Universidade Federal do Pampa
Natalie Lourenço Vargas

Analisadores Sintáticos de Linguagens Naturais: Uma Comparação do Estado da Arte

Alegrete
2015

Natalie Lourenço Vargas

Analisadores Sintáticos de Linguagens Naturais: Uma Comparação do Estado da Arte

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Fábio Natanael Kepler

Alegrete

2015

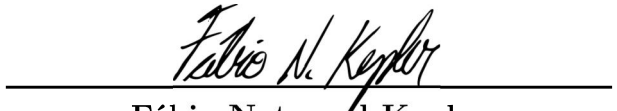
Natalie Lourenço Vargas

Analisadores Sintáticos de Linguagens Naturais: Uma Comparação do Estado da Arte

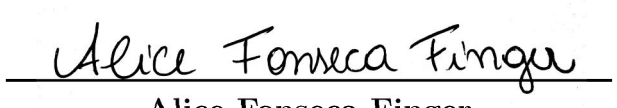
Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho de Conclusão de Curso defendido e aprovado em 03 de dezembro de 2015.

Banca examinadora:



Fábio Natanael Kepler
Orientador



Alice Fonseca Finger
Universidade Federal do Pampa

Marcelo Resende Thielo
Universidade Federal do Pampa

A Deus, à minha família e a todos aqueles que de alguma forma me apoiaram até aqui.

Agradecimentos

Gostaria de agradecer primeiramente a Deus que me deu esta oportunidade e o ânimo necessários para que eu conseguisse chegar até aqui. Agradeço também à minha família que com certeza é a minha base e me incentiva todo o dia. Agradeço ao meu orientador, pelo tempo dedicado à orientação deste trabalho. Aos meus colegas pela troca de conhecimento, companheirismo e dias de muita "zuera", especialmente na sala 303. A jornada foi longa, mas com certeza todos esses agradecimentos são um reflexo dos desafios que enfrentei, mas que felizmente foram vencidos de algma forma.

"O coelho branco colocou os óculos e perguntou:
- Com licença de Vossa Majestade, devo começar por onde?
- Comece pelo começo - disse o Rei, com ar grave - e vá até o fim. Então pare.
(Lewis Carroll em Alice no País das Maravilhas)

Resumo

Parsers, cuja área compõe o campo de Processamento de Linguagem Natural, vêm se tornando cada vez mais estudados e explorados. Além das suas aplicações estarem sempre em processo de evolução, técnicas como deep learning, por exemplo, vêm sendo incorporadas nos *parsers* com o intuito de melhorar o seu aprendizado. Isto significa então que a cada momento os *parsers* surgem com novas soluções tornando difícil a realização de um levantamento de informações no que diz respeito ao que de mais novo está sendo ou foi proposto até então. Além disso, tais levantamentos podem se tornar defasados ou mínimos, pelo fato já citado da área se encontrar em constante evolução. Este trabalho mostra os experimentos que realizamos com *parsers* por dependência e constituência a fim de realizarmos uma avaliação mais profunda dos mesmos. Para os *parsers* por dependência testamos córpus com 12 idiomas e para os de constituência córpus com 3 idiomas. Conseguimos também realizar testes e analisar as dependências projetivas e não projetivas dos *parsers* e dos *córpus*. Infelizmente não conseguimos atingir as mesmas análises para os *parsers* por dependência pela falta de córpus, porém encerramos o trabalho com algumas dicussões a respeito desse problema.

Palavras-chave: Inteligência artificial. Parsers. Corpus. Projetivo. Não Projetivo. Constituência. Dependência. CoNLL

Abstract

Parsers, whose area comprises the Natural Language Processing field, has become more studied and explored lately. Besides its applications being in a evolution process, techniques such as deep learning, for instance, has been incorporated in *parsers* in order to improve learning. This means that every moment *parsers* come up with new solutions and it makes hard to conduct a survey about what is trending in parsing field at the moment or which good solutions were proposed so far. Moreover, surveys can become lagged or small due to the fast evolution of this area. This work shows our experiments with dependency and constituency parsers in order to run an survey about it. Related to dependency parsers, we tested them into 12 languages and regarding to the constituency ones we ran with 3. It was possible to run tests and analyse dependencies projectivities and non projectivities from parsers and corpus. Unfortunately we did not achieve the same analysis and results regarding constituency parsers due to the lack of corpus, however we finish the work pointing out some discussions about this problem.

Key-words: Artificial intelligence. Parsers. Corpus. Projective. Non Projective. Dependency. Constituency. CoNLL

Lista de ilustrações

Lista de tabelas

Lista de siglas

IA Inteligência Artificial

PLN Processamento de Linguagem Natural

Sumário

1 Introdução

Define-se a linguística como o estudo científico da linguagem humana (MARTINET, 1970). Esse estudo pode ser considerado, mais especificamente, como o estudo da linguagem verbal, da gramática e da evolução dos idiomas. Uma das áreas que integram a linguística é a sintaxe, que por sua vez consiste no estudo de como a linguagem combina palavras para formar frases gramaticais. Com o avanço da computação, o campo da Inteligência Artificial(IA), mais precisamente a área de Processamento de Linguagem Natural (PLN), propõe uma automatização desse estudo. Dentro da área de sintaxe existe a chamada análise sintática (*parsing*, do inglês), que no contexto da computação é um processo que analisa uma sequência de entrada lida de um arquivo ou teclado como uma sentença e determina sua estrutura gramatical de acordo com um conjunto de regras já estabelecidas. Em um contexto geral, a área de PLN investiga *parsers*: algoritmos responsáveis por analisar sintaticamente as sentenças, a fim de que tais conhecimentos sejam aplicadas em problemas reais como tradução automática de textos, reconhecimento e sumarização automática.

Atualmente, muitas investigações a respeito desse assunto vêm tomando forma, tradutores de línguas, por exemplo, vêm sendo aperfeiçoados graças ao trabalho de muitos pesquisadores. Assim como a área de PLN, o campo dos *parsers* é cada vez mais estudado e difundido. Pode-se apontar, por exemplo, tentativas de estudo e treinamento de sistemas através do *CoNLL-X shared tasks*. Trata-se de uma Conferência de Aprendizado de Linguagem Natural onde o termo *shared task* corresponde a uma tarefa compartilhada entre participantes dentro do evento. Os organizadores então fornecem dados de treinamento (iguais para todos) a fim de que os participantes treinem e testem seus sistemas possibilitando assim uma melhor comparação entre os mesmos. A cada ano uma tarefa nova é proposta, por exemplo testar e achar o melhor *parser* que realize análise sintática para diversas línguas (BUCHHOLZ; MARSI, 2006). A primeira tarefa compartilhada foi incluída no evento em 1999 e vêm ocorrendo a cada ano desde então. A idealização de uma *shared task*, desta forma, evidencia uma preocupação e esforço não só em comparar e descobrir qual o *parser* que obteve o melhor resultado, mas também incentivar a descoberta e estudo de outras novas soluções.

A área de PLN portanto vem sendo bem explorada e vem crescendo muito rápido. Pode-se citar por exemplo a área de *Deep Learning* que vem se mostrando bastante promissora para aplicações de PLN, neste caso os *parsers*. Desta forma, o levantamento de informações em cima dos *parsers* que foram propostos até o momento podem criar análises um tanto quanto defasadas ou a pesquisa pode se tornar pequena, uma vez que soluções novas surgem muito rápido e é difícil acompanhá-las.

Este trabalho realizou uma investigação mais profunda no que diz respeito a esta área. Elencamos seis parsers do estado da arte com diferentes tipos de treinamento e teste para treze idiomas diferentes. Fizemos avaliações e comparações entre os idiomas e entre os *parsers* também. Verificamos quais técnicas poderiam ser mais úteis no que diz respeito ao desempenho de análise de idiomas diferentes.

1.1 Organização

Este trabalho está organizado da seguinte maneira - no Capítulo 2 discutimos os fundamentos do trabalho, as definições que compõem o nosso estudo e as ferramentas e dados que iremos usar em nossos experimentos. No Capítulo 3 trazemos trabalhos relacionados à proposta deste, explicando o que já foi realizado em termos de testes e comparações de *parsers*. No Capítulo 4 abordamos a metodologia pela qual foi viável a nossa proposta de trabalho. No capítulo 5 mostramos os resultados dos nossos testes e avaliações. No Capítulo 6 encerramos com as discussões finais sobre o nosso trabalho.

2 Fundamentação

Parsing (análise sintática no português), é o processo de estruturar uma representação linear de acordo com uma dada gramática. Essa definição tem sido mantida abstrata de propósito, a fim de permitir que uma ampla interpretação da mesma seja possível. A "representação linear" pode ser uma sentença, um programa de computador, um pedaço de música, entre outros (DICK; CERIEL, 1990). *Parsers*, portanto, são os algoritmos que fazem o papel de analisadores sintáticos das sentenças. Um *parser* é capaz de analisar uma sentença e recuperar sua estrutura sintática de acordo com uma dada gramática formal. A gramática é classificada como um formalismo de representação para a ordem e o arranjo das palavras de uma língua sendo no geral composta por um conjunto de regras. Entretanto, se tratando de análise de linguagens naturais como o português, é necessário que regras muito complexas sejam construídas a fim de que se possa tratar a ambiguidade da língua. Para que um *parser* consiga analisar as informações das sentenças é necessário que ele induza uma gramática para isto, pois é muito difícil criar uma gramática de língua humana manualmente.

Esta indução então consiste no *parser* aprender a gramática, isto é, ser treinado através de técnicas de aprendizado de máquina. O *parser* então receberá dados de treinamento a fim de que ele aprenda as regras e em seguida irá gerar um modelo. Este modelo será a gramática que o *parser* irá seguir para analisar as sentenças de um texto, por exemplo. Os *parsers* podem ser treinados por aprendizado supervisionado e não supervisionado e os tipos de dados para treinamento diferem de acordo com a técnica escolhida. Tais conceitos das técnicas e dados serão melhor abordados na seção 2.4

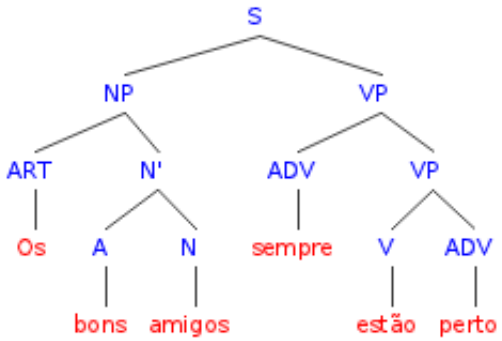
Neste caso, o tipo de parser será probabilístico e será capaz de apontar por exemplo, quem na frase é sujeito ou verbo e podem ser divididos em Constituência e Dependência. Tais conceitos serão melhor abordados na seção a seguir.

2.1 Parser por Constituência

O *Parser* por Constituência consiste na ideia de que grupos de palavras, dentro de declarações, podem ser representados agindo como unidades individuais. Um exemplo de constituência é a frase: Os bons amigos sempre estão perto, onde "Os bons amigos" correspondem a um sintagma nominal, ou seja, três palavras da frase são classificadas de forma única. Os parsers que realizam a análise sintática são capazes de processar, por exemplo, uma frase e a sua saída será uma sinalização do que é artigo, sujeito e complemento de uma frase. No caso dos parsers por constituência essa sinalização será uma árvore onde os nós folha são palavras da sentença. Conforme a Figura 1 podemos

ver a saída da análise do exemplo acima.

Figura 1 – Representação simples de um parser por constituência



Fonte: (NLX, 2015b)

A seguir temos uma breve introdução sobre os parsers por constituência que escolhemos para nosso trabalho.

- **Stanford Parser** [1]

  Stanford Parser é um *parser* estatístico de linguagem natural desenvolvido pelo grupo de Processamento de Linguagem natural da Universidade de Stanford (Stanford Natural Laguage Processing Group). Além do Inglês, este *parser* analisa línguas como árabe, alemão, francês e espanhol e tem sido mantido desde 2002 principalmente por Dan Klein e Christopher Manning. A aplicação é disponibilizada sob a licença GNU-GPL [2]

- **Berkeley Parser** [3]

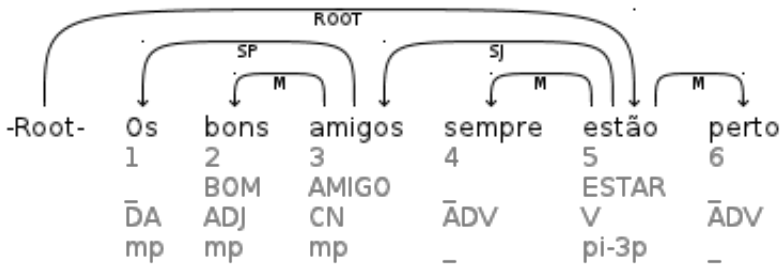
  Berkeley Parser é um *parser* mantido pelo grupo de linguagem Natural da Universidade de Berkeley. Tem tido sucesso no que diz respeito ao treinamento para línguas além do inglês.

2.2 Parser por Dependência

O *parser* por dependência, neste caso, faz uso da predição de palavras: cada palavra na frase tem relação com seus dependentes. Saber a identidade do verbo ajuda a determinar qual é o sujeito ou objeto da frase, por exemplo. Neste caso, a saída do parser não será uma árvore. Podemos ver na Figura 2 a saída de um parser por dependência.

[1] http://nlp.stanford.edu/software/lex-parser.shtml
[2] GNU General Public License é a designação da lincença para Software Livre. Mais informações: http://www.gnu.org/licenses/gpl-3.0.en.html
[3] http://nlp.cs.berkeley.edu/software.shtml

Figura 2 – Representação simples de um parser por dependência



Fonte: (NLX, 2015a)

2.2.1 Parsers Projetivos e Não Projetivos

Como vimos anteriormente, *parsers* por dependência não terão como saída uma árvore e sim uma relação de dependência. Por causa disso, muitas vezes os arcos que demonstram essa dependência podem se cruzar ocasionando uma certa dificuldade do parser em analisá-lo sintaticamente. Quando essas dependências se cruzam ela é chamada de não projetiva. Alguns *parsers* ignoram essas dependências e outros tentam criar técnicas para lidar com elas. Esses cruzamentos de arco ocorrem bastante em línguas onde a ordem dos constituintes (*free word order*) é muito alta. (HAWKINS, 1994)

A seguir temos uma breve introdução sobre os *parsers* por dependência que escolhemos para nosso trabalho:

- **Turbo Parser** [4]

  É um parser de dependência multilíngue exclusivamente não projetivo implementado em C++.

- **MST Parser** [5]

  MSTParser é a sigla para *Minimum Spanning Tree* (MCDONALD et al., 2005) e é um parser por dependência multilíngue.

- **MaltParser** [6]

  Parser implementado em java e projetivo.

- **Yara Parser**[7]

  YaraParser é implementado em Java e é essencialmente projetivo.

[4] https://github.com/andre-martins/TurboParser
[5] http://sourceforge.net/projects/mstparser/
[6] http://www.maltparser.org/download.html
[7] https://github.com/yahoo/YaraParser

2.3 Córpus

Um córpus sintático consiste em um texto onde seus termos e palavras já estão classificados sintaticamente, ou seja, suas respectivas classes gramaticais já estão definidas. O resultado do treinamento serão modelos em formato serializado contendo todas as regras de acordo com o córpus usado.

2.3.1 CoNLL

Fornecido como corpus de treino e teste nas *shared tasks*, CoNLL vem sendo amplamente usado pelos parsers por dependência multilíngue. Escolhemos 12 idiomas para análise (alemão, coreano, dinamarquês, espanhol, francês, hindi, inglês, italiano, japonês , neerlandês, português BR, português e sueco). Os córpus possuem uma leve diferença em termos de origem e anotação. Os córpus para as línguas Dinamarquesa, Neerlandesa e Portuguesa usam o formato CoNLL-X [8] e os demais usam o formato CoNLL-U [9].

2.3.2 WSJ (Wall Street Journal) Penn Treebank

Como o nome diz, está no formato *Treebank* (designa uma estrutura de dados em árvore). O *Penn Treebank* é a representação mais comum desde conceito onde têm-se os galhos (da árvore) dispostos entre parênteses seguindo hierarquicamente com marcações antes de cada nó. O WSJ [10] contém as anotações sintáticas manuais dos textos presentes no Wall Street Journal por três anos. É necessário pagar para usá-lo e é usado em *parsers* por constituência.

2.3.3 TychoBrahe

Córpus de língua portuguesa também no formato *Treebank*, anotado com 65 textos contendo 2.792.217 palavras de textos de autores nascidos entre 1380 e 1845. Destes 65, 16 deles estão anotados sintaticamente constituindo um total de 671.694 palavras. é necessária uma licença para usá-lo. [11]

2.3.4 IcePaHC

Córpus desenvolvido para a língua islandesa. [12] Usado para *parsers* por constituência e compatível com o formato *treebank*.

[8] http://ilk.uvt.nl/conll/post_task_data.html
[9] https://github.com/ryanmcd/uni-dep-tb
[10] https://catalog.ldc.upenn.edu/LDC99T42
[11] http://www.tycho.iel.unicamp.br/corpus/
[12] http://www.linguist.is/icelandic_treebank/Icelandic_Parsed_Historical_Corpus_(IcePaHC)

2.4 Aprendizado de Máquina

Aprendizado de Máquina consiste no desenvolvimento de algoritmos e técnicas com a finalidade de fazer um computador aprender. Esse aprendizado é, na verdade, um aperfeiçoamento de desempenho em alguma ação ou tarefa. Podemos dizer, de forma ampla, que uma máquina aprende sempre que ocorrem mudanças em sua estrutura, programa ou dados (com base em suas entradas e na resposta a informações externas), e é esperado que isso reflita em uma melhoria futura no seu desempenho (NILSSON, 1996).

No contexto de análise sintática, como mencionado na seção anterior, para que os *parsers* executem suas ações é necessário que sejam previamente treinados . Tal treinamento é feito através de algoritmos computacionais de aprendizado de máquina onde dados são usados. Tais dados podem ser córpus sintaticamente anotados, como visto na seção 2.3 ou apenas textos puros. Existem dois tipos de aprendizado que determinam qual o tipo de dados serão usados para treino dos *parsers*: Aprendizado Supervisionado e Não Supervisionado.

Isto significa que esse modelo será o formato de regras que o *parser* usará para fazer a análise sintática dos textos.

O aprendizado supervisionado é uma abordagem que visa treinar um algoritmo para realizar uma tarefa baseando-se em um conjunto de entradas e saídas já conhecidos e corretos. Esse conjunto de dados pode ser chamado de supervisor, ou professor externo e é responsável por estimular as entradas por meio de padrões e observar a saída calculada pela mesma (BRAGA; FERREIRA; LUDERMIR, 2007). Para que técnicas de aprendizado supervisionado sejam usadas no processamento de textos, depende-se de grandes informações prévias para induzir gramáticas, ou seja, os corpus sintaticamente anotados. Essas estruturas de textos já classificadas funcionam como um gabarito para o treino dos *parsers*, ou seja, pode-se identificar se a frase foi classificada corretamente ou não.

No aprendizado não supervisionado, um algoritmo busca resolver um problema sem ter a exata noção dos resultados esperados para tal. Neste esquema de treinamento somente os padrões de entrada estão disponíveis, ao contrário do aprendizado supervisionado (BRAGA; FERREIRA; LUDERMIR, 2007). No contexto dos *parsers*, a técnica de aprendizado não supervisionado utiliza textos puros e busca a indução e detecção de estruturas, ou seja irá induzir uma gramática. Neste caso, não há textos já classificados que servirão como base de comparação com as saídas das análises. A razão pela qual existe um incentivo de uso dessa técnica consiste no fato de que atualmente existem poucos corpus sintaticamente anotados para o português, fazendo com que ela seja amplamente útil na indução das gramáticas.

2.5 Objetivos Específicos

O objetivo deste trabalho foi elencar *parsers* do estado da arte até o momento no campo da análise sintática realizando testes em 13 idiomas diferentes. Selecionamos parsers com diferentes técnicas de aprendizado e selecionamos córpus adequados aos mesmos. Realizamos um estudo realizando testes com córpus diferentes nos parsers e fizemos comparação entre os mesmos. Elegemos os que tinham maior desempenho para determinadas línguas, por exemplo. Com os resultados podemos perceber quais eram as dificuldades em fazer um mapeamento desta área, quais recursos eram os mais escassos e o que poderia ser feito para resolver este problema. Conseguimos fazer uma comparação mais detalhada sobre problemas com córpus de determinadas línguas e verificar o que a área vem tentando desenvolver a fim de melhorar problemas de análise sintática em alguns idiomas. Esperamos, sobretudo, que este trabalho sirva como uma referência para análise de *parsers*. Acreditamos que informações a respeito de uso de dados de treino, teste, parsers e como usá-los, por exemplo, possam ser relevantes e úteis a outras investigações nessa área.

3 Trabalhos Relacionados

Existem muitos trabalhos que estudam e abordam PLN no que diz respeito aos *parsers*, entretanto há conhecimento de poucos trabalhos que realizam uma ampla investigação e comparação dos mesmos. Fazendo um levantamento dos trabalhos, descobrimos que muitos testes e comparações se concentram em poucos *parsers* e córpus.

Kummerfeld et al. (2012) realiza um estudo mais profundo dos erros que os parsers podem gerar e classificá-los testando suas medidas F. Essas análises têm o intuito de responder questões levantadas sobre o comportamento dos parses por constituência.

Ainda na questão de erros, pode-se dizer que *parsers* são robustos se conseguem lidar com “fenômenos fora da sua faixa normal de entradas". O trabalho de Kakkonen (2008) propõe avaliações para verificar se *parsers* do estado da arte são robustos verificando se os mesmos são capazes de analisar sintaticamente sentenças que contém palavras com a escrita incorreta.

Evalita’07 Parsing Task (BOSCO et al., 2008) foi a primeira tentativa de comparação de parsers em língua italiana investigando os já existentes e usados até o momento, usando um *treebank* em comum para realizar os testes. Já o Evalita’09 Parsing Task (BOSCO et al., 2009) busca extender o estado da arte de *parsers* em italiano também investigando referências existentes e usadas até então. O estudo concentra-se na comparação de *parsers* por dependência e constituência.

Córpus também podem influenciar no processamento das informações. Bosco et al. (2010) aborda como diferentes anotações em *Treebanks* podem ter influência nas análises sintáticas por dependência. O trabalho usa diferentes anotações da língua italiana para demonstrar como uma mudança nas estruturas podem ocasionar uma melhora, por exemplo, no desenvolvimento dos *parsers*. Existe outro estudo de comparação de *parsers* em que é questionado apenas o uso do WSJ como *Treebank*. No trabalho de Gildea (2001) propõe-se além do WSJ, o uso do Brown Corpus (FRANCIS; KUCERA, 1979) para fins de verificação de performance das análises sintáticas.

Encontramos também trabalhos que executam testes em *parsers* para a língua francesa. Em (NIVRE et al., 2010) são testados três *parsers* por dependência para o francês e é feito uma comparação entre eles verificando qual deles analisava sintaticamente melhor o texto. Ainda na língua francesa, Seddah, Candito e Crabbé (2009) realiza avaliações de *parsers* treinando os mesmos com córpus em francês.

Por último, em termos de análises de *parsers* por constituência, temos (CARROLL; BRISCOE; SANFILIPPO, 1998) que apresenta uma visão geral do estado da arte de

metodologias e métricas para avaliação de *parsers*. O trabalho propõe, através de análises, novas ideias para medir a acurácia de um *parser*. Para saber se uma solução de fato é a melhor entre as que existem dentro de um grupo, é necessário executar testes nas soluções então disponíveis. O trabalho analisa o que de fato deve ser levado em conta em relação aos resultados para medir um *parser* verdadeiramente robusto da língua inglesa.

Para análises de *parsers* por dependência, podemos destacar Choi, Tetreault e Stent (2015) que desenvolveram uma ferramenta para avaliação de *parsers* por dependência. Tal avaliação foi realizada com córpus diferentes da língua inglesa e comparou-se o desempenho dos *parsers* em termos de acerto e velocidade de análise.

E por último, ainda no campo das análises por dependência, Kukkadapu, Malladi e Dara (2012) adota uma combinação do Malt, MST e Turbo Parsers para testes no córpus da língua Hindi, como parte de uma *shared task* do MTPIL 2012 Workshop, COLING 2012. Testes com os *parsers* foram realizados a fim de estudar quais eram os melhores parâmetros e treinamentos para um melhor desempenho da língua.

4 Metodologia

As análises começaram a partir da escolha dos parsers. Escolhemos quatro parsers por dependência do estado da arte e dois por constituência para a execução dos testes. Testamos os parsers por dependência usando corpus de 12 idiomas diferentes, sendo dois eles variações do português brasileiro e de Portugal. Após uma extensiva procura concluímos que usaríamos o formato CoNLL como córpus para os *parsers* de dependência e usaríamos o TychoBrahe, IcePaHC e WSJ para testes com os *parsers* por constituência. Nas seções a seguir explicaremos detalhadamente a execução dos testes.

4.1 Treinamento e Teste dos Parsers por Dependência

Para que um parser possa realizar a análise sintática, como mencionamos na seção 2.3, primeiro é necessário que ele seja treinado com um córpus de entrada que chamamos de córpus de treino. Os córpus no formato CoNLL (tanto de treino como de teste) originalmente não possuíam os mesmos tamanhos, o que ocasionava em um problema para compará-los. Este problema será melhor abordado na seção 5.2. Uma saída que encontramos foi estabelecer um número específico de 3000 frases para os córpus de treino e 300 para os córpus de teste. É possível ver na tabela 1 que tanto os dados treino e teste variam muito, inviabilizando treinos com diferentes padronizações de números de frases. Uma vez que o menor córpus possui um número aproximado de 300, não há como realizar testes com corpus maiores, por exemplo.

Tabela 1 – Número de Sentenças dos Córpus de Treino e Teste

Idioma	Córpus Treino	Córpus Teste
Alemão	14118	1000
Coreano	5437	299
Dinamarquês	5190	322
Espanhol	14138	300
Francês	14511	300
Hindi	4477	557
Inglês	39832	2416
Italiano	6389	400
Japonês	8277	299
Neerlandês	13349	386
Português BR	9600	1198
Português	9071	109
Sueco	4447	1219

Após o treino, o parser gera um arquivo de saída serializado chamado de modelo. Este arquivo terá as regras que o parser usará para finalmente realizar a análise sintática.

A partir daí para o teste, é necessário fornecer dois aquivos como entrada: um modelo e um arquivo de teste. O parser então fará a análise sintática e irá gerar um arquivo de saída. Para verificar se realmente o parser realizou a análise correta é necessário fazer uma comparação entre o arquivo de saída e o próprio arquivo de teste. Este método de avaliação usa o arquivo de teste como um gabarito, conferindo se o parser gerou um arquivo de saída parecido com o teste.

4.1.1 Treino e Teste Turbo Parser

Como o Turbo Parser é desenvolvido em C++, é necessário seguir as instruções designadas no arquivo README e gerar um executável. Após isso usamos as seguintes linhas para treino do parser:

```
./TurboParser -train -file_train=file_of_training
-file_model=models/parser.model
-prune_basic=false
 -model_type=basic
-logtostderr
```

O TurboParser oferece um modo padrão de treinamento que optamos por não usar. Quando usado este treinamento, às vezes o TurboParser leva muito tempo para treinar, além das 10 iterações que ele realiza de treino. Encontramos problemas com o córpus neerlandês, por exemplo. O treino padrão com córpus relativamente grandes, 1000 sentenças por exemplo, requer um bom computador para que haja uma maior rapidez no treinamento dos modelos. Algumas vezes durante a tentativa do treino completo o parser terminava a execução inesperadamente com algum erro que não conseguimos identificar.

Para o teste usamos a seguinte linha de comando:

```
./TurboParser -test
-evaluate
-file_model=models/model_trained
-file_test=d
-file_prediction=data/danish/danish_test.conll.predicted
-logtostderr
```

4.1.2 Treino e Teste MSTParser

Para treino e teste do MSTParser precisamos ficar atentos às *flags* das linhas de comando. Como o MSTParser aceita outro padrão de córpus, é necessário definir o CoNLL

como o formato padrão. Além disso, escolhemos 10 iterações para o treino do *parser* uma vez que o TurboParser realiza o mesmo número de iterações por padrão.

```
java -classpath ".:lib/trove.jar-Xmx1800m mstparser.DependencyParser
train train-file:train.txt model model-name:model.name format:CONLL iters:10
```

```
java -classpath ".:lib/trove.jar-Xmx1800m mstparser.DependencyParser
test model-name:dep.model test-file:test.txt output-file:out.txt format:MST
```

4.1.3 Treino e Teste MaltParser

Para o Malt Parser usamos apenas as configurações padrões de treinamento e teste:

```
java -jar maltparser-1.8.1.jar -c model -i train_file -m learn
```

```
java -jar maltparser-1.8.1.jar -c model -i test_file -o out_file -m parse
```

4.1.4 Treino e Teste Yara Parser

Para o YaraParser colocamos a flag definida com 10 iterações.

```
java -jar jar/YaraParser.jar train -train-file [train-file]
-model [model-file] iter:10
```

```
java -jar jar/YaraParser.jar parse_conll -input [test-file]
-out [output-file] -model [model-file]
```

4.2 Métodos de avaliação

Para avaliar os parsers por dependência usamos o EVALB07. Esse avaliador é amplamente usado nas *shared tasks*, como mencionamos na introdução desse trabalho.

O avaliador consiste em um algoritmo desenvolvido em Ruby e faz uma medição de quanto o parser acerta cada palavra das sentenças, e o resultado é visto como uma porcentagem de acertos.

O algoritmo usa então as métricas LAS (Label Attachment Score), UAS (Unlabeled Attachment Score), LA (Label Accuracy). LAS significa a porcentagem de acertos em relação ao arquivo de saída que o parser tem onde o rótulo da palavra (o que a palavra é sintaticamente) e o arco estão corretos. UAS significa o quanto o parser acertou em relação ao arco, no caso, se ele acertou a dependência. LA significa apenas o quanto o parser acertou o rótulo da palavra.

5 Experimentos e Resultados

5.1 Experimento com Parsers por Dependência

A seguir vamos explicar como foram os experimentos e os resultados obtidos dos testes e avaliações que realizamos.

5.1.1 Experimentos com o Tamanho dos Córpus Originais

O primeiro teste e avaliação que realizamos consistiu em testar cada parser com as informações originais dos córpus. Isto quer dizer que cada língua possuía um número de sentenças diferentes umas das outras.

As tabelas 2, 3, 4, 5 mostram os resultados obtidos com as avaliações do TurboParser, MSTParser, MaltParser e YaraParser, respectivamente.

Neste caso, não poderíamos realizar uma comparação entre as línguas: analisar qual língua o parser consegue analisar melhor, por exemplo. Isto porque uma vez que cada língua possui um número de sentenças diferentes, a análise se torna incerta. Se um parser recebe mais dados para aprender do corpus alemão e menos do português por exemplo, o parser receberá mais informações e irá gerar uma gramática maior (modelo) em comparação ao seu modelo gerado para o português.

Tabela 2 – Resultados com os Córpus com os Tamanhos Originais: TurboParser (em porcentagem de acertos)

	Sem Pontuação			Com Pontuação		
Idioma	LAS	UAS	LA	LAS	UAS	LA
Alemão	80,45	86,35	88,03	78,95	84,06	89,64
Coreano	88,84	93,12	93,01	89,02	83,23	93,12
Dinamarquês	85,41	90,30	89,78	84,74	88,93	91,25
Espanhol	82,65	86,32	89,26	81,01	84,29	90,38
Francês	81,26	85,60	87,80	79,60	83,38	89,37
Hindi	79,21	84,64	87,80	77,72	82,34	89,60
Inglês	90,48	91,90	94,77	88,98	90,23	95,35
Italiano	85,78	88,55	92,41	83,56	86,25	92,28
Japonês	74,21	80,92	88,28	76,88	82,86	89,53
Neerlandês	80,89	83,95	85,59	82,51	85,34	86,86
Português	90,24	94,30	91,76	90,05	93,48	93,05
Português BR	87,18	88,86	93,49	85,98	87,43	94,38
Sueco	83,24	87,21	89,57	82,26	85,83	90,61

Analisando os dados de avaliação podemos então fazer uma comparação entre os *parsers* tomando cada cada língua como estudo de caso. Cruzamos então todos os dados verificando qual o *parser* teve o melhor desempenho em cada etapa da avaliação. Por exemplo: analisando o alemão, qual parser obteve o melhor desempenho analisando a língua sem pontuação, com pontuação ou qual obteve o melhor LA.

Tabela 3 – Resultados com os Córpus com os Tamanhos Originais: MSTParser (em porcentagem de acertos)

	Sem Pontuação			Com Pontuação		
Idioma	LAS	UAS	LA	LAS	UAS	LA
Alemão	77,81	85,09	85,39	76,78	83,07	87,37
Coreano	87,14	92,31	91,58	87,34	92,44	91,71
Dinamarquês	83,09	89,20	88,38	82,89	88,12	90,04
Espanhol	80,81	85,86	87,05	79,24	83,75	88,43
Francês	79,29	84,75	86,01	77,47	82,23	87,80
Híndi	77,21	83,80	85,81	75,87	81,49	87,90
Inglês	89,87	92,11	93,79	89,13	91,11	94,49
Italiano	83,89	87,57	90,65	81,75	85,38	90,98
Japonês	77,01	84,54	89,26	79,42	86,12	90,40
Neerlandês	72,75	77,43	79,87	75,42	79,68	81,83
Português	86,06	92,02	88,85	86,63	91,66	90,59
Português BR	85,56	88,20	91,69	84,44	86,71	92,83
Sueco	81,36	86,91	87,45	80,83	85,82	88,71

Tabela 4 – Resultados com os Córpus com os Tamanhos Originais: MaltParser (em porcentagem de acertos)

	Sem Pontuação			Com Pontuação		
Idioma	LAS	UAS	LA	LAS	UAS	LA
Alemão	77,09	83,60	85,23	75,29	80,92	86,02
Coreano	82,04	88,45	88,61	82,33	88,64	88,79
Dinamarquês	80,92	86,47	86,09	80,16	84,91	87,03
Espanhol	80,23	84,43	86,92	78,75	82,53	87,84
Francês	78,60	83,84	84,89	76,40	80,96	85,74
Hindi	78,31	83,61	87,25	76,73	81,25	88,84
Inglês	88,58	90,37	93,17	87,35	89,02	93,61
Italiano	83,25	86,25	90,48	80,90	83,85	90,44
Japonês	76,48	83,85	89,38	78,95	85,51	90,47
Neerlandês	68,89	72,33	73,65	71,94	75,09	76,24
Português	84,16	90,87	86,06	83,32	88,98	85,99
Português BR	84,32	86,64	90,78	83,01	85,03	91,70
Sueco	81,58	85,99	88,15	80,90	84,88	89,02

Tabela 5 – Resultados com os Córpus com os Tamanhos Originais: YaraParser (em porcentagem de acertos)

	Sem Pontuação			Com Pontuação		
Idioma	LAS	UAS	LA	LAS	UAS	LA
Alemão	78,22	83,52	87,39	76,98	81,56	89,08
Coreano	88,34	92,97	92,28	88,52	93,08	92,40
Dinamarquês	81,92	87,35	87,01	82,26	86,91	88,88
Espanhol	83,02	86,43	89,75	81,60	84,67	90,83
Francês	82,12	86,54	88,45	80,78	84,62	89,93
Hindi	80,56	86,00	89,19	79,92	84,55	90,78
Inglês	91,84	93,34	95,72	91,25	92,58	96,19
Italiano	85,13	87,86	92,10	83,48	86,12	92,40
Japonês	69,04	79,25	82,99	72,28	81,45	84,75
Neerlandês	74,09	78,55	81,45	76,60	80,64	83,22
Português	83,27	92,52	86,19	84,81	92,62	88,34
Português BR	86,97	88,61	93,25	86,31	87,73	94,18
Sueco	85,12	88,70	91,09	84,74	87,96	91,97

Na tabela 6, podemos verificar os resultados. Decidimos mostrar o melhor desempenho de uma língua em cada aspecto para realizar uma análise mais minuciosa sobre qual parte o parser poderia ter melhor desempenho. Para a nossa surpresa, em muitos casos, um parser tinha o melhor desempenho para todos os quesitos. Por exemplo: TurboParser foi considerado o melhor na tarefa analisando o alemão acertando todos os quesitos:

LAS, UAS, LA tanto na análise com pontuação, quanto na sem pontuação. Interessante observar também que isso se repetiu várias vezes, pois TurboParser e YaraParser foram considerados os melhores em todos os quesitos para várias línguas.

Tabela 6 – Comparação dos *parsers* para cada língua.

	Sem Pontuação			Com Pontuação		
Idioma	LAS	UAS	LA	LAS	UAS	LA
Alemão	Turbo	Turbo	Turbo	Turbo	Turbo	Turbo
Coreano	Turbo	Turbo	Turbo	Turbo	Yara	Turbo
Dinamarquês	Turbo	Turbo	Turbo	Turbo	Turbo	Turbo
Espanhol	Yara	Yara	Yara	Yara	Yara	Yara
Francês	Yara	Yara	Yara	Yara	Yara	Yara
Hindi	Yara	Yara	Yara	Yara	Yara	Yara
Inglês	Yara	Yara	Yara	Yara	Yara	Yara
Italiano	Turbo	Turbo	Turbo	Turbo	Turbo	Yara
Japonês	MST	MST	Malt	MST	MST	Malt
Neerlandês	Turbo	Turbo	Turbo	Turbo	Turbo	Turbo
Português	Turbo	Turbo	Turbo	Turbo	Turbo	Turbo
Português BR	Turbo	Turbo	Turbo	Yara	Yara	Turbo
Sueco	Yara	Turbo	Yara	Turbo	Yara	Yara

5.1.2 Experimentos com Córpus de Tamanho Fixo

Neste experimento testamos todos os *parsers* com tamanhos fixos a fim de que pudéssemos analisá-los separadamente. Com córpus de treino de 3000 palavras e córpus de teste de 300, analisamos os melhores e piores desempenhos dos *parsers* na análise das línguas.

A tabela 7 mostra os resultados para os testes dos córpus no Turbo Parser. Podemos verificar que os melhores desempenhos foram das análises do português e as piores foram para o córpus neerlandês.

Tabela 7 – Resultados com os Córpus de Tamanhos Fixos: TurboParser (em porcentagem de acertos)

	Sem Pontuação			Com Pontuação		
Idioma	LAS	UAS	LA	LAS	UAS	LA
Alemão	80,50	85,42	88,81	80,22	84,48	90,21
Coreano	79,99	86,60	87,91	80,31	86,81	88,10
Dinamarquês	84,06	89,28	88,87	83,30	**87,79**	90,44
Espanhol	79,08	83,43	86,35	77,40	81,27	87,79
Francês	78,93	83,78	86,11	77,28	81,51	87,87
Hindi	79,23	84,81	88,28	77,74	82,47	90,07
Inglês	85,58	87,42	91,89	**84,19**	85,81	92,77
Italiano	83,05	86,32	90,34	80,61	83,81	90,24
Japonês	70,30	77,83	85,92	73,37	80,11	87,36
Neerlandês	**69,38**	**73,44**	**74,88**	**72,06**	**75,78**	**77,16**
Português	**85,85**	**89,74**	89,54	83,25	86,57	91,07
Português BR	85,29	87,32	**91,99**	83,76	85,51	**93,08**
Sueco	80,97	85,91	87,6	80,02	84,45	88,91

A tabela 8 mostra as análises para o MSTParser. Verificamos que o idioma com os melhores desempenhos foi o inglês e novamente o pior desempenho foi para o córpus neerlandês.

Tabela 8 – Resultados com os Córpus de Tamanhos Fixos: MSTParser (em porcentagem de acertos)

	Sem Pontuação			Com Pontuação		
Idioma	LAS	UAS	LA	LAS	UAS	LA
Alemão	78,11	84,13	86,88	77,56	82,84	88,50
Coreano	78,99	86,13	86,87	79,32	86,35	87,08
Dinamarquês	81,73	**88,28**	86,69	81,34	86,96	88,55
Espanhol	77,85	83,41	84,92	76,12	81,08	86,52
Francês	76,91	83,10	84,19	74,95	80,36	86,22
Hindi	77,67	84,06	86,53	76,32	81,73	88,58
Inglês	**85,06**	88,11	90,90	**84,53**	**87,21**	91,89
Italiano	81,51	85,78	88,50	79,06	83,25	89,04
Japonês	75,10	83,60	87,84	77,68	85,25	89,13
Neerlandês	**64,76**	**70,69**	**72,23**	**68,07**	**73,47**	**74,93**
Português	82,29	88,00	86,50	80,25	85,12	88,48
Português BR	84,78	87,63	**91,25**	83,85	86,33	**92,39**
Sueco	79,06	85,74	85,83	78,75	84,73	87,32

A tabela 9 mostra os resultados para o MaltParser. Os melhores índices foram para o inglês e os piores para córpus neerlandês.

Tabela 9 – Resultados com os Córpus de Tamanhos Fixos: MaltParser (em porcentagem de acertos)

	Sem Pontuação			Com Pontuação		
Idioma	LAS	UAS	LA	LAS	UAS	LA
Alemão	77,64	83,25	86,47	76,78	81,65	86,64
Coreano	74,97	83,43	83,82	75,37	83,69	84,07
Dinamarquês	78,98	85,12	84,62	78,36	83,64	85,84
Espanhol	77,77	82,88	84,96	76,24	80,83	86,16
Francês	76,49	82,20	83,12	74,06	79,05	83,99
Hindi	78,21	83,73	87,15	76,56	81,24	88,76
Inglês	**83,74**	85,94	**90,29**	**82,63**	84,66	**91,10**
Italiano	80,30	84,10	88,46	77,47	81,24	88,50
Japonês	74,41	82,87	87,63	77,10	84,64	88,95
Neerlandês	**59,77**	**65,53**	**65,99**	**63,69**	**68,90**	**69,32**
Português	82,89	**90,37**	84,28	81,93	**88,24**	84,39
Português BR	81,29	84,30	89,35	80,17	82,24	90,38
Sueco	79,70	84,69	87,12	78,96	83,44	88,29

A tabela 10 mostra os resultados para o Yara Parser. Verificamos que os melhores desempenhos foram para o córpus de língua inglesa e os piores para córpus neerlandês.

Tabela 10 – Resultados com os Córpus de Tamanhos Fixos: YaraParser (em porcentagem de acertos)

	Sem Pontuação			Com Pontuação		
Idioma	LAS	UAS	LA	LAS	UAS	LA
Alemão	78,41	83,33	88,20	77,99	82,27	89,64
Coreano	76,86	84,70	84,20	77,23	84,95	84,45
Dinamarquês	79,74	85,58	85,43	80,18	85,19	87,47
Espanhol	78,12	82,67	85,84	76,78	80,83	87,32
Francês	78,84	83,98	86,29	77,31	81,77	88,03
Hindi	80,43	86,01	89,31	79,41	84,13	90,94
Inglês	**86,49**	**88,94**	**92,41**	**85,91**	**88,06**	**93,23**
Italiano	81,73	85,14	89,75	79,64	83,04	90,10
Japonês	66,88	78,03	81,08	70,33	80,25	83,04
Neerlandês	**63,37**	**68,12**	**71,00**	**66,83**	**71,14**	**73,79**
Português	81,79	87,60	86,62	79,61	84,57	88,56
Português BR	84,94	87,13	91,62	83,94	85,85	92,69
Sueco	83,47	87,29	90,25	83,35	86,77	91,26

5.1.3 O caso do Corpus Neerlandês

Após os testes anteriores verificamos que o as análises para a língua neerlandesa apresentava sempre índices muito baixos de acerto, ou seja, os quatro *parsers* tinham um nível de acertos inferior se compararmos com outros idiomas. A primeira coisa que pensamos em dizer é o fato da língua neerlandesa possuir uma estrutura muito diferente de línguas como português e inglês. Porém, se não executarmos uma investigação mais profunda é muito difícil fazer tal afirmação. Decidimos então procurar outras versões de córpus para a língua no formato CoNLL, pois o córpus que obtivemos originalmente parecia estar com algum problema. A maioria dos córpus por dependência são originados a partir de córpus por constituência através de algoritmos que fazem a conversão de estrutura. Caso não encontrássemos outros dados de treino e teste para a língua neerlandesa, usaríamos ferramentes para a conversão do mesmo.

Felizmente isso não foi preciso pois encontramos uma outra versão do corpus neerlandês, originalmente vindo do córpus por constituência Alpino. [1]

Da mesma forma como fizemos com os dados anteriores, o córpus de treino ficou então com 3000 sentenças e o córpus de teste com 300. Esses novos dados de treino foram testados nos *parsers* Malt Parser e MSTParser e foram comparados com os resultados do córpous neerlandês antigo. Analisando as tabelas 11 e 12 podemos verificar um ganho de desempenho com novo córpus.

Um corpus com má anotação pode ser um problema, e não necessariamente a língua. Erros podem ocasionar análises muito ruins. Além disso, córpus por dependência, como já explicado, sofrem o processo de conversão de estruturas por constituência. Tais conversões podem acarretar em problemas no córpus.

Tabela 11 – Trocando o Córpus no MaltParser (em porcentagem de acertos)

	Sem Pontuação			Com Pontuação		
Idioma	LAS	UAS	LA	LAS	UAS	LA
Alpino	59,77	65,53	65,99	63,69	68,90	69,32
Alpino2	72,25	74,95	76,29	75,23	77,63	78,83

Tabela 12 – Trocando o Córpus no MSTParser (em porcentagem de acertos)

	Sem Pontuação			Com Pontuação		
Idioma	LAS	UAS	LA	LAS	UAS	LA
Alpino	64,76	70,69	72,23	68,07	73,47	74,93
Alpino2	77,25	80,45	83,50	74,53	78,11	81,54

5.1.4 Parser Projetivo e Não Projetivo: MSTParser

O MSTParser realiza, por padrão, análises projetivas nos córpus. Isso significa que, se um córpus por dependência possui um cruzamento de arcos, a análise do *parser*

[1] http://www.let.rug.nl/bplank/alpino2conll/

irá ignorá-lo. Para testarmos um melhor desempenho com o MSTParser, verificamos que era possível trocarmos a *flag* de projetividade ou não projetividade na linha de comando. A partir disso adicionamos a flag decode-type:non-proj para realizarmos análises não projetivas.

A tabela 13 mostra a comparação de resultados para análises projetivas e não projetivas. Podemos verificar que para algumas línguas a análise não projetiva era a melhor e para outras a projetiva apresentava melhores resultados. Mas apesar disto, a projetividade e a não projetividade não apresentavam resultados tão diferentes.

Tabela 13 – Comparando Projetivo e Não projetivo MSTParser (em porcentagem de acertos)

		Sem Pontuação			Com Pontuação		
		LAS	UAS	LA	LAS	UAS	LA
Alemão	Proj	78,11	84,13	86,88	77,56	82,84	88,5
	NProj	78,30	84,38	87,18	77,85	83,12	88,8
Coreano	Proj	78,99	86,13	86,87	79,32	86,35	87,08
	NProj	81,73	88,55	86,87	80,81	86,76	88,72
Dinamarquês	Proj	81,73	88,28	86,69	81,34	86,96	88,55
	NProj	77,16	82,79	84,4	75,23	80,25	86,03
Espanhol	Proj	77,85	83,41	84,92	76,12	81,08	86,52
	NProj	76,83	82,85	84,19	74,66	79,93	86,20
Francês	Proj	76,91	83,1	84,19	74,95	80,36	86,22
	NProj	76,74	83,19	85,89	75,07	80,54	88,04
Hindi	Proj	77,67	84,06	86,53	76,32	81,73	88,58
	NProj	84,34	87,04	90,59	83,17	85,56	91,61
Inglês	Proj	85,06	88,11	90,9	84,53	87,21	91,89
	NProj	84,34	87,04	90,59	83,17	85,56	91,61
Italiano	Proj	81,51	85,78	88,5	79,06	83,25	89,04
	NProj	81,08	85,19	88,67	78,37	82,4	89,14
Japonês	Proj	75,1	83,6	87,84	77,68	85,25	89,13
	NProj	75,06	83,6	87,79	77,68	85,29	89,13
Neerlandês	Proj	64,76	70,69	72,23	68,07	73,47	74,93
	NProj	67,07	72,59	73,72	70,86	75,09	76,26
Português	Proj	82,29	88,00	86,5	80,25	85,12	88,48
	NProj	82,47	88,28	86,68	79,97	84,93	88,63
Português BR	Proj	84,78	87,63	91,25	83,85	86,33	92,39
	NProj	84,16	87,00	90,89	82,85	85,34	92,07
Sueco	Proj	79,06	85,74	85,83	78,75	84,73	87,32
	Nproj	78,32	84,50	85,17	77,77	83,27	86,77

5.1.5 Dependências Projetivas: MaltParser

O MaltParser realiza análises exclusivamente projetivas. Verificamos então que é possível modificar o córpus a fim de mudarmos essa situação. Este *parser* oferece uma *flag* em que podemos tirar a projeção do córpus. Usamos a seguinte linha para realizar esse processo:

```
java -jar maltparser-1.8.1.jar -c pproj -m proj -i corpus
-o projectivized.conll -pp baseline
```

Por padrão, os córpus originalmente possuíam algumas dependências não projetivas. Usando a linha de comando, conseguimos eliminar tais dependências deixando-as projetivas.

Podemos verificar as figuras 3 e 4 uma pequena amostra do córpus para português brasileiro.

Verificamos aqui as dependências projetivas. A linha 7 se relaciona com a palavra da linha 12, na primeira figura. Usando o MaltParser a depedência muda. Isso acontece pois a relação 7 para 12 ocasionaria um cruzamento nos arcos das outras dependências, criando uma dependência não projetiva e o que o MaltParser faz é mudar a dependência.

Figura 3 – Córpus de Treino CoNLL para o Português original

```
1     Apesar       _     ADV   ADV   _     2     mwe    _     _
2     do     _     ADP   ADP   _     8     adpmod       _     _
3     histórico    _     NOUN  NOUN  _     2     adpobj       _     _
4     ,      _     .     .     _     2     p      _     _
5     Tenório      _     NOUN  PNOUN _     8     nsubj  _     _
6     não    _     ADV   ADV   _     8     neg    _     _
7     se     _     PRON  PRON  _     12    nsubj  _     _
8     acha   _     VERB  VERB  _     0     ROOT   _     _
9     "      _     .     .     _     10    p      _     _
10    a      _     DET   DET   _     12    det    _     _
11    "      _     .     .     _     10    p      _     _
12    referência   _     NOUN  NOUN  _     8     xcomp  _     _
13    .      _     .     .     _     8     p      _     _
```

Fonte: (MCDONALD et al., 2013)

Figura 4 – Córpus de Treino CoNLL para o Português com a mudança de dependência

```
1     Apesar       _     ADV   ADV   _     2     mwe    _     _
2     do     _     ADP   ADP   _     8     adpmod       _     _
3     histórico    _     NOUN  NOUN  _     2     adpobj       _     _
4     ,      _     .     .     _     2     p      _     _
5     Tenório      _     NOUN  PNOUN _     8     nsubj  _     _
6     não    _     ADV   ADV   _     8     neg    _     _
7     se     _     PRON  PRON  _     8     nsubj  _     _
8     acha   _     VERB  VERB  _     0     ROOT   _     _
9     "      _     .     .     _     10    p      _     _
10    a      _     DET   DET   _     12    det    _     _
11    "      _     .     .     _     10    p      _     _
12    referência   _     NOUN  NOUN  _     8     xcomp  _     _
13    .      _     .     .     _     8     p      _     _
```

Fonte: (MCDONALD et al., 2013)

Como havíamos realizado testes com o MaltParser com os córpus originalmente não projetivos, repetimos os testes mas agora com os córpus projetivos.

Na tabela 14 podemos verificar uma comparação entre os resultados dos acertos. A tabela mostra os testes realizados com o MaltParser (que usa análises projetivas) em córpus não projetivos (originalmente) e projetivos (modificados).

Tabela 14 – Comparando Dependências Projetivas e Não Projetivas MaltParser (em porcentagem de acertos)

		Sem Pontuação			Com Pontuação		
		LAS	UAS	LA	LAS	UAS	LA
Alemão	NProj	77,64	83,25	86,47	76,78	81,65	86,64
	Proj	77,59	83,39	87,38	77,44	82,53	88,61
Coreano	NProj	74,97	83,43	83,82	75,37	83,69	84,07
	Proj	74,97	83,35	83,39	75,37	83,62	84,15
Dinamarquês	NProj	78,98	85,12	84,62	78,36	83,64	85,84
	Proj	79,37	86,34	84,97	78,94	84,93	87,08
Espanhol	NProj	77,77	82,88	84,96	76,24	80,83	86,16
	Proj	78,02	83,14	85,16	76,76	81,39	86,67
Francês	NProj	76,49	82,20	83,12	74,06	79,05	83,99
	Proj	76,69	82,62	83,74	74,89	80,13	85,50
Hindi	NProj	78,21	83,73	87,15	76,56	81,24	88,76
	Proj	78,17	83,77	87,09	76,64	81,39	88,91
Inglês	NProj	83,74	85,94	90,29	82,63	84,66	91,10
	Proj	83,74	85,94	90,29	82,63	84,66	91,10
Italiano	NProj	80,3	84,10	88,46	77,47	81,24	88,50
	Proj	80,4	84,35	88,50	77,83	81,72	88,91
Japonês	NProj	74,41	82,87	87,63	77,10	84,64	88,95
	Proj	74,41	82,87	87,63	77,10	84,64	88,95
Neerlandês	NProj	59,77	65,53	65,99	63,69	68,90	69,32
	Proj	61,93	68,46	68,30	65,63	71,51	71,40
Português	NProj	82,89	90,37	84,28	81,93	88,24	84,39
	Proj	83,11	88,48	87,72	80,5	85,12	89,33
Português BR	NProj	81,29	84,30	89,35	80,17	82,24	90,38
	Proj	81,87	84,44	89,22	80,18	82,44	90,57
Sueco	NProj	79,70	84,69	87,12	78,96	83,44	88,29
	Proj	79,35	84,69	86,74	78,85	83,67	88,00

5.2 Experimentos com os Parsers por Constituência

Devido a falta de recursos para parsers por constituência, conseguimos apenas realizar poucos testes. A maioria dos testes não é possível pois a maioria dos córpus possuem formatos diferentes ou precisam de licença ou são pagos.

5.2.1 Treinos e testes no Stanford Parser

Este Parser possui dentro de suas aplicações modelos treinados para o inglês, porém muitas pesquisas em tornos de outros idiomas estão sendo mostrados, como por exemplo seus modelos treinados para outras línguas como árabe e francês. A partir disso pensamos em uma maneira de tentarmos incluir o português como um desafio para o treino do parser. Para esta tarefa usamos o córpus ThychoBrahe para treino e teste. Após esta tarefa, realizamos a avaliação do *parser* para o português mas os resultados foram muito abaixo do esperado. Comparados com o inglês ou o francês por exemplo, verificamos uma anomalia no treino das árvores do córpus de treino. Acreditamos que o StanfordParser ainda não consegue processar direito as informações para a língua portuguesa.

5.2.2 Treinos e testes no Berkeley Parser

Para o Berkeley Parser usamos então o córpus ThychoBrahe para treino e teste. Após isso, rodamos o WJS no parser para fins de comparação entre as duas línguas.

Tentamos usar o Córpus Islandês (IcePaHC) para treino e teste também. Apesar do treino ter sido possível, a saída e a avaliação que fizemos dentro do próprio parser, apontavam métricas muito baixas. Provavelmente é pelo fato de que ainda o treino ou teste não foram bem executados, mas até então não conseguimos achar uma forma para mudar isso. Pela falta de outros córpus colocamos aqui apenas uma observação que não conseguimos realizar maiores experimentos.

6 Conclusões Finais

Conseguimos, portanto, realizar vários estudos de *parsers* devido a ampla disponibilização dos córpus CoNLL. Conseguimos comparar textitparsers por dependência e testá-los em vários idiomas. Além disso, foi possível verificar a diferença (mesmo que pequena) de resultados para um *parser* que usa o modo projetivo e não projetivo. Da mesma forma que foi possível testar *parsers* projetivos com córpus modificados para possuírem apenas dependências projetivas. Infelizmente os testes com os *parsers* por constituência não foram muito bem sucedidos. Acreditamos que isso aconteça pelo fato de que seus dados de treinamento e testes não são muito disponibilizados e seus formatos não possuem uma padronização. É interessante observar que o ConNLL Shared Tasks é um evento que realmente vem fazendo a diferença, e que seria interessante que parsers por constituência pudessem ser inclusos em tais eventos. Temos em mente que isso poderia ajudar em uma maior divulgação dos córpus criando possibilidades maiores para análises mais profundas desses parsers.

Sobre as execuções e avaliações dos *parsers*, verificamos que o TurboParser foi um dos que mais se destacaram em algumas tarefas. Verificamos que sua metodologia não projetiva pode auxiliar línguas em que a ordem dos constituintes podem variar bastante. O YaraParser também se mostrou bastante avançado em termos de análise sintática. Apesar de suas análises serem de natureza projetiva em córpus originalmente não projetivos, apresentou um *parser* um alto desempenho comparado com os outros *parsers* usados neste trabalho.

Já o MST Parser se mostrou muito eficiente para a língua japonesa quando usados os córpus originais (número original de sentenças). Entretando, verificando suas análises projetivas e não projetivas, podemos concluir que nem sempre a mudança do tipo de análise ocasionará uma melhora significativa de desempenho. Algumas línguas mudam pouco com essa mudança e algumas nem mostram grandes diferenças, como o coreano e o japonês. Da mesma forma, trocar o córpus de não projetivo para projetivo no MaltParser não irá criar grandes variações nos resultados. Analisando mais profundamente os testes relacionados à projeção, pode-se verificar um grande esforço atualmente na adaptação e desenvolvimento de parsers multilíngue. E isso só é possível graças ao estudo mais profundo de como as línguas se relacionam ou como funcionam suas regras, pois muitos *parsers* no seu início apenas eram desenvolvidos para análise da língua inglesa. Esperamos, portanto, que este trabalho possa auxiliar nas próximas tentativas de execução de *parsers* e avaliação dos mesmos.

Podemos identificar problemas em conseguir realizar análises mais amplas para

parsers por constituência devido a falta de dados. Por outro lado, os *parsers* por dependência mostram-se dispostos a evoluir e adotar cada vez mais um padrão único de córpus. Além disso esperamos realizar mais análises no que diz respeito às relações projetivas e não projetivas, pois acreditamos que isso possa melhorar o desempenho dos *parsers* em diferentes línguas.

Referências

BOSCO, C. et al. Comparing italian parsers on a common treebank: the evalita experience. In: *LREC*. [S.l.: s.n.], 2008. Citado na página 29.

BOSCO, C. et al. Evalita'09 parsing task: comparing dependency parsers and treebanks. *Proceedings of EVALITA*, v. 9, 2009. Citado na página 29.

BOSCO, C. et al. Comparing the influence of different treebank annotations on dependency parsing. In: CHAIR), N. C. C. et al. (Ed.). *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. Valletta, Malta: European Language Resources Association (ELRA), 2010. ISBN 2-9517408-6-7. Citado na página 29.

BRAGA, A. de P.; FERREIRA, A. C. P. de L.; LUDERMIR, T. B. *Redes neurais artificiais: teoria e aplicações, 2ª Edição*. [S.l.]: LTC Editora, 2007. Citado na página 27.

BUCHHOLZ, S.; MARSI, E. Conll-x shared task on multilingual dependency parsing. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the Tenth Conference on Computational Natural Language Learning*. [S.l.], 2006. p. 149–164. Citado na página 21.

CARROLL, J.; BRISCOE, T.; SANFILIPPO, A. Parser evaluation: a survey and a new proposal. In: *Proceedings of the 1st International Conference on Language Resources and Evaluation*. [S.l.: s.n.], 1998. p. 447–454. Citado na página 29.

CHOI, J. D.; TETREAULT, J.; STENT, A. It depends: Dependency parser comparison using a web-based evaluation tool. In: . [S.l.: s.n.], 2015. Citado na página 30.

DICK, G.; CERIEL, H. *Parsing Techniques, a Practical Guide*. [S.l.], 1990. Citado na página 23.

FRANCIS, W. N.; KUCERA, H. Brown corpus manual. *Brown University*, 1979. Citado na página 29.

GILDEA, D. Corpus variation and parser performance. In: *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*. [S.l.: s.n.], 2001. p. 167–202. Citado na página 29.

HAWKINS, J. A. *A performance theory of order and constituency*. [S.l.]: Cambridge University Press, 1994. Citado na página 25.

KAKKONEN, T. Robustness evaluation of two ccg, a pcfg and a link grammar parsers. *arXiv preprint arXiv:0801.3817*, 2008. Citado na página 29.

KUKKADAPU, P.; MALLADI, D. K.; DARA, A. Ensembling various dependency parsers: Adopting turbo parser for indian languages. In: *24th International Conference on Computational Linguistics*. [S.l.: s.n.], 2012. p. 179. Citado na página 30.

KUMMERFELD, J. K. et al. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. [S.l.], 2012. p. 1048–1059. Citado na página 29.

MARTINET, A. *Elementos de linguística geral*. [S.l.]: Livraria Sá da Costa, 1970. Citado na página 21.

MCDONALD, R. et al. Non-projective dependency parsing using spanning tree algorithms. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. [S.l.], 2005. p. 523–530. Citado na página 25.

MCDONALD, R. T. et al. Universal dependency annotation for multilingual parsing. In: CITESEER. *ACL (2)*. [S.l.], 2013. p. 92–97. Citado na página 41.

NILSSON, N. J. Introduction to machine learning. an early draft of a proposed textbook. Citeseer, 1996. Citado na página 27.

NIVRE, J. et al. Benchmarking of statistical dependency parsers for french. In: *The 23rd International Conference on Computational Linguistics (Coling 2010)*. [S.l.: s.n.], 2010. p. 833–841. Citado na página 29.

NLX, G. d. F. e. L. N. *LX DepParser*. 2015. Disponível em: <http://lxcenter.di.fc.ul.pt/services/pt/LXServicesParserDepPT.html>. Citado na página 25.

NLX, G. d. F. e. L. N. *LX Parser*. 2015. Disponível em: <http://lxcenter.di.fc.ul.pt/services/pt/LXParserPT.html>. Citado na página 24.

SEDDAH, D.; CANDITO, M.; CRABBÉ, B. Cross parser evaluation and tagset variation: A french treebank study. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 11th International Conference on Parsing Technologies*. [S.l.], 2009. p. 150–161. Citado na página 29.

Índice