjuntivas. Cláusulas conjuntivas no Floresta são normalmente marcadas com a TAG 'CU' (*Coordinating conjunction*)(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. Foi necessário aumentar os tipos sintáticos de orações na tentativa de que essa situação não ocorresse.

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.

2.4.1.2 Adaptações no analisador para o português

Foi utilizado o parser desenvolvido por Dan Bikel [BIK02] para treinar e executar a análise das sentenças do português, O analisador implementa e estende os modelos de análise de Michael Collins [COL99], que inclue análise lexicalizada orientada ao núcleo das sentenças, modelos que incorporam diferentes níveis de informações estruturais, que ja foram descritas anteriormente nesse trabalho.

O modelo de análise utilizado é essencialmente o Modelo 2 de Collins [COL99].

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 de heads das sentenças, tratamento especial para quando os heads são explicitamente marcados, características morfológicas, e algumas opções de ajuste do analisador ao Floresta.

Modelos baseados no head das sentenças deve permitir saber quem é o filho do head 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 descobrir o head da sentença.

Para o português, para cada tipo de componente uma lista ordenada de tipos sintáticos é fornecida. modificamos essas regras como apropriado para ser utilizado com o Floresta.

Foi necessário tambem modificar o parser para indicação explícita dos heads, e utilização de arquivos de configuração dessas regras caso não esteja indicada no corpus.

Cada pacote de idioma também pode ser codificado com base nas características morfológicas de uma palavra, estas são especificamente importantes para palavras desconhecidas. Cinco características são codificadas para cada palavra, capitalização, hifenização, numérico, inflexão e derivação. Os três primeiros indicam respectivamente se as palavras estão capitalizadas contém hífem ou estão sob forma de número. Para a maior parte, o código para crialos não precisava mudanças. Já para os dois últimos ítens tivemos que fazer modificações para trabalhar corretamente com o português. Pois como falado anteriormente, inflexões verbais e derivações no português são os grandes problemas.

As características inflexionais e derivacionais indicam a presença particular de sufixos nas palavras. Foi criado uma lista de 39 inflexões verbais ou nominais reconhecidas par ao português, isso exigiu cuidado para não bater falsos positivos e, ao mesmo tempo evitar a propagação de características das palavras. Deste modo temos únicos modos para lidar com varias terminações na terceira pessoa do plural do subjuntivo, mas separando 'ado' e 'ido' para evitar falsos positivo 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 nominal ou verbal. 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 ingles o plural pode ocorrer com 's' depois de várias consoantes diferentes.

Outras pequenas modificações foram feitas no parser, por exemplo o modelo de Knesser-Ney foi utilizado ao invés de utilizar o padrão que é Witten-Bell. Outro exemplo foi utilizar o parametro nunkownWordThreshold=2 ao invés de 6 que é o padrão, e desligar todos os parametros que faze referencia ao PTB, e finalmente nao permitindo que oparser crie produções unárias.

2.4.1.3 Experimentos

Wing e Baldridge trabalharam com três diferentes configurações de dados e parser 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 para o inglês. A segunda configuração leva em consideração o corpus Floresta sem alteração mas utilizando o pacote de configuração para o português. 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 tereceiro de 67.1%

Os desempenhos com relação a configuração básica tiveram grande melhora, simplesmente colocando definindo o pacote para português ao invés de utilizar o padrão inglês.

Outra melhora substancial nos resultados se deve a informação de heads baseados nas regras do português informadas manualmente. Ao se adaptar um analisador como o de Bikel par auma nova linguagem vale claramente a pena colocar um mínimo de esforço para se definir um head-find rules razoavel.

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 utilizado para analisar sentenças da lingua portuguesa do Brasil. Implementando o método de modelo gerativo de Michael Collins [COL99]. Como resultado apresenta uma ferramenta para processamento de linguagem natural PAPO formado por varios 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 provaveis 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 aparentes que surgissem na análise de um conjunto seleto de sentenças. Realizando análise apenas nas sentenças consideradas mais dificeis.

Essa avaliação inicial motivou o surgimento da versão II de sua ferramenta, que salvo um único teste, jamais teve desempenho inferior a versão inicial, na verdade superou significativamente na maioria dos casos.

Entretanto nao 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. Foram identificados problemas quanto a ruídos na base de regras, segundo Bonfante provenientes do pré-processamento do treebank, e não originário dessa. E caracteristicas do modelo de análise sintatica do treebank utilizado que virtualmente impedem um aprendizado efetivo por parte de sua ferramenta, e que poderiam ser neutralizados de forma relativamente fácil por meio de um pré-processamento, ainda automático, mais elaborado.

Em todos os experimentos realizados na tese de Bonfante, o sistema utiliza como fonte de exemplos de análise, o CENTENFolha, proveniente do Floresta Sintática. Um corpus de cunho jornalístico anotado com o parser simbólico e o esquema de anotação propostos e descritos por Eckhard Bick [BIC00]. Como este esquema de anotação tem características bastante peculiares que tornam o pré-processamento não trivial, a ponto de inclusive ser uma fonte significativa de ruído existente na entrada do gerador de modelos para sua ferramenta.

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 saida 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 são identificados para cada regra. Para preencher o núcleo de um sintagma é necessário que seja recuperada a regra na qual ela é pai.

2.4.2.1 Experimentos

Para avaliara 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 terminasse a analise de uma sentença no prazo máximo de cinco minutos, considera sem solução.

Os resultados sao apresentados de forma quantitativa e quanto ao tempo de processamento, para processar as 23 sentenças a ferramenta levou em média 47 segundos, seus melhores resultados quanto a Precision é 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 é um universo pequeno de casos para se avaliar uma ferramenta com tal proposito.

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 Gramática Livre de Contexto Probabilística, que a partir de agora vamos nos referenciar como PCFG (*Probabilistic context-free grammar*). Coll PCFGs são uma extensão de uma gramática livre de contexto, só que existe uma probabilidade associada a cada regra de substituição.

Segundo Collins [COL99], 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.

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.

Um algoritmo bastante usado é o *Inside-Outside* (Charniak 1993). Usando um corpus grande e o algoritmo, o modelo pode ser treinado automaticamente, considerando todas as análises possíveis da sentença no corpus de treino.

A ambigüidade 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. $S \rightarrow SP^1 \mid SV$
2. $SV \rightarrow SV\ SP^2 \mid V\ SN \mid V\ SP^3$
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 sentença 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 condição de contorno de que 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 $P(\alpha i|wi)$, onde αi é a palavra e wi 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 $A \rightarrow \alpha$ 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:

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 (2003), 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.

Tendo a ambiguidade como 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 sintágmas 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

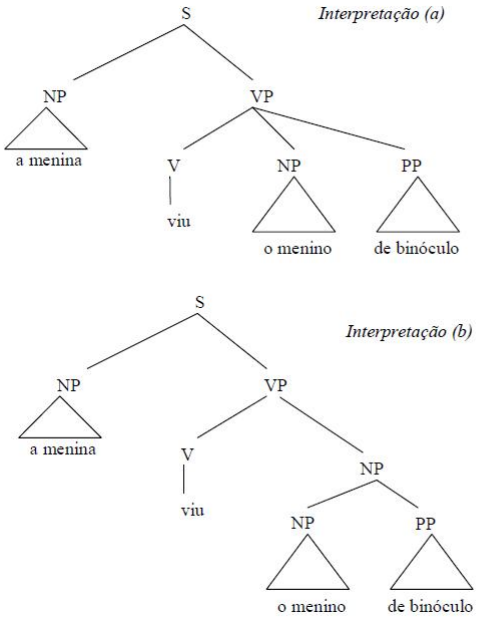

Figura 4: Imagem das árvores da frase *A menina viu o menino*

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.

A falta de *treebanks* no início das pesquisas com *parsers* estatísticos gerou um grande

numero de abordagens quanto ao treino e avaliação dos *parsers* existentes.

3.3 Problemas encontrados na Gramática Livre de Contexto Probabilística

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.

Desafortunadamente, após algumas pesquisas descobriu-se que o uso apenas de PCFGs era insuficiente para PLN.

A percepção dessas falhas 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 de uma PCFG, tendo resultados encorajadores. Considerando um modelo baseado em regras, mais uma vez, com mais sensibilidade ao contexto que PCFG. Outras pesquisas considerando versões parcialmente supervisionadas do algoritmo Inside-Outside: Considerando a idéia de que o *treebank* como TBI, relativamente plano, com árvores não tão anotadas, e que o algoritmo de aprendizagem seria capaz de usar as informações desse *treebank*, enquanto aprendia mais detalhes de maneira não supervisionada.

Todos esses modelos continuavam com falta de sensibilidade lexical.

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 triplos 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 ambigüidade.

3.6 History-based Models

Uma terceira linha de pesquisa, *history-based models*, foi desenvolvida por pesquisadores da IBM. Estes modelos foram caracterizados por duas diferenças de uma simples PCFG. Primeiro, árvore de análise representada foi incrementada de duas maneiras: *tags* não terminais foram estendidos para incluir informação como itens léxicos (*heads*, *words*), ou categoria semântica; o condicionamento ao contexto foi estendido para prever a potencialidade de todas as estruturas anteriores geradas, ao invés de apenas expandir os símbolos não terminais como na PCFG. Segundo, mais poderoso método de aprendizado de máquina, em particular, árvores de decisão, foram usadas para estimar os parâmetros. A idéia básica foi expandir as funcionalidades e o contexto, incluindo todas as fontes de informação para desambiguação ; então usar árvores de decisão para aprender exatamente qual conjunto de combinações era importante para analisar. Um importante avanço nessa pesquisa foi a troca de uso de gramática "*hand-crafted*" por modelos de treino utilizando *treebanks*.

History-based, é um modelo gerativo probabilístico que usa a vantagem de possuir informação lingüística detalhada para resolver ambigüidade. Os autores abordam o sentido de contexto da forma mais fiel possível com o que nós humanos consideramos, racionalizando a quantidade de informação da sentença que é necessária e suficiente para determinar sua análise.

3.7 Modelos de Michael Collins

Collins [COL97] 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, ente 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.

3.7.1 Modelo 1

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:

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

Aonde $NUC(nuc)$ representa o núcleo do sintagma, que recebe o item lexical nuc de

seu pai Pai; $E_1...E_n e D_1...D_m$ são seus 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).

3.7.1.1 Adicionando Distância

Collins [COL97], 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).

3.7.2 Modelo 2

O Modelo 2, 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.

3.7.3 Modelo 3

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. $S(+gap) \rightarrow SN-CSV(+gap)$

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

4. $SV(+gap) \rightarrow VerboTraceSN$

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

4 Experimentos e resultados obtidos

4.1 Metodologia

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: a) de pré-processamento ou pós-processamento dos casos; ou b) no código do *parser*. Nesse sentido, a avaliação quantitativa é um componente importante e será feita de forma rigorosa. Pretende-se utilizar 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 treino, desenvolvimento e teste utilizado é o Bosque da Floresta Sintática que contém um total de 5200 sentenças separadas em 4160 para treino 520 para desenvolvimento e 520 para teste, nas primeiras baterias de teste utilizamos pouco menos de 100 sentenças para ajustar parâmetros mais rapidamente, na última foi utilizado 520 sentenças na fase de desenvolvimento e teste.

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 na construções da língua que pode gerar ambiguidade ou estruturas complexas, o que dificulta a análise (em alguns casos dificulta até a análise humana).

Não foi feita alteração quanto as TAGS utilizadas no corpus de trabalho, apenas

quanto a pontuação de hífen, que será substituida por Underline.

O Conjunto de experimentos foi dividido em sub-grupos que tentam avaliar a melhor configuração com respeito a algum fator relevante ou parâmetro. Para cada grupo é eleita uma melhor configuração e a partir destes os prosseguem.

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 de GEIC/Parseval [BLA91], possivelmente 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:

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

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

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

O termo *Labeled* se refere ao fato de que uma 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 tres medidas: *crossing braqkets, precision e recall*, neste trabalho não utilizaremos a medida de *crossing braquets*.

Estas métricas são chamadas métricas estruturais, e são baseadas na avaliação dos limites dos sintágmas. Os algoritmos de parsing tem 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 constituíntes. Assim dado um nó em uma árvore sintática, a sequencia de palavras dominadas por esse nó forma um sintágma, sendo o limite do sintágma representado por um interválo inteiro *[i,j]*, em que i representa o índice da primeira palavra e j o da última palavra do sintágma.

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 sintágma, ou nível de sentença. As medidas de Labeled Precision e Labeled Recall são computadas da seguinte forma:

Os limites dos sintágmas na resposta (análise produzida pelo parser) e no gold (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 do treebank a ser usado na avaliação. O Labeled Recall é definido como a percentagem no gold que também é encontrada na resposta ((A K)/K). A Labeled Precision é definica como a percentagem de limites no sintagma da resposta que também é encontrada no gold ((A K)/A).

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

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

1. (Eles ((chegaram) ontem))
2. ((Eles (chegaram)) (ontem))

Temos:

limite para os sintagmas em (1): (0,2),(1,1) e (1,2).
limite para os sintagmas em (2): (0,2),(1,1),(0,1) e (2,2).

Pontuações em (2): Labeled Precision = 2/4 = 50%, Labeled Recall = 2/3 = 66.7%

Essas pontuações tem que ser consiteradas juntas para ter significado. Por exemplo, tratando a sentença como uma lista de palavras (Eles chegaram ontem) teriamos 100% de Labeled Precision. Entretanto, Labeled Recall seria baixa (1/3 = 33%).

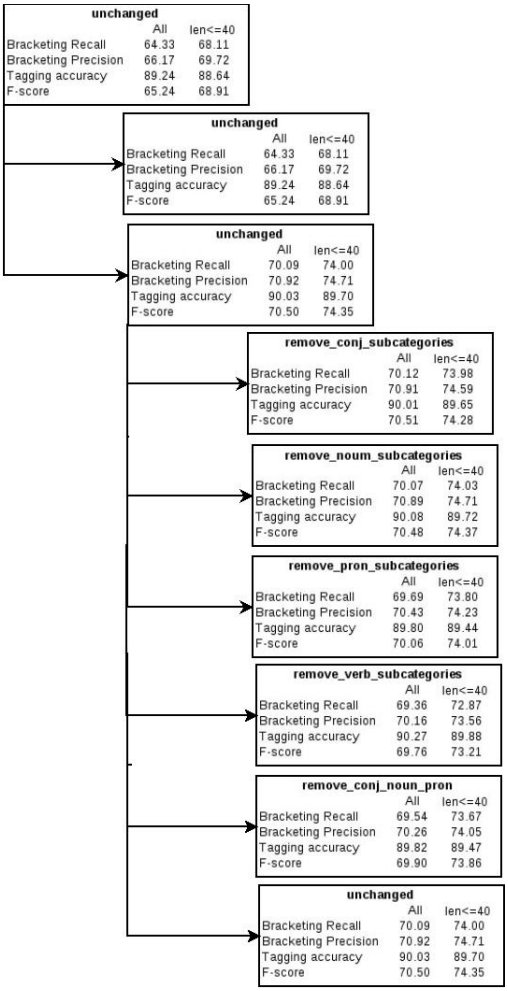

Figura 5: Evolução dos resultados

4.3 Descrição dos experimentos

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

O corpus do Bosque ja tem anotados muitos dos *heads* das sentenças disponibilizadas, porém o parser de bikel precisa de todos os *heads*, e verificou-se que nem todas as sentenças 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 *head* de um constituinte e optou-se por se usar esse módulo e construir regras baseadas nas regras de formação da lingua portuguesa incrementalmente (já exemplificado em capítulo anterior).

4.3.1 Configurações

Primeiramente trabalhamos com a configuração default para o inglês, com parâmetros e head-find rules definidas para a gramática definida para o PTB.

A segunda configuração de testes usamos as configurações para o português definidas por Dan Bikel em seu parser e com as regras de head-find rules básicas, ou seja, primeiro o núcleo da sentença é a primeira palavra da direita para a esquerda, regra essa que é padrão na lingua inglesa, logo se alterou para que o núcleo da sentença seja a primeira palavra da esquerda para a direita, regra que ja se aproxima da regra de formação da lingua portuguesa.

Após testes com essas configurações iniciais, trabalhou-se incrementalmente na evolução tanto de parametros melhores do parser quanto a definição das head-find rules para a lingua portuguesa.

É interessante mensionar que a construção das regras foi incremental e experimentos foram sendo feitos para avaliar a qualidade das regras criadas como ilustrado na figura 5 ....

O último conjunto de regras utilizado é o que estamos usando atualmente, bem melhor que o original conforme resultados apresentados. Acreditamos que não teremos ganho incremento nos resultados tentando melhorar as regras de head-find, e sim se ajustarmos melhor os parâmetros da ferramenta.

A segunda sequência de experimentos tera como objetivo avaliar a granularidade das POS tags no corpus utilizado. Será feito o agrupamento de TAGS que possuem subgrupos dentro da mesma categoria.

A ideia básica é que tags devem ser distintos quando a categorias tem distribuições sintáticas diferentes, por outro lado se duas classes tem mesma distribuição ou distribuição próxima, separa-las apenas levará a perda de qualidade quanto a informação sintática constante nas sentenças ..., ...

Para avaliar o efeito da distinção de categoria verbal em diferentes distribuições ....

Removendo subcategorias de verbo separadamente... Removendo subcategorias de conjução separatamente... Removendo subcategorias de pronomes separadamente... Removendo subcategorias de substantivo separadamente...

Removento todas as sub categorias das TAGS na qual possuem sub categorias.

4.3.2 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 e muito a tarefa de análise morfológica. Dificuldade esta também relatada por Wing e Baldridge em seu trabalho.

Verificou-se conforme esperado que para verbos é bastante relevante as sub-categorias pois a distribuição sintática das diferentes categorias verbais é bem distinta, da mesma forma para pronomes porque os possessivos tem diferente distribuição que os pessoais; os .. tem posição de pré-modificadores nominal e os outros ...

Diferenciando conjunçao coordenada e subordinativa nao houve diferença significativa nos resultados.

O último caso relativo a nomes refere-se o que ... do corpus

Foi dada grande atenção quando ao tratamento das palavras desconhecidas pois acreditamos que um ajuste nesse ponto se pode alcançar desempenhos melhores.

4.3.3 Experimentos com lematização das palavras

Como experimento mais avançado foram feitos testes utilizado lematização das palavras do corpus, primeiro com todas as palavras das sentenças, logo apenas com os verbos. A fim de verificar a qualidade do treino executado pela ferramenta de Bikel.

Um analisador sintáxico automático permite resolver corretamente as ambiguidades ligadas à sintaxe. Ao aplicar-se as regras gramaticais à frase e às proposições que a compõem, pode-se, na maioria dos casos, diferenciar verbos, substantivos, adjetivos e substitui-los pela sua forma canônica (singular de um substantivo, infinitivo de um verbo por exemplo), mas também identificar as palavras compostas e as locuções. Esse trabalho aplicado ao texto se chama lematização.

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

No português, palavras aparecem em várias formas flexionadas. Por exemplo, o verbo

'caminhar' pode aparecer como 'caminha', 'caminhado', 'caminhou'. A forma de base, 'caminhar', que se pode olhar em um dicionário, é chamado o lema para a palavra. A combinação da forma de base com a parte do discurso é muitas vezes chamada de lexema da palavra.

Lematização está intimamente relacionado com decorrentes. A diferença é que um derivado opera em uma única palavra, sem o conhecimento do contexto e, portanto, não pode discriminar entre palavras que têm significados diferentes dependendo da parte do discurso.

Para os experimentos realizados nesse trabalho, foi utilizado o lematizadorde palavras TreeTagger (Pablo Gamallo Otero , Grupo de Procesamento de Linguagem Natural da Universidade de Santiago de Compostela), 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 lingua morfologicamente rica, onde um verbo pode aparecer em dezenas de formas,diferentemente do inglês. Acreditávamos que a lematização poderia contribuir para se atingir melhores resultados.

4.4 Resultados

Primeiro teste

Regras de *head-find* usadas:

```
(
(NP (l N PROP PRON-PERS PRON-INDP N-ADJ NP))
(VP (l V-FIN V-INF V-PCP V-GER) (l VP))
(ADJP (l ADJ ADJP) (l PRON-DET))
(ADVP (r ADV ADVP))
(CU (r CONJ-C CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Resultados obtidos

```
Bracketing Recall         =  62.38
Bracketing Precision      =  68.80
Tagging accuracy          =  58.84
```

Achamos que os resultados estavam baixos devido as TAGS de anotação dos verbos possuírem um hífen, e o *parser* não reconhece o hífen como parte da TAG.

Segundo teste

Alteramos as *tags* que possuíam o caractere hífen ("-"), pois o *parser* desconsidera a parte a direita do mesmo, e o programa de avaliação não. Por isso os resultados da avaliação são prejudicados.

As regras de *head-find* foram alteradas para:

```
(
(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))
)
```

Resultados obtidos

```
Bracketing Recall         =  61.47
Bracketing Precision      =  68.20
Tagging accuracy          =  79.83
```

Apenas a precisão das tags melhorou, porém não houve melhora nas métricas *Bracketing Recall* e *Bracketing Precision*.

Terceiro teste

No próximo teste pretendemos verificar o quanto o *parser* consegue melhorar seu desempenho quando não tem que especificar se o verbo esta em uma das quatro categorias, tendo apenas a anotação genérica de verbo.

As regras de *head-find* foram alteradas para:

```
(
(NP (l N PROP PRONPERS PRONINDP NADJ NP))
(VP (l V) (l VP))
(ADJP (l ADJ NUM) (l 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))
)
```

Resultados obtidos

```
Bracketing Recall         =  60.42
Bracketing Precision      =  67.42
Tagging accuracy          =  80.73
```

Novamente, a precisão das tags aumentou, porém o *Precision* e *Recall* diminuíram. A generalização da tag V fez com que categorias sintáticas externas como ICL, ACL e FCL fossem influenciadas.

Quarto teste

Como já verificado com as tags, o *parser* também exibe problemas com o caractere hífen (“-”) no texto. Após substituir todos os hífens por *underscore* (“_”), os resultados obtidos foram melhores.

As regras de *head-find* são as mesmas.

```
(
(NP (l N PROP PRONPERS PRONINDP NADJ NP))
(VP (l V) (l VP))
(ADJP (l ADJ NUM) (l 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))
)
```

Resultados obtidos

```
Bracketing Recall         =  67.42
Bracketing Precision      =  68.09
Tagging accuracy          =  90.36
```

Percebemos uma melhora significativa nos valores de *Bracketing Recall* e *Tagging accuracy*, a partir de agora sempre substituiremos os caracteres "-" por "_".

Quinto teste

Na próxima experiência vamos utilizar novamente a especialização das categorias de verbo e com um corpus de desenvolvimento e teste maior.

```
(
(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))
)
```

Resultados obtidos

```
Bracketing Recall         =  72.50
Bracketing Precision      =  72.73
Tagging accuracy          =  90.46
```

Verificamos uma melhora expressiva em relação aos primeiros testes realizados após a retirada dos hífens das *tags* de categoria de verbos e após a substituição dos caracteres "-" por "_".

Uma avaliação mais aprofundada será feita posteriormente.

5 Considerações finais

Percebemos que o critério de anotação utilizada em um corpus tem grande influência nos resultados. Verificamos uma melhora expressiva com relação aos primeiros testes realizados após a substituição dos caracteres "-" por "_".

Do ponto de vista experimental, obtivemos resultados de treinamento razoáveis, cuja qualidade parece bastante dependente de uma anotação bem feita. Os próximos passos em nosso trabalho envolvem a análise, de um ponto de vista estrutural das regras implementadas por Bikel em seu *parser*.

Uma das principais alterações que tivemos que fazer foi a construção das *head-find rules* para os constituintes encontrados no corpus utilizado, primeiro porque as TAGS utilizadas pelo *Treebank*, não são as mesmas e segundo porque as regras de formação dos constituintes para o português são diferentes do inglês. Notamos que as *head-find rules* são de grande importância para o desempenho do *parser* de Bikel que implementa o modelo gerativo baseado na noção *head-centering*, em que o núcleo é o elemento principal de todo o processo de geração.

O *Parser* de Bikel, além de ser possível emular os modelos de Collins, também possibilita diferentes parametrizações quanto a algoritmos utilizados e implementação de regras de *head-find*, por exemplo. Alguns parâmetros melhoram a performance quanto a acertos no momento de análise de sentenças mas levam um tempo maior para finalizar, já outras configurações permitem que o tempo de análise das sentenças seja mais rápido porém a qualidade do resultado diminui consideravelmente.

6 Trabalhos futuros

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 com performance semelhante. Pode ser facilmente estendida para novos domínios e novas linguagens. Baseada no modelo 2 de Collins, permite uma grande gama de parametrizações.

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.

Referência Bibliografia

[ABE03] (Abeillé, Anne,Eds.). **Treebanks**: building and using parsed corpora. 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, Douglas. **Methodological issues regarding corpus-based analyses of linguistic variation**. *Literary and Linguistic Computing*, 1990.

[BIB93] Biber, Douglas. **Representativeness in corpus design**. *Literary and Linguistic Computin*, 1993.

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

[BIK02] Bikel, Dan. **Design of a multi-lingual, parallel-processing statistical parsing engine**. 2002.

[BIK04] Bikel, Dan. **Intricacies of collins' parsing model**. *Computational Linguistics*, 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.

[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, Eugene. **Statistical techniques for natural language parsing**. *AI Magazine*, 1997.

[COL97] Collins, Michael. **Three generative, lexicalised models for statistical parsing**. In: *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid, Spain., 1997. p.16–23.

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

[GÓM97] Gómez, Pascual Cantos; Pérez, Aquilino Sánchez. **Predicability 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*, 1997.

[GÓM97a] 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, Daniel; Martin, James H. **Speech and language processing**. Prentice-Hall, 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, Mitchell; Kim, Grace; Marcinkiewicz, Mary Ann; MacIntyre, Robert; Bies, Ann; Ferguson, Mark; Katz, Karen; Schasberger, Britta. **The Penn Treebank**: annotating predicate argument structure. In: *Proceedings of the 1994 Human Language Technology Workshop.*, 1994.

[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.

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

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

[SIN97] Sinclair, John. **Corpus evidence in language description**. 1997.

[WIN06] Wing, Benjamim; Baldridge, Jason. **Adaptation of data and models for probabilistic parsing of portuguese**. *PROPOR 2006*, 2006.