Universidade Federal de São Carlos
Pablo Botton da Costa

Um analisador sintático neural multilíngue baseado em transições

São Carlos
2016

Pablo Botton da Costa

Um analisador sintático neural multilíngue baseado em transições

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Profa. Dra. Helena de Medeiros Caseli

Coorientador: Prof. Dr. Fabio Natanael Kepler

São Carlos

2016

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de Dissertação de Mestrado do candidato Pablo Botton da Costa, realizada em 24/01/2017.

Profª. Drª. Helena de Medeiros Caseli
(UFSCar)

Profª. Drª. Sandra Maria Aluísio
(USP)

Prof. Dr. Ivandré Paraboni
(USP)

Certifico que a sessão de defesa foi realizada com a participação à distância do membro Ivandré Paraboni, depois das arguições e deliberações realizadas, o participante à distância está de acordo com o conteúdo do parecer da comissão examinadora redigido no relatório de defesa do aluno Pablo Botton da Costa.

Profª. Drª. Helena de Medeiros Caseli
Presidente da Comissão Examinadora
(UFSCar)

"We are merely explorers of infinity in the pursuit of absolute perfection" (DESCONHECIDO)

Resumo

Um analisador sintático de dependência consiste em um modelo capaz de extrair a estrutura de dependência de uma sentença em língua natural. No Processamento de Linguagem Natural (PLN), os métodos multilíngues tem sido cada vez mais utilizados (BROWN et al., 1995; COHEN; DAS; SMITH, 2011), inclusive na tarefa de análise de dependência. Intuitivamente, um analisador sintático multilíngue pode ser visto como um vetor de analisadores sintáticos treinados individualmente em cada língua. Contudo, a tarefa realizada com base neste vetor torna-se inviável devido a sua alta demanda por recursos. Como alternativa, diversos métodos de análise sintática foram propostos (MCDONALD; PETROV; HALL, 2011; TACKSTROM; MCDONALD; USZKOREIT, 2012; TITOV; HENDERSON, 2007), mas todos dependentes de alinhamento entre palavras (TACKSTROM; MCDONALD; USZKOREIT, 2012) ou de técnicas de agrupamento, o que também aumenta a complexidade associada ao modelo (TSARFATY et al., 2013; BOHNET et al., 2013a). Uma solução simples surgiu recentemente com a construção de recursos universais (NIVRE et al., 2016a). Estes recursos universais têm o potencial de diminuir a complexidade associada à construção de um modelo multilíngue, uma vez que não é necessário um mapeamento entre as diferentes notações das línguas. Nesta linha, este trabalho apresenta um modelo para análise sintática universal de dependência: o NNParser. O modelo em questão é uma modificação da proposta de Chen e Manning (2014) com um modelo mais guloso e preciso na captura de representações distribuídas (MIKOLOV et al., 2011). Nos experimentos aqui apresentados o NNParser atingiu 93,08% de UAS para o inglês no córpus *Penn Treebank* e resultados melhores do que o estado da arte, o Stack LSTM, para o português (87,93% × 86,2% LAS) e o espanhol (86,95% × 85,7% LAS) no córpus UD 1.2.

Palavras-chave: Processamento de Linguagem Natural, Análise sintática de dependência, Análise sintática baseada em transições, Processamento multilíngue, Aprendizado Neural, Representação distribuída.

Abstract

A dependency parser consists in inducing a model that is capable of extracting the right dependency tree from an input natural language sentence. Nowadays, the multilingual techniques are being used more and more in Natural Language Processing (NLP) (BROWN et al., 1995; COHEN; DAS; SMITH, 2011), especially in the dependency parsing task. Intuitively, a multilingual parser can be seen as vector of different parsers, in which each one is individually trained on one language. However, this approach can be a really pain in the neck in terms of processing time and resources. As an alternative, many parsing techniques have been developed in order to solve this problem (MCDONALD; PETROV; HALL, 2011; TACKSTROM; MCDONALD; USZKOREIT, 2012; TITOV; HENDERSON, 2007) but all of them depends on word alignment (TACKSTROM; MCDONALD; USZKOREIT, 2012) or word clustering, which increases the complexity since it is difficult to induce alignments between words and syntactic resources (TSARFATY et al., 2013; BOHNET et al., 2013a). A simple solution proposed recently (NIVRE et al., 2016a) uses an universal annotated corpus in order to reduce the complexity associated with the construction of a multilingual parser. In this context, this work presents an universal model for dependency parsing: the NNParser. Our model is a modification of Chen e Manning (2014) with a more greedy and accurate model to capture distributional representations (MIKOLOV et al., 2011). The NNparser reached 93.08% UAS in English Penn Treebank (WSJ) and better results than the state of the art Stack LSTM parser for Portuguese (87.93% $\times$ 86.2% LAS) and Spanish (86.95% $\times$ 85.7% LAS) on the universal dependencies corpus.

Key-words: Dependency parser, Natural Language Processing, Multilingual Parsing, Neural Networks, Word Embeddings.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

O termo "análise sintática" refere-se à tarefa de analisar uma sentença em língua natural com o intuito de definir suas classes sintáticas e morfossintática[1]. Essa análise é realizada com base em um conjunto de regras bem definidas e estruturadas. Por exemplo, uma regra normalmente usada como a regra inicial de uma gramática é a que define uma sentença a partir de seus sintagmas nominal e verbal, possivelmente expressa como: S $\rightarrow$ SN SV. O arranjo desses símbolos define o funcionamento do analisador sintático.

Contudo, a simples utilização dessas regras não garante que o analisador sintático encontrará a solução menos ambígua. A ambiguidade inerente nas línguas naturais é um problema que pode ser tratado por meio da análise sintática probabilística. Seguindo tal estratégia, atribuem-se pesos às regras de modo que o analisador possa decidir qual a melhor regra a ser aplicada em um dado momento, lidando com a ambiguidade inerente de uma língua natural.

A análise sintática probabilística aplicada ao Processamento de Língua Natural (PLN) é uma tarefa que consiste em receber como entrada uma frase (ou sentença) e retornar as estruturas gramaticais que compõem a entrada, geralmente em forma de árvore. A realização desta tarefa pode se dar de duas formas: constituência ou dependência.

Na análise de constituência, intuitivamente associa-se a construção de classes sintáticas por meio da combinação de pedaços de sentenças (*chunks*). Então, neste âmbito, pode-se construir um decisor para classificar pedaços de sentença de forma iterativa e, assim, combiná-los a fim de obter a estrutura sintática da sentença.

Embora bastante investigada na literatura (SOCHER et al., 2013; COLLINS, 2003; KLEIN; MANNING, 2003), a análise de constituência não é o objeto de investigação neste trabalho. Aqui o objetivo é investigar a análise de dependência, na qual o foco está na dependência de uma palavra em relação a outra dentro de uma dada sentença. A esta relação de dependência entre palavras é atribuído um rótulo sintático que denota a classe da relação.

Existem duas formas de se realizar a análise de dependência: (1) baseada em grafos e (2) baseada em transições. Em ambas as estratégias, uma sentença é descrita como uma relação de dependentes e governantes (na forma de grafos). Um analisador

[1] Classes sintáticas e morfossintáticas são elementos que descrevem uma sentença. Estes elementos descrevem níveis mais básicos de uma sentença, uma palavra, ou níveis sofisticados como a interação que palavras têm umas com as outras. Para estas interações atribuem-se classes relativas ao tipo de interação. Por exemplo, na frase "o menino viu a menina" as palavras "menino" e "viu" recebem, respectivamente, as seguintes classes: "substantivo" e "verbo", no nível morfossintático; e "sintagma nominal" e "sintagma verbal" no nível sintático.

de dependência baseado em grafos tem o potencial de tomar horas de processamento uma vez que seu processamento baseia-se em tarefas custosas computacionalmente, como extração de características e construção do modelo de decisão (ambas tarefas que não são computadas em tempo linear).

Um *parser* de dependência baseado em transições, por outro lado, divide o problema de análise sintática em uma série de decisões sobre um conjunto inicial de configuração (*buffer* e pilha), tornando o modelo rápido e eficiente uma vez que o tempo de processamento está relacionado ao tamanho da entrada. Essas decisões são, então, combinadas de forma iterativa até um estado final e posterior extração de suas estruturas sintáticas (YAMADA; MATSUMOTO, 2003; NIVRE, 2003).

A análise de transições pode ser feita através de um classificador, que seja capaz de extrair características de uma configuração para definir quais operações serão feitas, em cada estado, a fim de obter-se o grafo final da sentença. Estas características podem ser: contextos das palavras, sub-grafos já construídos, bem como símbolos sintáticos e morfossintáticos destes elementos.

Nessa estratégia, como proposto em (NIVRE, 2003; HALL; NILSSON, 2006; ZHANG; NIVRE, 2011b), existem dois tipos possíveis de operações: a *arc-standard* e a *arc-eager*. Operações da forma *arc-standard* definem como certos nós do grafo serão conectados. Já a forma *arc-eager* caracteriza-se por decisões mais prolongadas. Estas decisões prolongadas preferencialmente realizam operações que expandam o espectro de busca ao invés de atribuir uma classe sintática de forma gulosa.

A combinação de um modelo de transição com o incremento de *features* que são extraídas a cada etapa do processo de análise sintática leva à construção de um modelo de decisão guloso. Esse tipo de modelo tem como característica a capacidade de manter o tempo de processamento linear com a sentença de entrada (NIVRE, 2003; NIVRE, 2014).

Idealmente, um *parser* deve funcionar para qualquer linguagem. Contudo, os *parsers* são construídos de forma específica para um determinado conjunto de dados, em uma língua específica, e estes recursos não são anotados de forma homogênica. Esse processo torna o modelo gerado específico para uma língua, o que dificulta a tarefa de *parsing* (TSARFATY et al., 2013).

Uma solução simples para este problema seria utilizar um vetor de *parsers*, cada um treinado especificamente em uma língua. Contudo, essa solução leva a um uso excessivo de recursos computacionais que não estão disponíveis. Para contornar este problema, este trabalho propõe a construção de um modelo de análise sintática multilíngue que combina recursos linguísticos (córpus) considerando suas similaridades linguísticas.

Diferentes línguas compartilham similaridades morfossintáticas/sintáticas e fonéticas (línguas irmãs). Combinando o poder da universalidade na anotação dos recursos

linguísticos Nivre et al. (2016b) propõem a construção de recursos homogêneos, isto é, que compartilham o mesmo conjunto de etiquetas (PETROV; DAS; MCDONALD, 2012) possibilitando, assim, a utilização destes recursos e facilitando a implementação de um modelo multilíngue.

Uma definição universal para o conjunto de etiquetas permite a utilização de recursos sem se preocupar especificadamente com cada língua. Habilitando, assim, o modelo a obter um desempenho médio para todas as línguas que compartilham desta homogeneidade.

Além de tudo o que foi exposto em relação à análise sintática, o PLN como um todo vem sofrendo diversos avanços recentes. Entre as revoluções importantes nos últimos 10 anos, destacam-se os paradigmas da representação distribuída e a transferência de linguagem.

A representação distribuída consiste em representar, de forma condensada, características distribuídas de uma palavra na forma de dados quantificados como vetores numéricos. Já a transferência de linguagem, é uma forma de aprendizado de máquina que caracteriza-se por criar um mapeamento entre diferentes espaços. Por exemplo, pode-se mapear etiquetas específicas em duas línguas desde que estas possuam alguma similaridade.

Uma forma interessante utilizada para representação distribuída de palavras é através do *deep learning*. O *deep learning* é uma área do aprendizado de máquina que possibilita o aprendizado de características (*features*) de forma automática. Ele é capaz de aprender representações cada vez mais complexas a partir de exemplos simples. Sua motivação está em extrair características de forma automática. Assim, através de exemplos simples, métodos de *deep learning* conseguem aprender tarefas complexas como: a análise sintática, o reconhecimento de voz e a identificação de objetos em imagens.

A próxima seção (1.1) traz o objetivo deste trabalho e as hipóteses nele investigadas. Em seguida, são apresentados os principais conceitos que norteiam este trabalho: representação distribuída (1.2), linguagens irmãs (1.3), transferência de linguagem (1.4), análise de dependência baseada em transições (1.5) e a diferenciação entre sentenças projetivas e não projetivas (1.6). Por fim, a última seção (1.7) traz a organização deste documento.

1.1 Objetivo e hipóteses

Este trabalho tem como objetivo:

"Desenvolver um *parser* de dependência multilíngue baseado em transições"

Para tanto, as hipóteses que norteiam este trabalho são:

1. A abordagem neural (*deep learning*) é uma boa estratégia para a geração do *parser* de dependência baseado em transições. Por meio de testes e comparações com diferentes córpus e modelos, pretende-se verificar qual é a melhor configuração de modelo neural para a geração de um *parser* de dependência;

2. O uso de línguas irmãs melhora o desempenho do *parser* de dependência baseado em transições. Dadas as similaridades fonéticas/sintáticas entre línguas irmãs, a combinação de linguagens irmãs oferece ao modelo um maior conjunto de dados de treinamento aumentando, assim, a assertividade do modelo;

3. A modelagem *one-hot* é uma boa estratégia para o *parser* de dependência baseado em transições. Uma modelagem *one-hot* será utilizada para modelar diferentes classes do *parser*. Este tipo de modelagem torna o modelo mais guloso uma vez que classes não vistas durante o processo de treinamento são penalizadas.

1.2 Representação distribuída

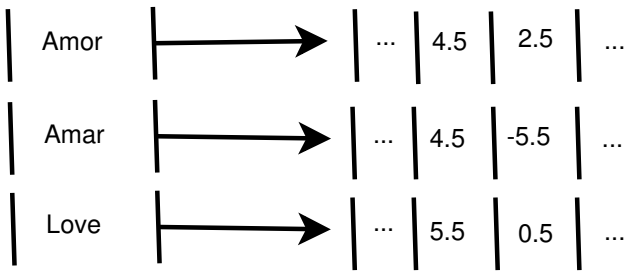
A ideia de representar uma palavra em vetores (normalmente referenciada como representação distribuída ou *word embeddings*) é algo que vêm se mostrando muito útil em diversas tarefas de PLN, como tradução automática (MIKOLOV; LE; SUTSKEVER, 2013) e *parsing* e *chunking* (SOCHER et al., 2013; CHEN; MANNING, 2014; COLLOBERT, 2011). Esta forma de representação tem como objetivo valer-se de técnicas não supervisionadas para encontrar representações vetoriais de objetos, isto é, pode-se representar diferentes objetos como: palavras (MIKOLOV et al., 2013), estados de *parsing* (CHEN; MANNING, 2014; DYER et al., 2015), etc.

Utilizando-se de uma representação em forma de vetor, como a ilustrada na Figura 1, uma palavra pode ser comparada com outra a fim de encontrar similaridades. O vetor de características possui valores de pesos positivos ou negativos. Essas representações vetoriais são aprendidas a partir de um córpus grande, geralmente na casa dos gigabytes.

Como pode-se perceber pela Figura 1, um vetor de palavras possui vários pesos. Esses vetores podem ser induzidos usando diversos algoritmos, dos quais os mais utilizados atualmente são (PENNINGTON; SOCHER; MANNING, 2014a) e (MIKOLOV et al., 2013).

Um dos maiores avanços em aprendizado de máquina para PLN é a capacidade de representar palavras. Uma forma simples de representar palavras é a utilização de n-gramas. Estes "n-gramas" tem a capacidade de representar uma palavra de um texto e seus diferentes contextos.

Figura 1 – Representação Neural distribuída da palavra Amor em inglês, espanhol e português.



Fonte: próprio autor

Por exemplo, uma sentença com três palavras "O menino viu a bola" pode ser representada por um n-grama de tamanho cinco. Este tipo de representação permite criar inferências probabilísticas a partir do texto, como "Qual o contexto mais provável para a palavra menino?": "? menino ?"; "Qual o terceiro elemento mais provável no tri-grama?": "O menino ? ? ?". Estas inferências podem ser facilmente extraídas com base no teorema de Bayes, o qual tem como objetivo encontrar a hipótese A dada a teoria B ou, traduzindo para PLN, a palavra (A) é representada pelo contexto (B), como ilusta a equação a seguir:

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B|A)\Pr(A) + \Pr(B|\neg A)\Pr(\neg A)} \tag{1.1}$$

Mikolov et al. (2013) apresentam um método simples para indução de representações distribuídas de palavras através de "n-gramas contínuos". Este algoritmo é capaz de extrair *features* de palavras que compõem um n-grama, o que pode ser feito de duas formas: (1) uma palavra em relação ao seu contexto (C-BOW) ou (2) a representação do contexto dada a palavra alvo (Skip-gram).

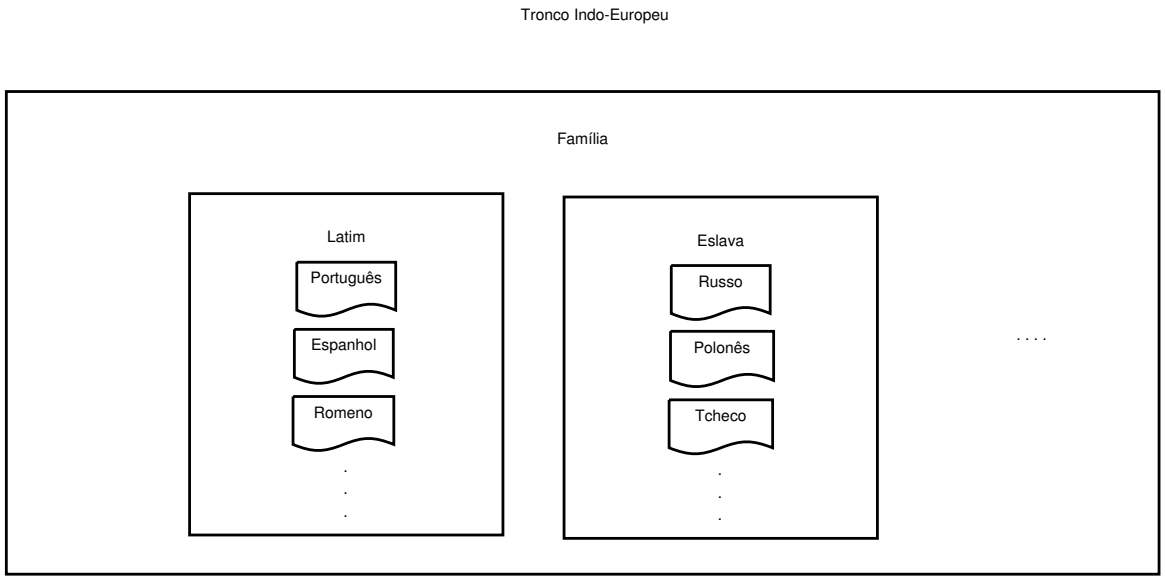
A forma mais utilizada para induzir representações distribuídas de palavras é o algoritmo C-BOW, que utiliza uma representação contínua de n-gramas, isto é, à medida que o texto é extraído pelo modelo desloca-se o n-grama alimentando, assim, uma rede neural que é capaz de extrair representações distribuídas dos diferentes n-gramas presentes no texto. O aprendizado destes n-gramas é feito através da modelagem de uma janela dinâmica (similar à construída pelo teorema de Bayes) e o aprendizado das características dos vetores é feito por meio da análise das variações das palavras em relação ao texto (*cross-entropy*).

A forma C-BOW foi utilizada neste trabalho.

1.3 Linguagens irmãs

Um par de línguas (ou linguagens) irmãs é um par de línguas que possuem características em comum, como o fato de pertencerem ao mesmo tronco linguístico (português e polonês, por exemplo) ou à mesma família (português e espanhol, por exemplo), como ilustrado na Figura 2. As similaridades entre línguas irmãs podem se dar em nível sintático, semântico ou lexical. O português e o espanhol possuem diversas palavras em comum (semelhança lexical), além de possuírem uma semelhança muito grande na construção de frases (semelhança sintática).

Figura 2 – Estrutura do Tronco Linguístico Indo-Europeu.



Fonte: próprio autor

Assim, neste trabalho considera-se como uma linguagem irmã aquela que: (i) possuir maior quantidade de semelhanças, (ii) estiver presente no mesmo tronco linguístico e (iii) possuir a mesma linguagem mãe em comum. Neste trabalho considera-se português e espanhol como línguas irmãs.

1.4 Transferência de linguagem

O método de transferência de linguagem investigado neste trabalho é o mesmo proposto em (GUO et al., 2015). Este método tem como objetivo induzir representações distribuídas multilíngues a partir de características distribuídas monolíngues. Estas representações são extraídas com base em similaridades presentes em dicionários bilíngues.

Guo et al. (2015) propõem uma forma muito simples de extrair e combinar representações. Por meio de um dicionário bilíngue, combinam-se representações distribuídas

através de correlações canônicas (FARUQUI; DYER, 2014; AMMAR et al., 2016b) para serem utilizadas na análise sintática. A correlação canônica (CCA) é um técnica capaz de combinar dois conjuntos de vetores com base em suas similaridades. Para palavras de línguas diferentes, uma forma bem intuitiva de encontrar uma correlação é utilizando dicionários bilíngues. Uma vez encontrada a correspondência dos conjuntos, o CCA combina ambas as representações das palavras em um novo espaço. Esta nova representação combinada contém características das duas línguas.

Métodos de transferência de linguagem e multilingualidade vêm sendo utilizados em diversas atividades do PLN (AMMAR et al., 2016b; FARUQUI; DYER, 2014; AMMAR et al., 2016a; GUO et al., 2015). Uma forma bem simples e intuitiva para análise sintática é a combinação de recursos com base em similaridade linguísticas. Nesse sentido, Petrov, Das e McDonald (2012) e Ammar et al. (2016a) propõem a utilização de um conjunto universal de etiquetas para simplificar a combinação de recursos. Dese modo, a tarefa de análise sintática é transformada na construção de um modelo universal.

Assim, este trabalho utilizou ambas as formas para realizar análise sintática multilíngue.

1.5 *Parser* baseado em transições vs. *parser* baseado em grafos

Como já mencionado anteriormente, existem duas formas de se realizar *parsing* de dependência: com base em transições ou com base em grafos. O *parser* baseado em transições é o mais rápido e guloso, no qual a inferência das operações é linear, o que permite combinar formas de busca para maximizar as escolhas de operações por estado, tendo um universo de escolhas mais amplo. O processamento da sentença se dá da esquerda para a direita, o que permite um histórico global das características, ao mesmo tempo que possibilita uma decisão local.

A forma baseada em grafos é uma forma mais lenta e exaustiva para busca de estados. Ela permite fatoração de *high-order*, o que é ótimo para extrair características locais em cada estado. Suas decisões são globais, uma vez que o processamento da sentença é dinâmico ou por aglutinação de estados. A "lentidão" está na fatoração dos estados. A fatoração dos estados é a aglutinação dos nós envolvidos em transições. Isto é, toda a dependência entre termos é mapeada e aglutinada a fim de extrair características que descrevam as dependências entre as palavras mapeadas, o que torna a tarefa não linear, uma vez que todas as aglutinações possíveis teriam que ser mapeadas.

Ambas as formas obtêm resultados similares em termos de acurácia (NIVRE, 2014). Contudo, considerando-se o processamento e a computação de *features*, o analisador baseado em transições tem a enorme vantagem de seu processamento se manter linear.

Neste trabalho, utilizou-se a forma de análise de dependência baseada em transições.

1.6 Sentenças projetivas vs. sentenças não projetivas

A forma de representação de uma sentença em análise de dependência é através de grafos. Estes grafos, por sua vez, estão divididos em duas classes: os grafos com projetividade e os grafos sem projetividade. A projetividade é inerente à sentença, isto é, o grafo é não projetivo quando há alguma relação "cruzada" originalmente presente na sentença que ele representa. Se o grafo for gerado sem nenhum cruzamento, então a sentença (e consequentemente o grafo) é dita projetiva.

Figura 3 – Exemplo de grafo não projetivo.

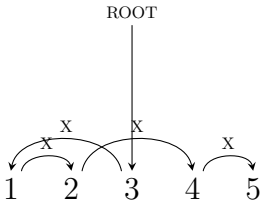
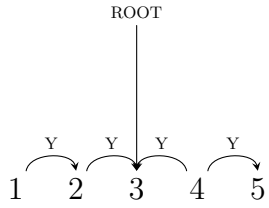


Figura 4 – Exemplo de grafo projetivo.



Fonte: próprio autor

Mais especificamente, a projetividade está relacionada à existência (ou não) de cruzamentos entre arcos do grafo. As sentenças das Figuras 3 (não projetiva) e 4 (projetiva) exemplificam essa diferença. Na Figura 3, entre a relação [1-2] e [3-1] existe um cruzamento de arcos caracterizando uma sentença não projetiva.

1.7 Organização deste documento

Este documento está organizado como segue. No capítulo 2 são apresentados os principais conceitos e a fundamentação teórica deste trabalho. Em seguida, no capítulo 3, apresenta-se a revisão do estado da arte para as técnicas de análise sintática multilíngue.

O capítulo 4 apresenta a arquitetura do modelo de transições aqui proposto: o NNParser. Mais especificamente, este capítulo apresenta a estrutura da rede neural para as tarefas de análise sintática mono e multilíngue, os recursos utilizados nos experimentos e as formas de avaliação do modelo.

O quinto capítulo descreve os experimentos realizados em análise sintática usando as diversas abordagens investigadas neste trabalho. Por fim, o sexto capítulo apresenta as considerações finais e algumas propostas de trabalhos futuros para o NNParser.

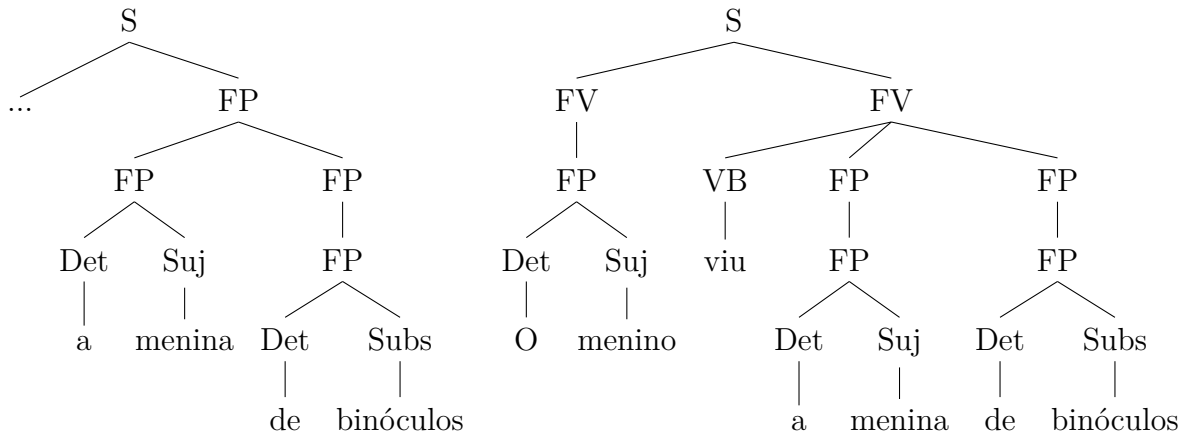
2 Fundamentação teórica

A análise sintática (*parsing*) consiste em, dada uma sentença, extrair a melhor árvore sintática para ela. Esta tarefa de extrair a melhor árvore sintática para uma sentença é executável a partir da construção de uma gramática livre de contexto (como a apresentada na Figura 6).

A gramática livre de contexto é um formalismo de representação para a ordem e o arranjo das palavras de uma língua, que é formalmente definida pela quádrupla $L = (V, \sum, R, S)$. Nesta quádrupla, V é o conjunto de símbolos não terminais e intermediários[1]; $\sum$ é o conjunto de terminais, que são a parte "final", isto é, palavras de uma língua natural por exemplo; R é o conjunto de produções da gramática, escrita na forma $\alpha \rightarrow \beta$ para os conjuntos $\alpha \in V$ e $\beta \in (V \cup \sum)$; e S é o símbolo inicial da gramática. Seguindo o que está especificado em uma gramática, o analisador sintático verifica as produções possíveis para uma certa entrada (sentença).

Por exemplo, considere a sentença em português apresentada a seguir "O menino viu a menina de binóculos". Esta sentença pode apresentar diversas árvores sintáticas, e isso caracterizaria ambiguidade gramatical (quando mais de uma árvore correta é possível). Como percebe-se pelos exemplos da Figura 5, a sentença original pode sofrer diversas interpretações gramaticais devido à ambiguidade presente na língua natural. Estas interpretações podem alterar o sentido de uma frase como percebe-se no exemplo, para o termo "binóculos" pode ser atribuído à "menina" ou ao "menino", sendo que ambas as interpretações são possíveis.

Figura 5 – Duas possíveis árvores geradas para a sentença "O menino viu a menina de binóculos".



Fonte: adaptado de (BECK, 2012)

[1] Cada símbolo não terminal representa uma sub-linguagem da linguagem definida por L.

Para a realização da análise sintática é necessária a construção de muitas regras, por vezes bastante complexas. Para facilitar a tarefa de análise sintática, Manning e Schütze (1999) propõem a abordagem de **indução gramatical** para extração de regras de uma gramática utilizando técnicas de aprendizado supervisionado, não supervisionado ou semi-supervisionado. Mas, mesmo com o aprendizado automático o problema da ambiguidade ainda persiste. Para tentar solucionar esse problema Manning e Schütze (1999) propõem a utilização de um analisador sintático probabilístico.

Um analisador sintático probabilístico tenta solucionar a ambiguidade de uma gramática atribuindo pesos às produções[2]. A ambiguidade induz o analisador a interpretar uma sentença erroneamente levando, assim, a sentença a ter mais de uma interpretação (derivação) possível, como por exemplo a sentença "o menino viu a menina de binóculos". Em termos gramaticais, há mais de uma interpretação possível para a mesma sentença. No caso do exemplo anterior é possível perceber que a ambiguidade está presente na gramática (veja Figura 6) já que a primeira e a segunda regras geram duas derivações diferentes.

Figura 6 – Gramática utilizada para a geração das derivações para a sentença "O menino viu a menina de binóculos".

$$S \rightarrow NPVBP$$
$$S \rightarrow NPVBPPP$$
$$NP \rightarrow DetN$$
$$NP \rightarrow NPPP$$
$$VBP \rightarrow PNP$$
$$PP \rightarrow DetN$$
$$Det \rightarrow o$$
$$Det \rightarrow a$$
$$N \rightarrow menino$$
$$N \rightarrow menina$$
$$VB \rightarrow viu$$
$$N \rightarrow binoculos$$
$$P \rightarrow com$$

Fonte: (BECK, 2012)

Este capítulo está organizado como segue. A seção 2.1 descreve o recurso essencial no aprendizado automático de um analisador sintático: os bancos de árvores (também conhecidos como *treebanks*). Na seção 2.2, o funcionamento de um *parser* probabilístico é explicado uma vez que é este o tipo de analisador sintático implementado neste trabalho. A seção 2.3 apresenta o processo de análise sintática multilíngue, objetivo deste trabalho. A abordagem neural, seguida neste trabalho, é explicada na seção 2.4 com especial atenção para a estratégia de aprendizado profundo, descrita na seção 2.5. Por fim, a seção 2.6

[2] Uma produção é uma derivação de uma regra gramatical.

descreve o funcionamento de um analisador sintático por transição, tipo de analisador sintático aqui implementado.

2.1 *Treebanks* e estruturas de representação (Conjunto de textos)

Uma representação no formato banco de árvores[3] está composta por árvores sintáticas para sentenças expressas por meio de um conjunto de etiquetas sintáticas/morfossintáticas (presentes nos nós intermediários) e um conjunto de palavras (presentes nos nós folha). De acordo com Manning e Schütze (1999), existe um grande esforço na produção de córpus desse tipo no formato *Penn Treebank* por sua alta utilização na construção de analisadores sintáticos estatísticos tanto supervisionados quanto não supervisionados.

Assim, a estrutura para uma representação de análise sintática em níveis é denominada árvore sintática, como a ilustrada na Figura 5. Na árvore da Figura 5 é possível verificar que as folhas (ou nós terminais) são as palavras da sentença, enquanto os nós intermediários (posicionados acima dos terminais) representam as categorias sintáticas e morfossintáticas. As etiquetas presentes na árvore seguem o conjunto de etiquetas do *Penn Treebank*, presentes na Tabela 1.

A literatura apresenta várias referências a córpus anotados seguindo o formato Penn Treebank como o WSJ (MARCUS MITCHELL P.; MARCINKIEWICZ, 1994)[4], para o inglês, e o Tycho Brahe (IEL-UNICAMP; IME-USP, 2010), para o português.

Estruturas de representação, em especial os *treebanks*, são importantes para a construção e validação de modelos estatísticos/probabilísticos. O "aprendizado" de regras para análise sintática probabilística em sua maioria se dá através de *treebanks*. Eles também são utilizados para a validação destes modelos pela separação destes em recursos de treinamento/teste. Este trabalho irá utilizar *treebanks* para aprendizado dos modelos probabilísticos.

Outra forma de representação usada na análise sintática são as estruturas de dependência, que se caracterizam por representar sentenças e suas relações de dependência na forma de grafos. Um córpus bastante conhecido que segue esta estrutura é o *universal dependencies* (UD). Esse córpus é um compêndio de notícias jornalísticas em diversas línguas, e tem como característica principal o fato de ser homogêneo em sua notação. A principal vantagem de um córpus com notação homogênea, segundo Tsarfaty et al. (2013), é a capacidade em combinar diferentes línguas durante o treinamento sem a necessidade de uma tradução entre notações.

A forma de representar relações de dependência no córpus UD, ou qualquer córpus de dependência, é através de etiquetas de relação. As relações universais de "raiz"

[3] Do inglês, *Treebank*.
[4] WSJ é o córpus do *Wall Street Jornal*.

Tabela 1 – Etiquetas presentes no Penn Treebank.

Etiqueta	**Descrição**
S	Senteça
ADJP	Adjetivo de frase
FV	Frase Verbal
FP	Frase prepocisional
VB	Verbo
ADVP	Locução adverbial
NP	Sintagma Nominal
PP	Locução Proposicional
QP	Expressão Quantificada
VP	Verbo
WHNP	Wh-Sintagma Nominal
WHNPP	Wh-Locução Proposicional
CONJP	Frase de muitas palavras com conjunções
FRAG	Fragmento de frase
INTJ	Interjeição/Exclamação
LST	Marcador de lista
NAC	Não é um grupo constituinte
NX	Constituinte Nominal
PRN	Intercalação
PRT	Porção mínima de uma frase
RRC	Cláusula de redução relativa
UCP	Frase de coordenação
X	Não conhecido
WHADJP	Wh-Adjetivo de frase
WHADVP	Wh-Verbo

Fonte: (MANNING; SCHÜTZE, 1999)

presentes no córpus UD estão descritas na Tabela 2.

A seção 2.1.1 traz exemplos de estruturas de constituintes e de dependência e a seção 5.1 apresenta os diferentes *treebanks* utilizados neste trabalho.

2.1.1 Estruturas de constituintes e estruturas de dependência

Existem duas formas de representação de uma estrutura de árvore sintática: usando constituintes ou dependência. A forma mais comumente utilizada é a de constituintes, na qual a ideia é organizar as palavras de uma sentença em uma estrutura composta por constituintes aninhados. A representação de dependência utiliza grafos para explicar as relações entre palavras. Essas relações se dão em nível gramatical. Uma sentença na forma de dependência pode ser vista como um grafo de vértices/palavras e arestas direcionadas como relações de dependência. A Figura 7 traz exemplos de estrutura de dependência e de constituintes para a sentença “O menino viu a bola”.

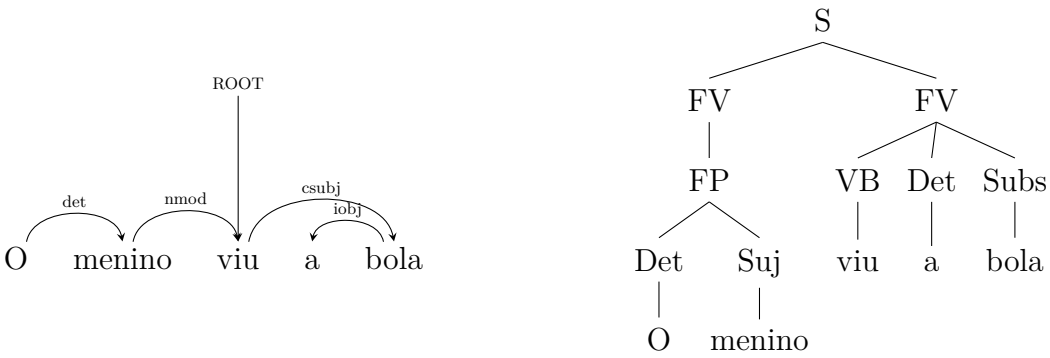
Definir corretamente o que é um constituinte não é algo fácil e leva a diversas discussões entre linguistas. Manning e Schütze (1999) apresentam duas principais formas

Tabela 2 – Etiquetas de relação presentes no UD.

Etiqueta	**Descrição**
xcomp	Complemento aberto
vocative	Vocativo
root	Raiz
reparandum	Disfluência sobreposta
punct	Pontuação
parataxis	Parataxe
orphan	Orfão
obl	Pronome nominal oblíquo
obj	Objeto
nummod	Modificador numérico
nsubj	Pronome do subordinado
nmod	Pronome do modificado
mark	Marcador
list	Lista
iobj	Relação indireta do objeto
det	Determinante
csubj	Relação clausal do sujeito
conj	Conjunção
cc	Conjunção coordenativo
amod	Modificador adjetiva
advcl	Moficador de adverbio de causa

Fonte: (NIVRE et al., 2016a)

Figura 7 – Estrutura de dependência (à esquerda) e de constituinte (à direita) para a sentença “O menino viu a bola”.



Fonte: próprio autor

para identificar corretamente um constituinte:

1. **Unidade de sentido** – um constituinte comporta-se como uma unidade que pode ser inserida em diferentes lugares de uma sentença sem modificar seu sentido. Por exemplo, as sentenças a seguir possuem o mesmo sentido:

    - Pablo conversou [com Alessandro] [sobre a vida].

- Pablo conversou [sobre a vida] [com Alessandro].

2. **Substituição/Expansão na sentença** – um constituinte comporta-se como uma unidade que pode ser inserida ou substituída na sentença. Por exemplo, os constituintes [na cadeira], [à esquerda daquela pessoa] e [ali] podem ser combinados/substituídos formando as sentenças:

   - Eu sento [ali].
   - Eu sento [na cadeira].
   - Eu sento [à esquerda daquela pessoa].
   - Eu sento [na cadeira] [à esquerda daquela pessoa / ali]

Formas como unidade de sentido, estrutura, substituição/expansão, semântica, etc. são algumas das inúmeras possibilidades de construção/definição de um constituinte como apontam Manning e Schütze (1999) tornando, assim, essa identificação uma tarefa complexa.

A estrutura de constituintes é muito estudada pois é facilmente relacionada a uma gramática da forma livre de contexto. Na área de PLN, as gramáticas livre de contexto são utilizadas para análise sintática, mas elas estão relacionadas com a Teoria X-barra que define como organizar uma gramática como explicam Manning e Schütze (1999).

A Teoria X-barra explica que a cabeça de uma regra gramatical só existe porque possui, em sua produção, uma palavra chave que suas palavras vizinhas[5] podem estar em qualquer ordem dentro da regra, e esta palavra que é fixa por consequência é a cabeça da frase. Como exemplo tem-se a gramática apresentada na Figura 8, a qual possui como cabeças de regra a lista ordenada $[SV, SP, S]$ e as respectivas listas de produções $[...VB..., ...P..., ...SV...]$. Percebe-se que a cabeça SV[6] possui um verbo em sua produção e a palavra associada a esta etiqueta sintática é quem governa a frase.

Figura 8 – Gramática Livre de Contexto na forma X-barra.

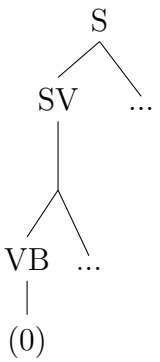
$$SV \rightarrow ...VB * ...$$
$$SP \rightarrow ...P * ...$$
$$S \rightarrow ...SV * ...$$

Como resultado da gramática tem-se a árvore de constituintes apresentada na Figura 9.

A Teoria X-barra possui outra particularidade muito interessante que permite transformar o modelo de estruturas de constituintes em um modelo de dependência.

[5] Nesse caso as palavras vizinhas devem pertencer ao mesmo constituinte.
[6] SV significa Sintagma Verbal.

Figura 9 – Árvore em forma de constituintes gerada com base na gramática da Figura 8.



Fonte: próprio autor

Atualmente, a maioria das pesquisas em análise sintática é realizada considerando-se constituintes, pela sua proximidade com a gramática e pela grande maioria dos recursos disponibilizados estarem nesta forma. Mas, como aponta Nivre (2014), a análise de dependência vem se tornando cada vez mais utilizada por sua facilidade em reduzir a esparsidade[7] dos dados, pois a palavra em uma estrutura de dependência pode estar "livre"[8] em qualquer lugar em uma produção de uma regra, antagonizando de um modelo de constituintes.

A escolha do NNParser ser um analisador sintático de dependência, dá-se por este tipo de analisador permitir uma maior liberdade na ordem das palavras tornando mais fácil a atividade de análise sintática multilíngue.

2.2 Análise sintática probabilística

A análise sintática probabilística é classificada por Manning e Schütze (1999) como uma implementação da técnica de *chunking*, que é o reconhecimento em alto nível de estruturas (*chunks*). Segundo esses autores, por meio do agrupamento de estruturas é possível condensar a descrição de uma sentença. Uma forma de capturar a regularidade dos *chunks* sobre um conjunto de palavras é aprender como as categorias são estruturadas via indução de gramática.

A indução de gramática tenta induzir empiricamente a estrutura a partir de um conjunto de árvores anotadas sintaticamente (banco de árvores, *TreeBank*). Esse con-

[7] Esparsidade, no contexto gramatical, refere-se à quantidade de etiquetas sintáticas/morfossintáticas presentes no domínio do problema. Isto é, as possibilidades de símbolos para representação, as quais são menores em uma estrutura de dependência.

[8] O termo "livre" em estruturas de dependência explica que: palavras podem estar em qualquer lugar de uma frase, desde que estejam em uma estrutura de dependência bem formada. Outras vezes este termo "livre" é atrelado ao grau de liberdade de um verbo, por exemplo, dentro de um sintagma verbal que é permitido graças à teoria X-barra.

junto de árvores serve como recurso para a indução e a extração de regras, obtidas automaticamente com base em características presentes nessas árvores. Para a indução são, geralmente, utilizados métodos de aprendizado bayesiano. Um método bayesiano vale-se da frequência de uma determinada regra e com base nessa frequência o modelo calcula as probabilidades da regra. Esse modelo pode levar em consideração a presença de rótulos nas classes dos dados (modelo supervisionado), ou somente medidas de agrupamento (modelo não supervisionado). Por exemplo, um modelo supervisionado geralmente associa uma etiqueta sintática a uma produção formando uma regra e sendo esta associada a alguma probabilidade. Já um modelo não supervisionado, geralmente agrupa produções a partir de medidas induzidas, como o quão presente esta produção está no conjunto de dados a ser induzido.

O analisador sintático também tem como tarefa desambiguar sentenças através da indução de gramáticas e atribuição de pesos diferentes às regras. A gramática da Figura 10 é um exemplo de uma gramática utilizada em um analisador sintático probabilístico. No exemplo anterior, o analisador selecionaria a regra com maior probabilidade, neste caso a de $P(S) = [0.6]$.

Figura 10 – Gramática probabilística utilizada para a geração das derivações para a sentença "O menino viu a menina de binóculos".

$$S \rightarrow NPVBP[0.6]$$
$$S \rightarrow NPVBPPP[0.4]$$
$$NP \rightarrow DetN[0.7]$$
$$NP \rightarrow NPPP[0.3]$$
$$VBP \rightarrow PNP[1.0]$$
$$PP \rightarrow DetN[1.0]$$
$$Det \rightarrow o[0.5]$$
$$Det \rightarrow a[0.5]$$
$$N \rightarrow menino[0.5]$$
$$N \rightarrow menina[0.3]$$
$$VB \rightarrow viu[0.2]$$
$$N \rightarrow binoculos[1.0]$$
$$P \rightarrow com[1.0]$$

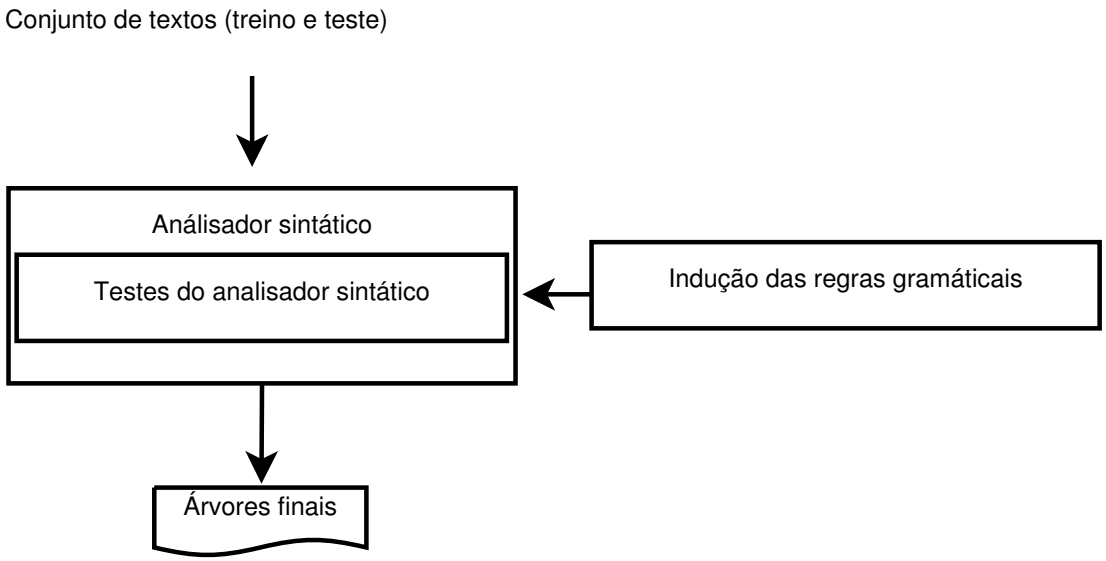
Fonte: (BECK, 2012)

Para construir as árvores sintáticas, como as dos exemplos da seção anterior, a partir de uma gramática, o analisador sintático utiliza-se de algoritmos como CYK (YOUNGER, 1967). Algoritmos como esse são capazes de construir uma árvore a partir de uma gramática. No caso, o algoritmo utiliza-se do método de programação dinâmica para maximizar as regras com maior peso probabilístico. Basicamente, o algoritmo encontra, para cada nó da árvore de derivação possível, todas as regras e as suas probabilidades associadas.

Para construir um parser probabilístico pode-se simplesmente escrever todas as regras gramaticais possíveis para uma linguagem, mas isso demanda muito tempo e traz a dificuldade de cobrir todos os casos possíveis. Então, como apontam Manning e Schütze (1999), a alternativa é criar um modelo de aprendizado que incrementalmente "aprenda" regras gramaticais e induzindo novos pesos por meio de uma etapa de treinamento, com posterior verificação da eficácia em uma etapa de teste.

A Figura 11 representa o funcionamento de um parser probabilístico. Que em uma primeira etapa define quais os *treebanks* serão utilizados para teste e treino[9]. Em seguida, o analisador fica encarregado de induzir a gramática gerando, assim, um modelo probabilístico. Durante a etapa de geração do modelo também são realizados pequenos testes para verificar a eficiência da indução gramatical e, por fim, o analisador gera o conjunto de árvores definitivas considerando aquelas com melhor ganho de acordo com alguma medida de avaliação calculada para o conjunto de teste.

Figura 11 – Processo de análise sintática probabilística.



Fonte: próprio autor

A construção e utilização de modelos probabilísticos baseados em aprendizado de máquina para análise sintática se faz necessária pela ambiguidade inerente das diferentes linguagens humanas. A correta identificação de sujeitos, contextos, dependência de palavras e suas classes gramaticais facilita o funcionamento de diversas tarefas como tradução e sumarização automáticas. O *parser* probabilístico tem como objetivo reduzir a ambiguidade gramatical e, por este motivo, serve como ferramenta chave para a análise sintática em língua natural.

[9] Esses conjuntos de treinamento e teste são textos previamente anotados e revisados. Esses córpus são, geralmente, conjuntos enormes de sentenças.

2.2.1 Análise sintática supervisionada vs. não supervisionada

Métodos de aprendizado de máquina são muito utilizados na área de PLN pois são capazes de resolver total ou parcialmente certos problemas, tal como a ambiguidade para análise sintática. O aprendizado pode ser realizado basicamente de três formas: (i) supervisionada para todos os dados presentes é possível classificar em alguma classe; (ii) não supervisionada, os dados não rotulados (sem rótulos para classes) devem ser agrupados com base em alguma característica e (iii) semi-supervisionada, na qual tem-se a união do agrupamento de dados não rotulados com alguma parcela de informação da classe.

Para a análise sintática tem-se a utilização de métodos probabilísticos a fim de reduzir a ambiguidade gramatical. Os analisadores sintáticos probalísticos são inicializados e configurados através de um modelo, como demonstra a Figura 11.

A configuração do modelo é a principal etapa no aprendizado de máquina. Esta etapa pode também ser chamada de treinamento onde, iterativamente, dados são extraídos e processados em um modelo probabilístico até ocorrer uma convergência.

Modelos de análise sintática probabilística supervisionada dependem de grande quantidade de dados[10] para induzir uma gramática automaticamente dificultando, assim, o aprendizado para línguas como o português que possuem poucos córpus com anotação sintática disponíveis. Outra grande limitação é se o treinamento for realizado com um córpus de um determinando domínio enquanto o teste é realizado com córpus de domínio diferente. Nesse caso, o desempenho será baixo, pois técnicas de supervisão têm limitações para classificar palavras com domínios ou contextos diferentes, por exemplo.

Já os modelos de aprendizado não supervisionado utilizam-se de dados não rotulados[11] e fazem a indução ou detecção de estruturas. O principal incentivo para utilizar aprendizado não supervisionado em análise sintática é que existem poucos córpus com anotação sintática disponíveis para o português, tornando a utilização dessa técnica muito atrativa para indução de gramáticas. Outra característica interessante desta técnica é que ela também pode ser combinada com outras técnicas como o agrupamento[12] e algoritmos de otimização de aprendizado.

Por fim, modelos de aprendizado semissupervisionados são uma união de aprendizado supervisionado com não supervisionado. No aprendizado não supervisionado, por exemplo, pode-se induzir o melhor constituinte possível para uma sentença sem utilizar as etiquetas de uma árvore. Já a técnica de supervisão utiliza as etiquetas presentes no córpus de treinamento para induzir os constituintes das frases e qual a sua etiqueta sintá-

[10] Por exemplo, na forma de *treebanks*.
[11] No contexto de análise sintática, dados não rotulados são dados sem nenhuma marcação sintática/morfossintática.
[12] Do inglês, *clustering*.

tica. Unindo um modelo não supervisionado com pequenas taxas de supervisão pode-se, por exemplo, aprender onde estão constituintes de uma frase (não supervisionado) e suas etiquetas (supervisionado).

Atualmente, para a tarefa de análise sintática probabilística têm-se utilizado diversas técnicas. As principais técnicas utilizadas estão presentes na Tabela 3, onde os modelos são divididos em métodos supervisionados ou não supervisionados. Alguns métodos são o estado da arte até hoje para diversas línguas, entre elas o inglês.

Tabela 3 – Modelos supervisionados vs. não supervisionados

Modelo	**Supervisionado**	**Não supervisionado**
CCM+DMV[13]		X
Self-trained parser[14]	X	
Charniak and Johnson's Parser[15]	X	
Sparsity in dependency grammar induction [16]		X
Tree Substitution Grammars [17]		X

Fonte: próprio autor

O estado da arte para análise sintática é o analisador sintático supervisionado proposto por McClosky, Charniak e Johnson (2006) (*Self-trained parser*) inicialmente para o inglês. Esse *parser* utiliza uma técnica de re-ranqueamento (*re-ranking*) de modelos previamente testados em conjunto com dados aprendidos previamente a partir de dados não etiquetados. McClosky, Charniak e Johnson (2006) advertem que é possível o re-ranqueamento das árvores sintáticas, pois o modelo utiliza uma técnica de combinação de classificadores onde dados não supervisionados são agregados ao classificador original.

Como explica Ng (2009), modelos de aprendizado semissupervisionado que utilizam-se de *bootstrap* o fazem através de uma grande quantidade de dados, e de duas formas com auto-treinamento ou com co-treinamento. Uma técnica de aprendizado com auto-treinamento é aquela na qual o modelo é capaz de associar uma etiqueta em um conjunto de dados não etiquetados (não supervisionado). A medida que a crença construída é adquirida, o modelo passa por uma supervisão (dados supervisionados), quase que como um método de reforço e correção para as crenças adquiridas.

Já um modelo com co-treinamento é uma escolha natural para quando se quer induzir uma etiqueta para dois conjuntos de dados independentes. Essa independência é algo razoavelmente assumido e raramente é satisfeito na prática. Ambas as formas de

[13] (KLEIN; MANNING, 2004)
[14] (MCCLOSKY; CHARNIAK; JOHNSON, 2006)
[15] (CHARNIAK; JOHNSON, 2005)
[16] (GILLENWATER et al., 2010)
[17] (BLUNSOM; COHN, 2010)

aprendizado, com (co/auto)-treinamento, são as principais do estado da arte para análise sintática.

A técnica utilizada em (NG, 2009) é um método não supervisionado para a indução de crenças sobre uma representação (estados de análise sintática). Métodos atuais de aprendizado profundo necessitam de uma leve supervisão durante o treinamento. Então, o modelo se torna "semissupervisionado" no sentido que há reforço no aprendizado a partir de representações adquiridas durante o aprendizado.

2.3 Análise sintática multilíngue

Como já mencionado anteriormente, idealmente um analisador sintático deveria obter um desempenho médio para todas as línguas. Mas, atualmente, nenhum modelo de análise sintática consegue alcançar os mesmos 92, 1% obtidos no modelo de (MCCLOSKY; CHARNIAK; JOHNSON, 2006; CHARNIAK; JOHNSON, 2005) para inglês, em outro idioma, por exemplo, o português. Alguns pesquisadores, dentre eles destacam-se Bohnet et al. (2013a) e Tsarfaty et al. (2013), buscam explicar qual o problema que ocorre com modelos para análise sintática automática. O problema de se obter um alto desempenho para o inglês e não para o português, por exemplo, deve-se ao fato de que o analisador sintático, na etapa de indução de regras gramaticais, especifica o modelo com base no córpus de treinamento. Esse problema, como explicam Tsarfaty et al. (2013), se dá pelo fato do inglês ser uma língua muito simples do ponto de vista morfossintático, diferentemente de línguas ricas morfologicamente, como é o caso do português, espanhol, sueco, etc.

O esforço atual em pesquisas voltadas para a geração de um *parser* multilíngue está focado nas línguas ricas morfologicamente, como apontam Bohnet et al. (2013a). Esses autores explicam que a efetividade de um analisador sintático para línguas ricas morfologicamente tende a ser menor do que para línguas como o inglês.

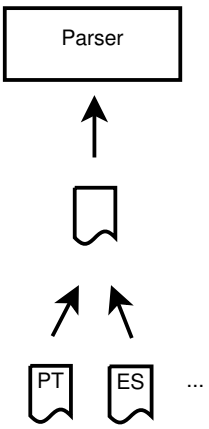
O problema das línguas ricas morfologicamente pode ser definido como um problema difícil de se resolver. Tsarfaty et al. (2013) relatam que para este tipo de línguas existe uma separação entre morfologia e sintaxe, porque as informações de relação entre elementos sintáticos são indicadas na forma das palavras, e essas palavras podem ocorrer livremente em uma sentença. Essa liberdade de poder mudar de posição em uma sentença dificulta a interpretação do modelo sintático de um modo geral, pois em um modelo de análise de constituintes a posição de uma palavra na sentença importa.

Outro problema apontado por Tsarfaty et al. (2013) é a esparsidade dos dados em relação à proporção dos elementos, o que reflete em um grande número de palavras diferentes que não foram vistas anteriormente no modelo. Eles ainda explicam que esse problema pode ser facilmente resolvido usando um *parser* não supervisionado ou por meio de um *parser* de dependência, pois em um modelo de dependência não interessa a posição

de uma palavra na frase e sim com quem ela mais se relaciona diminuindo, assim, a separação entre morfologia e sintaxe.

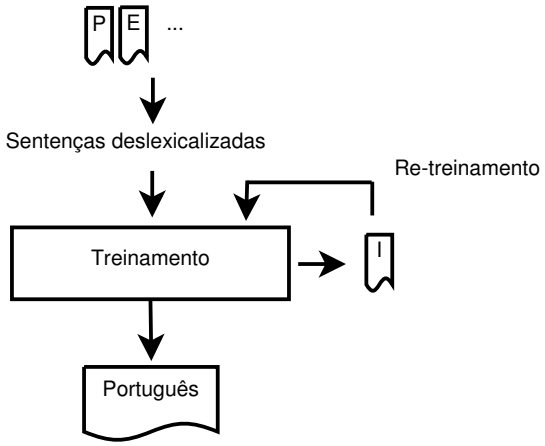
A ideia por trás de um analisador sintático multilíngue, como ilustrado na Figura 12, é existir um modelo aprendido de forma supervisionada ou não supervisionada que seja capaz de processar qualquer língua de entrada. Como aponta o estado da arte, isto pode ocorrer de diversas formas, mas as principais são baseadas em: (i) simplificação de etiquetas sintáticas e clusters de palavras, como os trabalhos de (MCDONALD; PETROV; HALL, 2011) e (TACKSTROM; MCDONALD; USZKOREIT, 2012), ou (ii) alinhamento de árvores sintáticas, como os trabalhos de (SNYDER; NASEEM; BARZILAY, 2009) e (MCDONALD; PETROV; HALL, 2011).

Figura 12 – Processo de análise sintática multilíngue



Fonte: próprio autor

Figura 13 – Análise de dependência deslexicalizada sem alinhamento

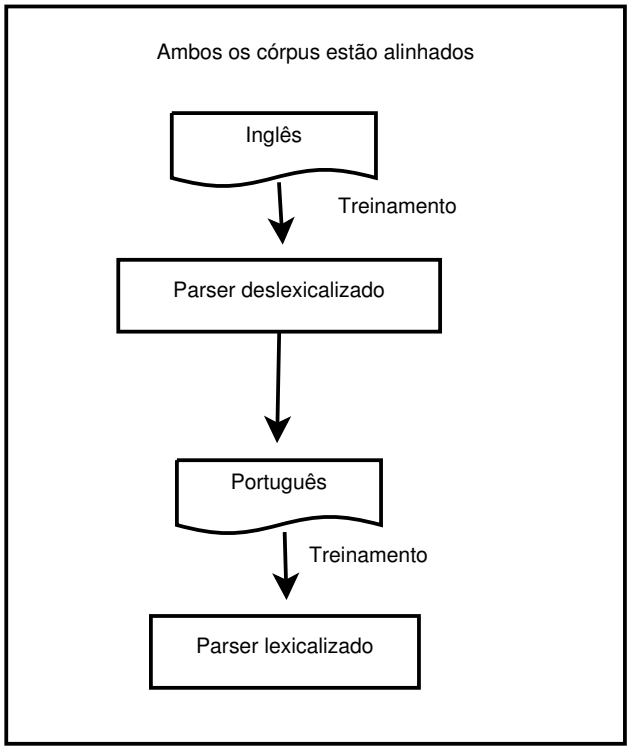


Fonte: adaptado de (MCDONALD; PETROV; HALL, 2011)

McDonald, Petrov e Hall (2011) apresentam duas propostas para a análise de dependência em linguagens ricas morfologicamente. Na primeira proposta são utilizadas técnicas de aprendizado não supervisionado sem a utilização de tokens/folha deslexicalizados[18]. Esta forma de treinamento permite que um analisador seja treinado em qualquer linguagem e seja testado em outra, seguindo a ideia do parser multilíngue em sua plenitude e utilizando-se de um córpus concatenado.

Como pode-se perceber pelo exemplo na Figura 13, o modelo proposto não utiliza alinhamento sintático entre árvores, para isso McDonald, Petrov e Hall (2011) propõem utilizar um mapeamento entre os córpus por meio do mesmo conjunto de etiquetas morfossintáticas. Este mapeamento de etiquetas, em conjunto com a ausência de palavras, permite que o modelo analise qualquer língua pertencente ao conjunto de treinamento. O exemplo demonstra que há um retreinamento do *parser* deslexicalizado, onde este retreinamento permite que ocorra a análise da língua alvo, no caso o português. Outro ponto importante é que esse modelo realiza análise a partir do nível morfossintático (sem a necessidade da presença de palavras).

Figura 14 – Análise de dependência deslexicalizada com alinhamento



Fonte: adaptado de (MCDONALD; PETROV; HALL, 2011)

Em sua segunda proposta, McDonald, Petrov e Hall (2011) propõem algo similar a (SNYDER; NASEEM; BARZILAY, 2009), que é utilizar o alinhamento sintático entre

[18] Em análise de dependência deslexicalizada considera-se somente os itens morfossintáticos.

as árvores do conjunto de treinamento lexicalizado realizado por meio de um modelo de mapeamento[19]. Com esse modelo é possível analisar sintaticamente qualquer linguagem desde que esteja no conjunto de treinamento lexicalizado[20].

A Figura 14 traz um exemplo do funcionamento do modelo proposto por McDonald, Petrov e Hall (2011) de mapeamento deslexicalizado com córpus paralelos. Neste exemplo percebe-se que o modelo está dividido em três etapas. Na primeira etapa, utiliza-se um analisador sintático deslexicalizado de dependência treinado em inglês para criar um córpus dourado no português. A segunda etapa utiliza o córpus "dourado" em português como treinamento de outro analisador sintático lexicalizado. Este modelo permite, assim, que uma terceira etapa analise sintaticamente uma sentença em português. Mas para este modelo funcionar corretamente ambos os córpus devem estar alinhados e com o mesmo conjunto de etiquetas morfssintáticas.

Uma alternativa para o problema também é a construção de dicionários morfossintáticos para linguagens ricas, o que é fortemente discutido em (MCDONALD; PETROV; HALL, 2011). Hajic (2000) demonstra que com a simples construção de dicionários baseados somente em informações léxicas pode-se aumentar consideravelmente o desempenho e a cobertura de um analisador morfossintático, sem a necessidade de se construir mais recursos anotados. Já Goldberg e Elhadad (2013) propõem a utilização de recursos léxicos em nível gramatical, demonstrando o aumento de resultados para o hebreu moderno.

Pesquisas recentes vêm testando modelos semissupervisionados (KOO; CARRERAS; COLLINS, 2008), transferência de linguagem, árvores deslexicalizadas e o uso de textos paralelos e não paralelos (MCDONALD; PETROV; HALL, 2011), e *cluster* de palavras, com recursos deslexicalizados entre linguagens (TACKSTROM; MCDONALD; USZKOREIT, 2012). Essas pesquisas demonstram a importância de se juntar dados não supervisionados aos modelos, pois existem poucos recursos anotados em diversas linguagens, principalmente para linguagens ricas morfologicamente. Tackstrom, McDonald e Uszkoreit (2012) ainda explicam que a utilização de *clusters* auxilia na redução da esparsidade dos dados em línguas ricas morfologicamente.

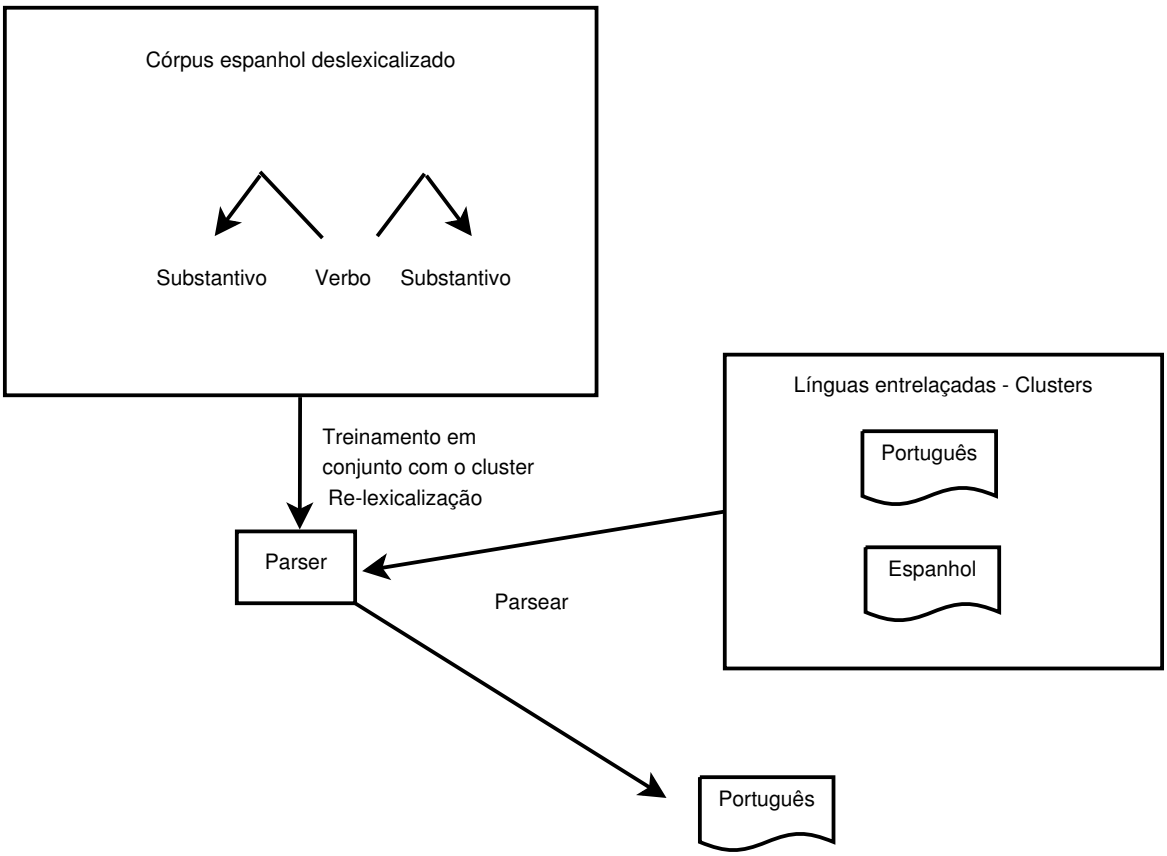
Como percebe-se pela ilustração da Figura 15, a proposta de Tackstrom, McDonald e Uszkoreit (2012) prevê o treinamento de um *cluster* não supervisionado que aprende características de semelhanças entre palavras de línguas diferentes (português e espanhol, no exemplo). O autor adverte que a fonte de recursos utilizada deve estar entrelaçada, isto é, trata-se de sentenças paralelas. O *cluster* é, então, utilizado para a ligação com o conjunto de treinamento sintático, que foi utilizado no treinamento do *parser*. Por meio dessa ligação é possível que o analisador sintático gerado a partir do córpus em espanhol

[19] Um modelo de mapeamento possui as mesmas etiquetas sintáticas/morfossintáticas em todas as árvores das línguas teste/treinamento.

[20] Utilizando palavras em conjunto com níveis morfossintáticos.

analise qualquer sentença em alguma linguagem alvo, no caso o português. Esse modelo permite escapar do alinhamento sintático, pois como adverte Nivre (2014), o alinhamento automático é uma tarefa difícil.

Figura 15 – Análise de dependência deslexicalizada com *cluster* de palavras



Fonte: adaptado de (TACKSTROM; MCDONALD; USZKOREIT, 2012)

Alguns trabalhos recentes destacam que a utilização de inferência morfossintática e sintática conjunta melhora os resultados, tanto que Lee, Naradowsky e Smith (2011) destacam a utilização do modelo conjunto e também de recursos de linguagens irmãs como boas estratégias para melhorar os resultados. Em (BOHNET et al., 2013b), destaca-se o uso de recursos morfossintáticos e sintáticos, bem como a utilização recursos léxicos como "atributos" previamente induzidos em métodos não supervisionados em combinação com um modelo de transição de dependência.

Como aponta Nivre (2014), a análise morfossintática, os recursos léxicos e *cluster* distribuído de palavras são as principais estratégias para a solução do problema de *parser* multilíngue envolvendo linguagens ricas morfologicamente. Por fim, ele aponta ainda que o ideal seria a construção de uma gramática universal, a qual facilitaria o estudo linguístico das linguagens e, também, a motivação para a construção de córpus multilíngues[21] e, o

[21] Como o trabalho de McDonald, Petrov e Hall (2011) com anotação sintática, que pode ser encontrado

mais importante, a integração de recursos heterogêneos.

Neste trabalho foram utilizadas as técnicas de transferência de linguagem considerando-se (ou não) as similaridades linguísticas (línguas irmãs). Mais especificamente, no âmbito de transferência de linguagem utilizou-se uma técnica para encontrar a similaridade entre representações distribuídas de palavras através de dados paralelos. Também foi utilizada uma segunda forma de análise multilíngue, onde córpus são concatenados sem nenhuma regra prévia.

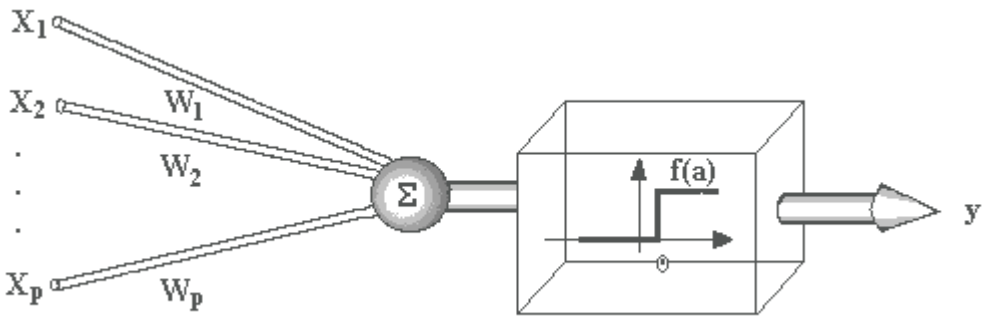
2.4 Aprendizado Neural

Um método neural de aprendizado é um modelo baseado no modo como os neurônios se comportam, considerando experiências passadas. Como no cérebro humano, o aprendizado neural é dividido em unidades de processamento, chamadas de neurônios, que podem estar organizadas em grupos formando, assim, as redes neurais de aprendizado.

Mcculloch e Pitts (1943) foram os primeiros a utilizar o termo "redes neurais" quando propuseram um modelo de aprendizado muito simples que simularia uma máquina. Porém, somente em (ROSENBLATT, 1958) foi proposta a ideia de rede de perceptron.

A Figura 16 representa um esquema de aprendizado neural, onde existem entradas que são denotadas pelos símbolos $X_1, X_2, ..., X_n$, com pesos associados representados por $W_1, W_2, ..., W_n$. Geralmente esses pesos são aprendidos automaticamente por meio de algum método de ajuste de função, por exemplo, métodos de regressão linear.

Figura 16 – Exemplo de um modelo neural.



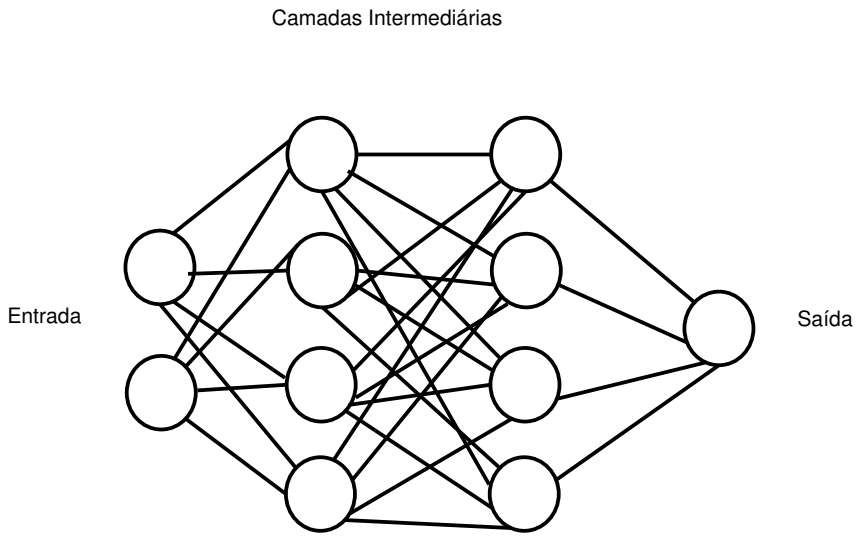
Fonte: (ANDRE, 2014)

em <https://code.google.com/p/uni-dep-tb/>, que incentiva a construção de córpus multilíngues não paralelos.

De acordo com Kovacs (2002), estruturas neurais de aprendizado são organizadas em camadas, como demonstra a Figura 17. Essas camadas são conectadas entre si[22] e cada camada possui unidades de aprendizado neural com inter-conexões para a comunicação. Basicamente, existem três tipos de camadas:

- A camada de entrada – onde sinais de entradas são propostos à arquitetura;
- As camadas intermediárias ou escondidas – onde ocorre o processamento da rede e seu aprendizado;
- A camada de saída – onde ocorre a decisão ou a apresentação dos resultados.

Figura 17 – Exemplo de uma arquitetura neural.



Fonte: próprio autor

Arquiteturas neurais se diferenciam pelo seu método de aprendizado iterativo, realizado por meio de um método linear que, quando converge, indica que o aprendizado da rede está concluído. Existem três maneiras de uma rede neural realizar o seu aprendizado:

- Aprendizado supervisionado – quando um sinal externo indica à rede o valor esperado, dadas as entradas;
- Aprendizado não supervisionado – quando existe uma auto-organização da rede;
- Aprendizado semissupervisionado – quado existe algum avaliador externo, algum nó decisor.

[22] Essa estrutura de rede neural conectada é conhecida como *Feed-Foward Neural Network* ou também denominada de rede "deep".

Kovacs (2002) explica que, para a correção dos fatores de aprendizado, existem duas formas: (i) o modo normal, onde a correção ocorre pelo cálculo do erro a cada iteração, e (ii) o modo *batch*, onde existe uma correção por ciclo de treinamento[23] e é utilizado um cálculo de erro médio a partir do conjunto de treinamento.

O modelo neural proposto por Mcculloch e Pitts (1943) era um modelo muito simples de aprendizado, com o qual se acreditava que seria possível representar e aprender quaisquer coisas com uma camada somente. Mas, Rosenblatt (1958) propôs a criação de uma rede neural com diversas camadas o que possibilitou que uma rede neural seja capaz de representar estruturas complexas.

A Figura 17 apresenta uma estrutura de rede neural multi-camadas comumente denominada *Multi Layer Perceptron*. Essa estrutura se diferencia do modelo original por representar o paradigma estatístico de reconhecimento de padrão, que tem três características:

- Conversão da entrada em um vetor de características na forma numérica, geralmente por meio de algum método de *cluster*;
- Aprendizado de pesos para cada elemento do vetor, onde um único valor numérico é atribuído;
- Unidade decisora, similar à proposta de Mcculloch e Pitts (1943), responsável por decidir se novo vetor de pesos é um exemplo positivo da classe/decisão alvo.

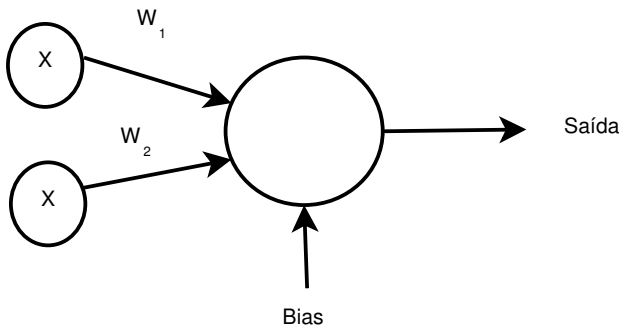
Também existe um sinal extra na unidade neural como pode-se perceber pela Figura 18, que é o sinal do *bias*. O *bias* é um peso extra que deve ser colocado no vetor antes da camada de decisão. Esse peso pode ser aprendido tornando, assim, a seleção dos vetores mais correta e confiável.

A arquitetura *feed-foward* foi contestada no período dos anos 60 e 70 por somente conseguir representar problemas com separação linear, como por exemplo a separação de objetos em imagens e problemas simples de classificação binária[24], problemas que podem ser separados por um plano que corte. As redes neurais foram esquecidas durante esse período, com poucos avanços relevantes.

A área ganhou novo fôlego quando Rumelhart, Hinton e Williams (1986) propuseram uma rede neural com aprendizado baseado em regressão linear (*backpropagation*), onde camadas intermediárias poderiam ser treinadas facilmente. Estas camadas intermediárias são responsáveis pela extração de características, permitindo a seleção de um vetor resposta mais correto. Também vale ressaltar que essa proposta permite um mapeamento mais correto para problemas não separáveis linearmente, diferente de seu modelo anterior.

[23] Existem N iterações em um ciclo.
[24] Ainda nos anos 60 o método de seleção era a função arco-tangente.

Figura 18 – Exemplo de um neurônio simples.



Fonte: adaptado de (SOCHER; MANNING; BENGIO, 2013)

Com a proposta da rede com *backpropagation*, uma unidade neural (neurônio) pode ser vista como um neurônio logístico[25]. Esta nova característica permite que o aprendizado seja suavizado, permitindo ao modelo selecionar corretamente o vetor de características. Rumelhart, Hinton e Williams (1986) também propuseram a utilização de um novo método seletor, ao invés de utilizar-se da função arco-tangente, o novo modelo de rede neural utilizaria a função sigmoide, o que permite uma representação mais poderosa das características.

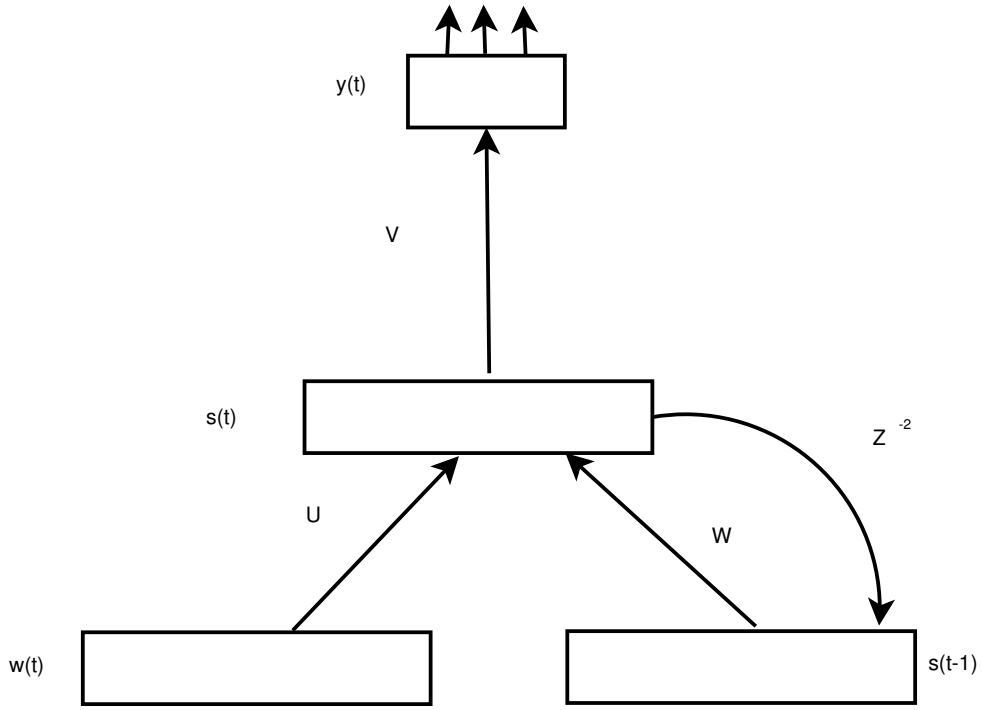
Outra forma muito utilizada de arquitetura neural é a recorrente. Esta arquitetura possui um poder de representação muito maior, mas é também mais complexa para realizar o aprendizado do vetor de características (*features*). Essa complexidade está associada ao tipo de conexão entre os nós de uma camada e entre as camadas, podendo ocorrer ciclos, mas isto pode ser facilmente contornado graças à organização dos neurônios. A Figura 19 demonstra a arquitetura recorrente.

Hintom (2014) explica que a rede neural recorrente com múltiplas camadas é um caso especial da *Feed-Foward Neural Network*, onde existem camadas "escondendo" camadas, tornando praticamente impossível retornar todo o caminho do aprendizado. Este tipo de modelo neural também é uma forma muito natural e fácil para modelar dados sequenciais.

A principal diferença entre as redes neurais recorrentes e *multi layer* é que os pesos aprendidos em um passo do treinamento, em uma rede recorrente, são reutilizados em todo o processo da rede, isto é, cada unidade escondida utiliza os mesmos pesos para aprender as características. Como demonstra a Figura 20, existem saídas a cada etapa das unidades escondidas, assim a rede recorrente guarda informação durante muito tempo, mas pode ser muito difícil de utilizar essa característica.

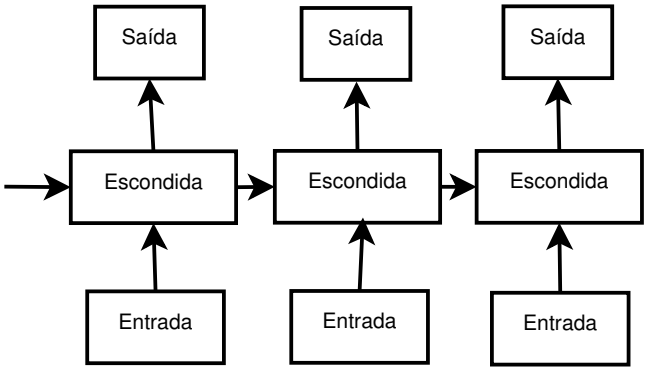
[25] *Non-Linear Neuron.*

Figura 19 – Diagrama de um modelo neural recorrente.



Fonte: (MIKOLOV et al., 2013)

Figura 20 – Exemplo de um modelo de aprendizado com camadas escondidas.

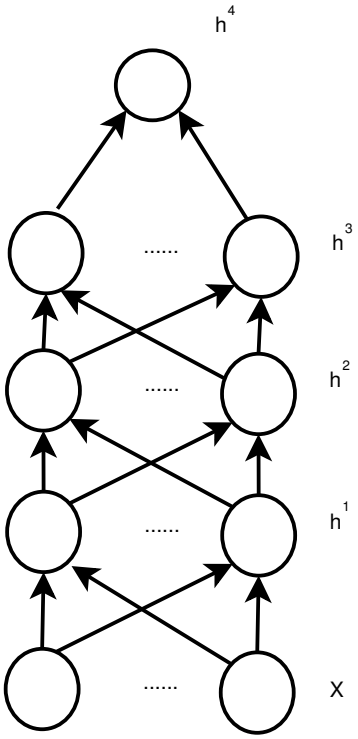


Fonte: próprio autor

2.5 *Deep learning*

O aprendizado profundo (do inglês, *Deep Learning*) é uma estratégia computacional para aprendizado de máquina que explora níveis de representação inferiores, no que tange uma arquitetura neural, e busca construir representações mais gerais. Ele pode ser visto em forma de árvore onde as folhas são características de entrada e a cada nó pai uma escolha generativa da melhor característica acontece.

Figura 21 – Exemplo de uma arquitetura *deep*.



Fonte: (SOCHER; MANNING; BENGIO, 2013)

Como explicam Socher, Manning e Bengio (2013), a maioria dos trabalhos atuais em arquiteturas *deep* estão concentrados em redes neurais com o objetivo de classificar *markov random fields*[26] com várias camadas e diversas outras arquiteturas de redes neurais com camadas. A arquitetura *deep*, como pode-se perceber pela Figura 21, é muito similar a uma *Feed-Foward Neural Network*: com o conjunto $X = \{X_1, X_2, ..., X_n\}$ como entrada, camadas[27] ocultas que aprendem representações mais abstratas à medida que sobe na hierarquia, e uma camada de decisão que é feita de forma supervisionada.

O *Deep Learning* vem sendo muito utilizado por sua capacidade de aprender muito facilmente níveis de representação complexa ou abstrata. Diferentemente de métodos de modelagem manual de *features*, que consomem muito tempo, o *deep learning* se baseia no fato de que os computadores deveriam fazer o mesmo que seres humanos, que criam representações diariamente, e propõe métodos simples, muitas vezes não supervisionados, para o aprendizado de características. Socher, Manning e Bengio (2013) afirmam que o *deep learning* proporciona esse aprendizado de estruturas por meio do aprendizado distribuído de características.

[26] *Markov Random Fields* é um esquema gráfico e teórico para classificação onde dados são vistos de forma temporal e a classificação é da seguinte forma: uma previsão (t) somente sofre influencia do estado anterior (t_{-1}).

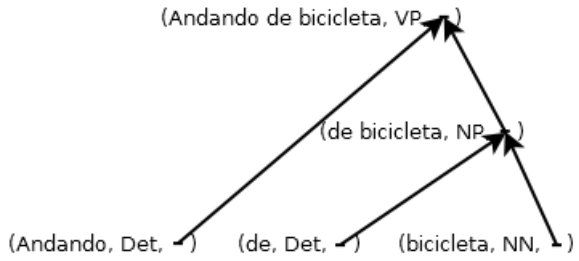
[27] $H = \{H_1, H_2, ..., H_n\}$

Diversas tarefas de PLN demandam representações complexas, como ocorre em um analisador sintático e suas representações complexas de árvores, tornando praticamente impossível mapear corretamente um domínio à mão. Aprender a representar palavras de forma distribuída ajuda a resolver esse e outros problemas, como apontam Socher, Manning e Bengio (2013).

A representação de palavras de forma distribuída auxilia diversas tarefas do PLN uma vez que a similaridade dos *clusters* de palavras auxilia a reduzir o erro em diversas aplicações de PLN, como no trabalho de (CHEN; MANNING, 2014) de parser de dependência[28].

O *cluster* de palavras com base neural tem como princípio o fato de que características que não são mutuamente exclusivas podem ser exponencialmente mais eficientes que técnicas como vizinho mais próximo e *hard clustering*. Também conhecida como *multi-clustering*, como apontam Socher, Manning e Bengio (2013), o *cluster* de palavras com base neural destaca-se por aprender diversas características de uma única palavra, não somente agrupando palavras como a maioria das técnicas de agrupamento. Sentenças em língua natural são compostas de palavras e frases e para lidar com essas estruturas, Socher, Manning e Bengio (2013) apontam que os modelos *deep* devem utilizar o princípio da composicionalidade, onde recursivamente o mesmo operador é aplicado repetidamente em diferentes componentes. Como pode-se perceber pela Figura 22, a composição se dá em nós da árvore sintática. Deve-se ver o exemplo como uma rede neural, onde cada nó pai é associado à composição do vetor características de seus filhos.

Figura 22 – Exemplo do funcionamento da análise de uma frase.



Fonte: adaptado de (SOCHER; MANNING; BENGIO, 2013)

Como apontam Socher, Manning e Bengio (2013), anteriormente ao ano de 2006, treinar modelos *deep* era algo impossível. Essa situação mudou com o surgimento de novos métodos de pré-treinamento das redes neurais, melhores métodos para estimar parâmetros, melhores métodos de regularização do modelo, e avanços no poder computacional.

[28] O uso de representação distribuída trouxe uma redução de 15,2% no erro em comparação a proposta original de (HALL; NILSSON, 2006).

Graças aos avanços na computação paralela e escalável, modelos *deep* podem ser muito rápidos como apontam Socher, Manning e Bengio (2013).

2.5.1 Representação distribuída

A forma mais comum de representar palavras e sentenças nos métodos de PLN baseados em regras ou estatísticos/probabilísticos é por meio de símbolos únicos. A representação mais intuitiva está presente na Figura 23 onde a palavra "Casa" é denotada por um vetor binário. Cada palavra do conjunto de palavras possíveis teria um vetor diferente tornando a montagem de regras lógicas e comparações computacionais algo possível.

Figura 23 – Representação vetorial intuitiva da palavra Casa.

$$Casa = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]$$

Fonte: próprio autor

Contudo, essa representação vetorial binária apresenta falhas como apontam Socher, Manning e Bengio (2013). Por ser uma representação muito simples, ela não consegue representar o significado semântico/sintático de uma palavra ou sentença. Como percebe-se pelo exemplo da Figura 24, as palavras "Casa" e "Apartamento" possuem significado semântico próximo que é o de moradia, porém com essa representação simbólica única[29] não é possível representar palavras e seus contextos tornando, assim, inviável a representação do sentido semântico dessas palavras.

Figura 24 – Comparação vetorial entre as palavras Casa e Apartamento.

$$Casa = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]$$

$$\wedge$$

$$Apartamento = [0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0] = 0$$

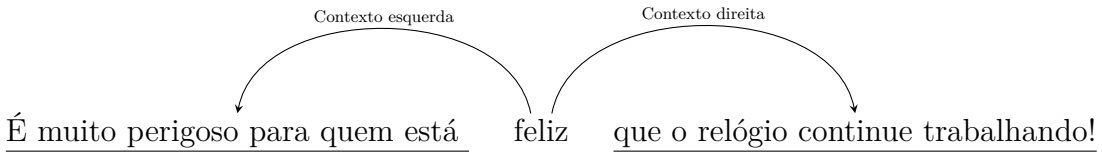
Fonte: próprio autor

Uma forma mais inteligente de representar palavras ou sentenças é agregando conhecimento de seus vizinhos ou contextos. Essa forma de representação é uma das técnicas de mais sucesso em PLN. Em 1957, J. R. Firth disse "*You shall know a word by the company it keeps*" e essa frase ecoa até hoje na área de PLN.

[29] Por vezes denominada *Atomic Symbol*.

Como pode-se perceber pela Figura 25, onde está representada a frase do autor Paulo Sant'Ana, a palavra "feliz" possui como contextos/vizinhos os demarcados pelas setas. As setas também representam as palavras que auxiliam a representação da palavra "feliz". O tamanho do contexto pode variar e essa variação auxilia no ajuste correto do entendimento do contexto ou significado semântico da palavra/sentença alvo.

Figura 25 – Representação do contexto da palavra "feliz".



Fonte: próprio autor

A utilização agregada de contexto de palavras com técnicas de clusterização para a representação de palavras atualmente apresenta os melhores resultados (SOCHER et al., 2013; CHEN; MANNING, 2014; MIKOLOV; LE; SUTSKEVER, 2013) como apontam Socher, Manning e Bengio (2013). Das técnicas, destacam-se *hard* e *soft* aglomeração. A técnica de classe, também chamada de *hard* aglomeração, é uma técnica que aprende similaridades entre classes de palavras baseada em informação distribuída. Um dos modelos mais utilizados é o *Brown* aglomeração proposto por Brown, Souza e Mercer (1992).

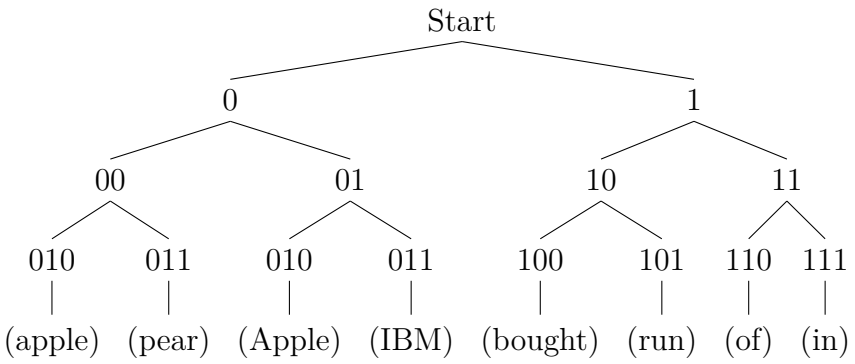
O algoritmo *Brown* aglomeração tem como premissa o particionamento de vocabulários presente em córpus de clusteres[30]. O algoritmo idealmente combina palavras por similaridades semânticas onde estas informações podem ser obtidas de forma supervisionada ou não supervisionada através do córpus.

Como saída do algoritmo tem-se uma árvore binária como a ilustrada na Figura 26. A saída é uma árvore binária pois o agrupamento é feito em pares. Percebe-se que o algoritmo consegue diferenciar nomes próprios e nomes de frutas, como mostra o exemplo de (Apple, apple). Esse algoritmo é utilizado em diversas aplicações de PLN, como análise de dependência (KOO; CARRERAS; COLLINS, 2008)[31]. Contudo, uma desvantagem desse algoritmo é que ele pode ser guloso (*greedy*) alcançando um máximo local, o que pode levar a uma solução não ótima.

Para garantir uma menor esparsidade nos dados, a técnica mais recomendada é a representação de palavras distribuídas ou *soft* aglomeração ou também representação distribuídas de palavras. Este tipo de técnica tem como objetivo aprender, para cada tópico, a distribuição sobre as palavras; ou predizer a qual tópico mais provável uma determinada palavra pertence. Alguns algoritmos utilizados são: Distribuição Latente Semântica (LSA), que utiliza projeções randômicas para agrupar as palavras; e Análise

[30] Este tipo de córpus apresenta classes e palavras presentes nestas classes.
[31] Tal trabalho utiliza uma versão semissupervisionada do algoritmo.

Figura 26 – Árvore do cluster binário de Brown.



Fonte: adaptado de (BROWN; SOUZA; MERCER, 1992)

Latente de Distribuição (da sigla em inglês LDA), que utiliza aglomeração generativo para agrupar as palavras.

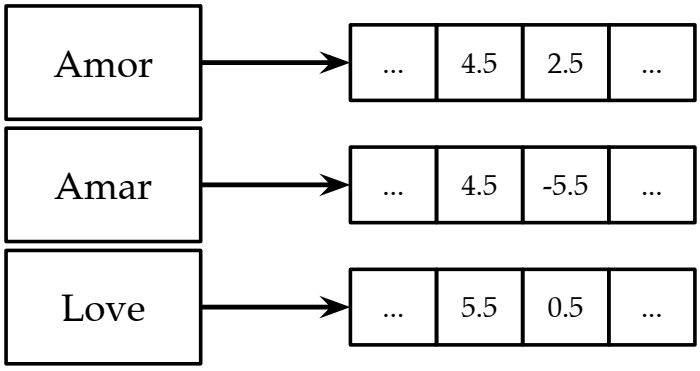
O algoritmo LDA é capaz de agrupar textos por assunto, como um classificador de assuntos, ou agregador de tópicos. Ele tem como cerne a aplicação probabilística de inferência (*bayes*) entre as palavras de um tópico, assim ele monta uma rede de um texto para poder classificá-lo. Já o algoritmo LSA divide o texto em tópicos como uma distribuição de forma matricial. Esse algoritmo tem como princípio a aproximação de palavras presentes em tópicos através de distâncias, por exemplo: distâncias entre palavras, distância de uma palavra para ponto final, quão próximas duas palavras são, etc. Ambos os algoritmos são utilizados em PLN para agrupar textos em tópicos.

2.5.2 Representação distribuída com redes neurais

Recentemente, diversos modelos neurais começaram a ser propostos com uma ideia similar à dos métodos distribuídos de palavras. Esses modelos neurais são construídos para representação embutida de palavras de forma distribuída. A ideia geral é combinar vetores de espaço semântico (*Soft Clustering*) com modelos de previsões probabilísticas. Trabalhos como (BENGIO et al., 2003), (KARLEN et al., 2008) e (TURIAN; RATINOV; BENGIO, 2010) utilizam esses modelos inclusive com métodos de aprendizado profundo. Nesses modelos, as palavras geralmente são representadas como um vetor denso de características, como o apresentado na Figura 27, onde palavras em diversas línguas apresentam características na forma decimal.

Estas características que estão condessadas na Figura 27 na forma decimal, representam características distribuídas de palavras em relação ao seus diferentes contextos. Esta representação densa representa as distâncias da palavra “amor” em relação ao ponto final da sentença, ao início do texto, à palavra com letra maiúscula mais próxima, etc.

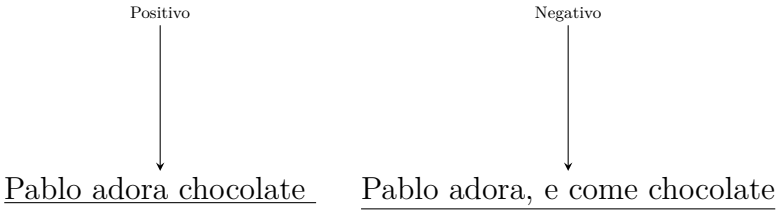
Figura 27 – Representação Neural distribuída da palavra Amor em inglês, espanhol e português.



Fonte: próprio autor

Dentre as propostas para aprendizado dos vetores usando métodos neurais destaca-se a de Collobert et al. (2011). A ideia geral da proposta é que uma palavra e seu contexto são um exemplo positivo, já uma palavra randômica presente nesse mesmo contexto é um exemplo negativo. Um exemplo dessa proposta pode ser verificado na Figura 28 onde a palavra "Pablo" aparece no contexto "adora chocolate" e, nesse exemplo, isto é classificado como positivo. Desse modo, toda aparição da palavra "Pablo" em contextos aleatórios será classificada como negativa, como a frase "Pablo adora, e come chocolate".

Figura 28 – Exemplos de treinamento positivo e negativo.



Fonte: próprio autor

Para Collobert et al. (2011), exemplos positivos possuem maior peso do que exemplos negativos. Para o cálculo destes pesos, eles propõem a construção de uma rede neural onde cada palavra é associada a um vetor de N dimensões, esse número de dimensões é definido por quem executa o algoritmo, mas é a quantidade de características que devem ser extraídas de uma palavra.

Idealmente, todas as palavras presentes na frase positiva do exemplo da Figura 28 formam uma matriz de palavras embutidas e os vetores dessas palavras são inicializados randomicamente. A matriz é descrita por $L \in R^{N \times |V|}$ ou graficamente como apresentado

no exemplo da Figura 29 para a sentença “o menino já comeu.”. Percebe-se que a matriz representa uma sentença onde podem ocorrer diversos vetores de representação denotados por V.

Figura 29 – Matriz de exemplo da sentença “O menino já comeu.”.

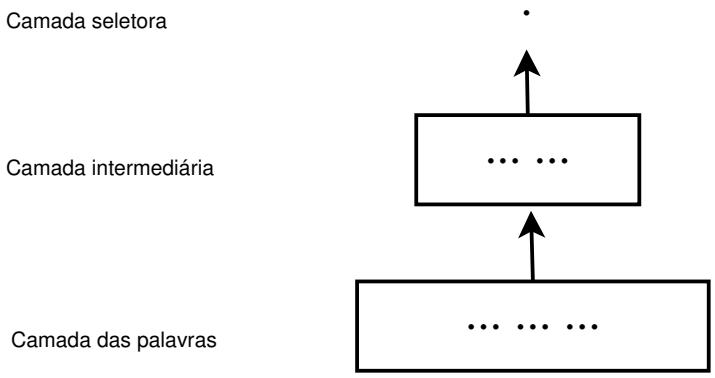
$$L = \begin{bmatrix} 0.15478 & 3.46834 & 6.13875 & -0.15348 \\ 8.43343 & 6.34435 & -2.55569 & 1.11234 \\ -0.34435 & 4.33454 & -5.99875 & -0.13443 \\ 6.53234 & -2.95465 & 0.95455 & 8.13443 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N \times |V|}$$

Fonte: próprio autor

Os vetores de características descritos nessa matriz são aprendidos por um modelo neural. Comumente essa matriz é denominada de *Look-up table*. Pode-se recuperar facilmente os vetores presentes nessa matriz já que uma coluna em L representa o vetor de uma palavra.

O objetivo do modelo neural é computar o valor do vetor x, o vetor de saída. Collobert et al. (2011) propõem a construção de um modelo neural para computar o valor de x onde as ativações neurais computariam o valor de saída de uma função usando métodos de regressão linear como *backpropagation*. Eles afirmam que uma rede neural *Feed-Foward* com três camadas é suficiente para computar o valor do vetor de características.

Figura 30 – Exemplo de um modelo neural de palavras.



Fonte: próprio autor

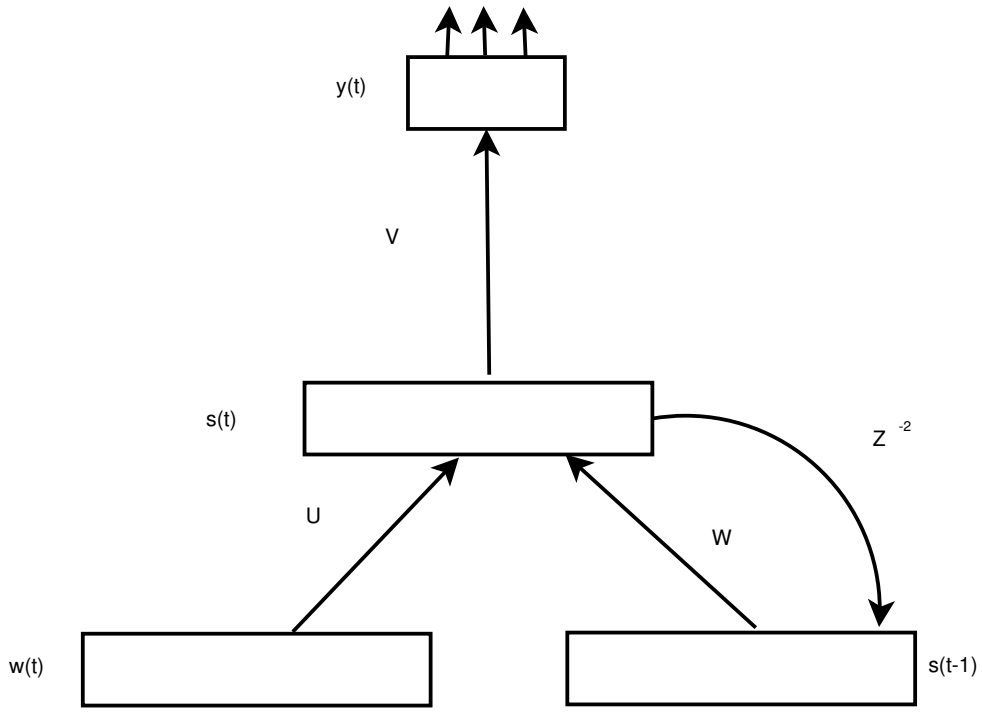
A Figura 30 representa o esquema em três camadas de como o modelo neural computaria o vetor x.

A representação de palavras na forma distribuída, feita através do treinamento de um modelo neural, tem-se mostrado inovadora como aponta a literatura (MIKOLOV

et al., 2013; COLLOBERT et al., 2011; SOCHER et al., 2013). O treinamento nestes modelos, como em quase todo método de aprendizado, se dá por exemplos (positivos ou negativos).

Mikolov et al. (2013) apresentam os melhores resultados atualmente para similaridades entre representações distribuídas. Nesse trabalho, os autores utilizam a mesma ideia[32] proposta por Collobert et al. (2011), mas com a utilização de um modelo neural recorrente. Em sua arquitetura recorrente (MIKOLOV et al., 2013), representada na Figura 31, o modelo neural proposto possui três camadas: uma de entrada, outra oculta com relações de recorrência com os pesos correspondentes das matrizes, e uma terceira onde são realizadas as classificações.

Figura 31 – Diagrama do modelo Neural Recorrente.



Fonte: (MIKOLOV et al., 2013)

Mikolov et al. (2013) explicam que o modelo possui como entrada o vetor $w(t)$ que representa a palavra no período t codificado. A camada $y(t)$ possui como saída a probabilidade distribuída das palavras. Já a camada escondida $s(t)$ traz a representação histórica da sentença[33]. Vale ressaltar que as dimensões das matrizes e das camadas de entrada/saída são iguais ao tamanho do vocabulário. Mikolov et al. (2013) explicam que os valores das camadas escondida e de entrada são computados pelas funções:

[32] Um modelo neural para aprender representações distribuídas de palavras.

[33] Isso é possível pois um modelo recorrente tem a "habilidade" de guardar o histórico por um longo período de treinamento.

$$s(t) = f(U_{w(t)} + W_{s(t-1)})$$
$$y(t) = g(V_{s(t)})$$

Nesse modelo, as palavras são representadas pelas colunas U, e W representa os pesos que são aprendidos em iterações anteriores. Já V representa os vetores de características que foram aprendidos através do algoritmo *backpropagation*, f e g são funções de composição.

Vale ressaltar que esse modelo consegue criar analogias testando as dimensões das matrizes de similaridades, simplesmente realizando operações algébricas. Isto é possível pois as matrizes aprendidas no modelo neural representam vetores de características sintáticas/semânticas das palavras. Como exemplo[34] tem-se:

$$vetor(rei) - vetor(homem) + vetor(mulher) = vetor(rainha)$$

A Tabela 4 tem o objetivo de comparar os modelos neurais de representação distribuída. Como aponta Mikolov et al. (2013), seu modelo foi treinado em um córpus de notíciais jornalísticas[35] com 320 milhões de palavras e um vocabulário de 82 mil palavras. Já o modelo de Collobert et al. (2011) foi treinado em outro córpus. Para permitir a comparação entre os modelos, para cada par de palavra alvo na relação, Mikolov et al. (2013) mediram a similaridade da relação de cada par de palavras com os pares ideais (MIKOLOV et al., 2013; COLLOBERT et al., 2011) em relação ao córpus de teste, e usaram essa medida como resultado final de comparação. Os métodos RNN e CW foram propostos, respectivamente, por (MIKOLOV et al., 2013) e (COLLOBERT et al., 2011).

Tabela 4 – Tabela comparativa entre os modelos de representação distribuída.

Modelo	**Diferença Máxima (%)**
RNN-80	0.389
RNN-320	0.408
RNN-640	0.416
RNN-1600	**0.418**
CW-10	0.363
CW-50	0.363

Fonte: adaptado de (MIKOLOV et al., 2013)

A medida utilizada para comparação dos modelos é a proposta no *SemVal 2012 - task 2*, Medindo similaridades relacionais (JURGENS, 2012). Os nomes presentes na

[34] Este exemplo, bem como o algoritmo de Mikolov, está disponível no site: https://code.google.com/p/word2vec/

[35] Detalhes das especificações podem ser conferidos em (MIKOLOV et al., 2013).

Tabela 4 estão divididos em tamanho das dimensões de treinamento. O modelo com 1600 dimensões proposto por Mikolov et al. (2013) obteve o melhor resultado e isso deve-se ao fato de apresentar uma grande concentração de características permitindo, assim, identificar corretamente as similaridades semânticas, como ele mesmo diz "à medida que aumentamos as dimensões, os resultados também aumentam".

Como apontam Socher, Manning e Bengio (2013), métodos neurais de representação distribuída de palavras embutidas, quando comparados a métodos como o LSA, têm se demonstrado mais significativos com o acréscimo de supervisão de uma ou múltiplas tarefas como por exemplo em análise de sentimento onde usualmente métodos não supervisionados não conseguem capturar ou expressar sentimentos.

A Tabela 5 apresenta resultados para a medida da correlação de *spearman*, aplicada em diferentes córpus, para os principais modelos existentes de representações distribuídas.

O trabalho proposto por Pennington, Socher e Manning (2014a), o *Glove*, utiliza o princípio proposto por Mikolov et al. (2013) para representações distribuídas de palavras através de modelos neurais recorrentes. Mais especificamente, Pennington, Socher e Manning (2014a) propõem uma modelagem através de janelas de contexto, de forma similar à Mikolov et al. (2013), mas eles utilizam uma matriz de ocorrência para diminuir a velocidade de processamento.

Como apontam Pennington, Socher e Manning (2014a), o modelo proposto por Mikolov et al. (2013) gasta muito tempo computando palavras que já foram vistas pelo modelo, o que acarreta um processamento extra em palavras que geralmente são associadas em algum contexto fixo. Pennington, Socher e Manning (2014a) citam como exemplo a palavra "the", que geralmente é associada a algum sujeito ou substantivo. Então, através do uso de uma matriz de ocorrências, o modelo "ignora" esses casos com base na frequência e refaz o cálculo para palavras muito frequentes.

Tabela 5 – Tabela comparativa entre os principais modelos de representação distribuída.

Modelo	WS353	MC	RG	SCWS	RW
RNN-80	68.4	79.6	75.4	59.4	45.5
GloVe	65.8	72.7	77.8	53.9	38.1

Fonte: adaptado de (PENNINGTON; SOCHER; MANNING, 2014a)

Outros modelos para representação distribuída de palavras vêm sendo propostos no decorrer dos anos, por exemplo, a ligeira modificação do modelo proposto originalmente em (MIKOLOV et al., 2013). No trabalho de Ling et al. (2015), a molagem final é feita por uma concatenação do vetor das palavras do contexto ao invés da tradicional

soma proposta por Mikolov et al. (2013). Essa alteração leva o modelo a extrair mais características morfossintáticas, uma vez que a ordem das palavras é fixa.

Recentemente, o trabalho de Joulin et al. (2016) demonstrou que é possível extrair características de palavras através de classificadores neurais, de forma a conseguir um desempenho similar ao estado da arte tendo ganhos de velocidade e processamento. Este ganho na velocidade de processamento se dá por uma pré-modelagem das palavras e pela modelagem das classes do problema em uma árvore de Huffman. Esta modelagem permite uma melhor busca pela classe ideal em um tempo muito inferior à tradicional soft-max.

Neste trabalho, representações distribuídas foram utilizadas para tratar palavras e etiquetas sintáticas/morfossintáticas. Estas representações possibilitam o aprendizado do classificador neural. Mais, especificamente a camada de entrada da rede neural utiliza-se de representações distribuídas afim de realizar classificações. Já para a modelagem das diferentes classes, optou-se por uma modelagem *one-hot*, visto que esta é uma modelagem mais gulosa, o que condiz com um sistema baseado por transições.

2.5.3 Deep Learning para análise sintática

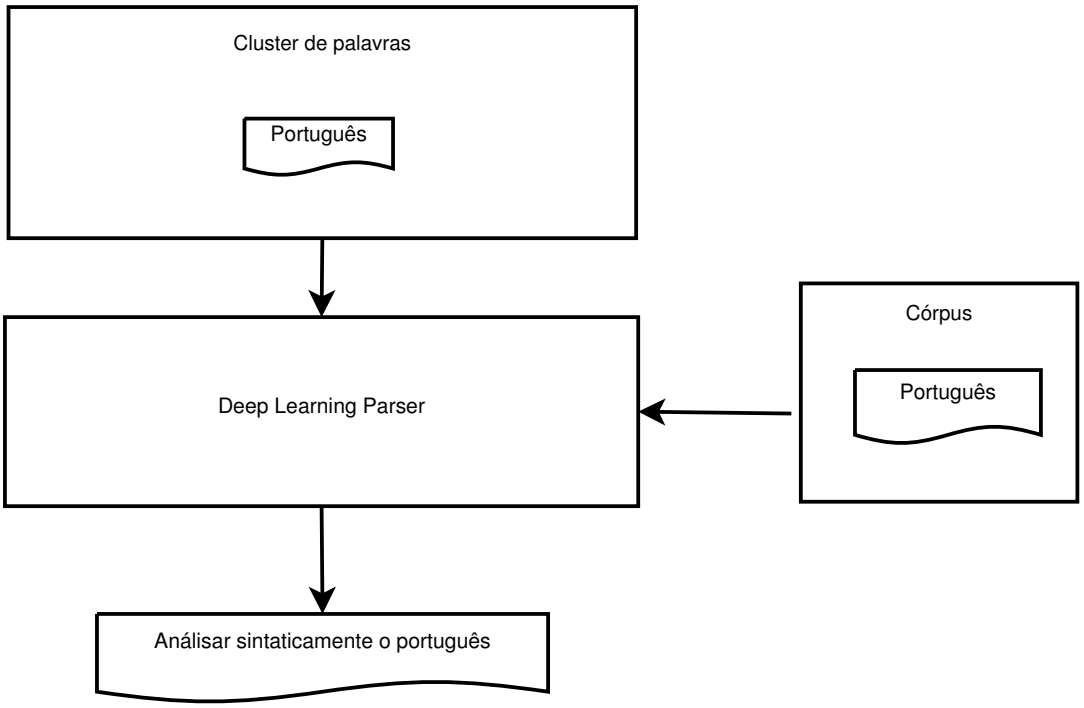
A tarefa de análise sintática também tem se beneficiado com os avanços nas pesquisas com representações distribuídas em redes neurais e *deep learning*. O principal foco atual em aprendizado profundo para análise sintática se dá em redes neurais recursivas para análise de constituintes (SOCHER et al., 2013) e um método neural guloso de classificação para análise de dependência baseado em transições (CHEN; MANNING, 2014).

A Figura 32 ilustra o processo de análise sintática usando uma rede neural. A ideia é utilizar um conjunto de representações distribuídas de palavras (*cluster* de palavras) e córpus com anotação sintática para extração das regras gramaticais. A combinação dessas entradas em um modelo neural torna possível analisar sintaticamente qualquer sentença, mesmo que alguma palavra da sentença de entrada não esteja nos córpus ou no agrupamento de palavras, o que só é possível graças à similaridade semântica de vetores embutida na representação distribuída. Por exemplo, se uma palavra não for encontrada, utiliza-se um método de aproximação de vetores de palavras que encontra uma representação similar para ela.

Outra característica que modelos neurais para análise sintática possuem é a capacidade de "fazer muito com pouco". Graças às representações distribuídas, é possível extrair características válidas para o aprendizado de constituintes por exemplo, ou ainda para a construção de *chunks* de regras como demonstra Collobert (2011).

A Tabela 6 apresenta alguns dos principais métodos para análise sintática neurais. Esses métodos estão divididos em métodos de constituintes e de dependência. Também

Figura 32 – Diagrama do modelo neural para análise sintática.



Fonte: próprio autor

estão presentes os córpus utilizados tanto para extração de regras quanto para o *cluster* de palavras, e também os respectivos autores e a taxa de acerto obtida nas avaliações por eles relatadas.

Tabela 6 – Tabela comparativa entre os modelos de análise sintática neural.

Modelo/autor	**Forma de análise**	**Córpus**	**Taxa de acerto**
CVG (SOCHER et al., 2013)	Constituintes	WSJ córpus	90.4% [36]
Chen (CHEN; MANNING, 2014)	Dependência	WSJ + Wikipédia	91.0% [37]

Fonte: próprio autor

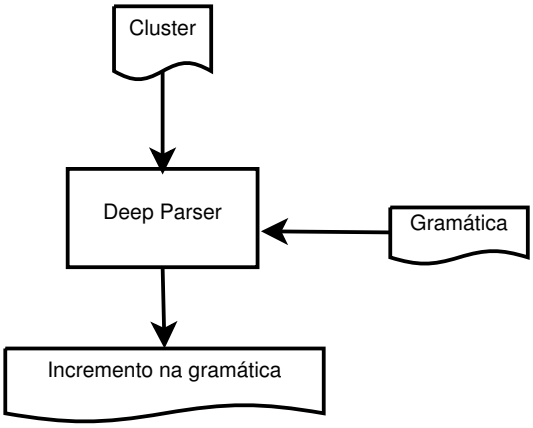
A proposta de Socher et al. (2013) utiliza um modelo neural por recorrência que tem como entrada um conjunto de vetores de palavras e uma gramática probabilística da língua fonte. A gramática é reutilizada para extração das regras no modelo neural. O conjunto de vetores de palavras é induzido a partir do córpus WSJ. Esse córpus, em especial a versão utilizada para construção dos vetores e indução da gramática, é um conjunto

[36] Este resultado se dá pois são consideradas todas as sentenças do córpus (sem limitação de tamanho).
[37] Este resultado representa o acerto das etiquetas sintáticas e seus respectivos termos dependentes.

de textos no formato Penn treebank com textos jornalísticos do inglês e cada sentença com sua anotação sintática e morfossintática. Esse córpus possui 15.000 sentenças.

A Figura 33 representa o modelo de análise sintática utilizado por Socher et al. (2013). Esse diagrama tem como entrada o modelo de representação distribuída[38]. Ele utiliza o córpus WSJ para o treinamento do vetor de palavras. A gramática também é induzida a partir do córpus WSJ. Como resultado, o modelo neural retorna uma gramática probabilística e esta é utilizada para o re-treinamento do modelo para analisar as sentenças.

Figura 33 – Diagrama do modelo neural para análise sintática de (SOCHER et al., 2013).



Fonte: adaptado de (SOCHER et al., 2013)

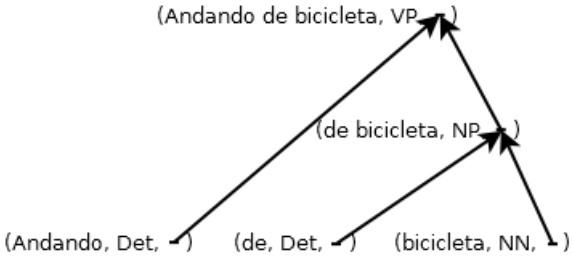
O processo de retreinamento tem como arquitetura uma rede neural que utiliza o princípio de composição de nós, que é a simples aplicação de uma função de composição. Socher et al. (2013) apontam que o modelo proposto utiliza uma função muito poderosa que combina diferentes frases com cabeças sintáticas (provenientes da gramática probabilística) e esse parâmetro, proveniente da composição, é dado como entrada para uma nova rede neural. Essa composição tem como objetivo extrair dessa gramática qual é a cabeça de regra, ou seja, qual é a etiqueta sintática.

Em seu modelo neural, Socher et al. (2013) propõem a simplificação do modelo em um nó com os pesos das estruturas sintáticas unidas. Os pesos de cada nó são condicionalmente dependentes em categorias de cada constituinte filho. Os autores apontam que essas diferentes funções de composição, quando combinadas com diferentes tipos de frases, apresentam uma grande melhora na precisão da decisão do analisador sintático. Graças à utilização de um modelo de representação distribuída de palavras, em combinação com o modelo neural proposto, é possível acertar a decisão e capturar a similaridade entre as frases/sentenças.

[38] O modelo utilizado está disponível em: https://code.google.com/p/word2vec/

A Figura 34 demonstra o funcionamento do vetor de composições, onde cada nó representa a tripla (sentença, categoria, vetor neural).

Figura 34 – Exemplo do funcionamento da análise de uma frase.



Fonte: (SOCHER et al., 2013)

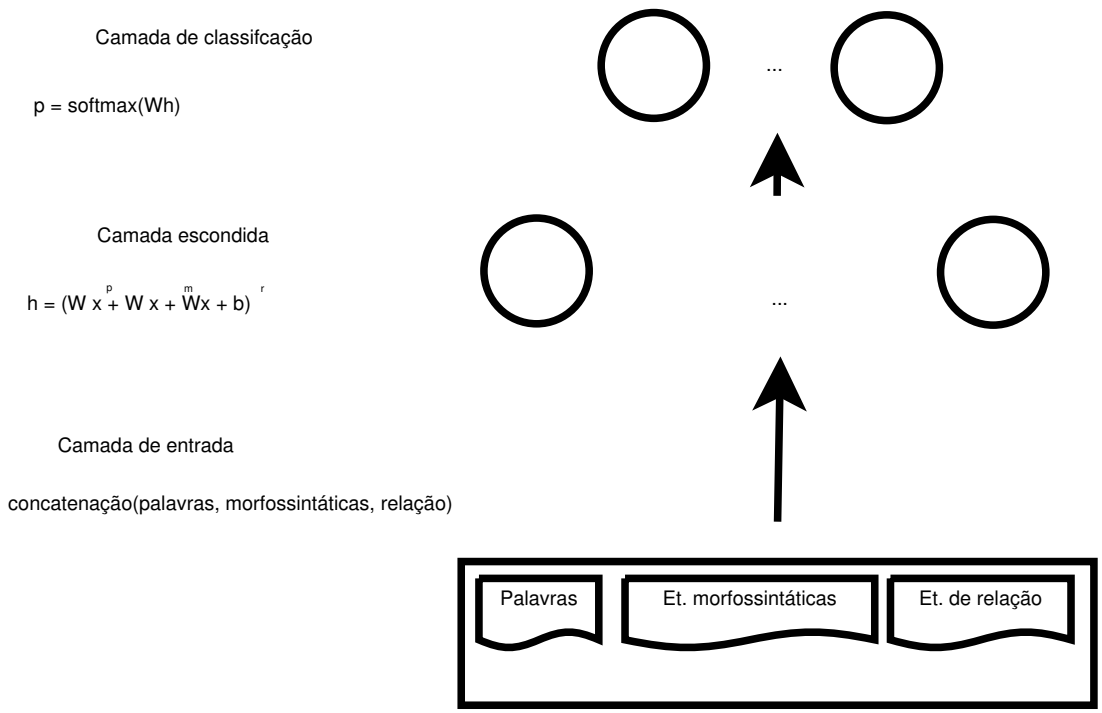
Em outro trabalho que utiliza redes neurais para análise sintática, só que desta vez de dependência, Chen e Manning (2014) propõem um modelo neural para classificação. O modelo é capaz de representar uma grande quantidade de características, através da representação distribuída, ao invés de simplesmente indicar qual é a melhor característica que representa o vetor do problema. Uma das grandes vantagens desse modelo é compactar, em uma representação distribuída, palavras, etiquetas morfossintáticas e relações de dependência.

Esse modelo de análise sintática de dependência baseia-se em (HALL; NILSSON, 2006) onde um modelo de transição é utilizado para a escolha de dependência. O modelo por transição é capaz de predizer uma sequência de transições a partir de alguma configuração inicial até algum terminal, o que é uma estrutura de dependência. Hall e Nilsson (2006) propõem um classificador para auxiliar esta tarefa. Esse classificador decide qual nó é derivado de qual com base em um conjunto de observações.

Ele também possui como entrada o conjunto de representações distribuídas de palavras, etiquetas morfossintáticas e sintáticas. Esses vetores são aprendidos utilizando técnicas de distância semântica, como os trabalhos de Mikolov, Yih e Zweig (2013) e Socher et al. (2013). Esse conjunto de vetores, como pode-se perceber pela Figura 35, é dado como entrada para o modelo neural de análise de dependência. Também percebe-se, por essa mesma figura, que o modelo possui três camadas, seguindo o modelo *Feed-Foward*, onde uma delas é a camada oculta. A camada oculta possui uma função de ativação h também denominada de função cúbica de ativação.

A função h tem como objetivo mapear valores, para posterior soma de pesos das entradas. A principal justificativa para a escolha dessa função é que, para análise sintática de dependência, faz mais sentido combinar vetores em pares, no caso uma tripla (representações de palavras, etiquetas morfossintáticas e etiquetas de relação). Outra

Figura 35 – Exemplo do funcionamento do modelo de (CHEN; MANNING, 2014).



Fonte: adaptado de (CHEN; MANNING, 2014)

vantagem que os autores apontam é que representações distribuídas podem ser replicadas para etiquetas morfossintáticas e sintáticas, similarmente ao que é feito para palavras.

O modelo apresenta uma camada de classificação ativada pela função p. Essa camada é responsável por retornar probabilidades de escolha de arcos de dependência, bem como etiquetas e palavras presentes.

O processo de extração da árvore sintática é feito de forma gulosa. A cada etapa são extraídas todas as palavras e etiquetas correspondentes de uma configuração c, computada na camada oculta através de $h(c)$, e selecionada a transição com maior probabilidade através da função de ativação $t = argmax_t W_i(t)h(c)$ e executada a transição $c \rightarrow t(c)$. A cada iteração, o algoritmo busca encontrar a melhor configuração da camada escondida. Isto é, ele encontra a melhor tripla ($W_{palavras}$, $W_{morfossintáticas}$, $W_{relação}$) e, uma vez encontrada, ele continua o modelo de transição.

O treinamento do modelo distribuído é feito a partir do córpus WSJ para etiquetas sintáticas e morfossintáticas. Os autores também apontam a necessidade de converter os córpus de treinamento e de teste para um modelo de dependência. O conjunto de treinamento possui 45 tipos de etiquetas morfossintáticas e também 17 tipos de etiquetas sintáticas. Já para as palavras, eles utilizaram uma fração da Wikipédia composta por 44 mil palavras. Os resultados em um modelo de dependência foram avaliados com

base na porcentagem de acerto com etiquetas e sem etiquetas considerando-se somente arcos de dependência. O resultado presente na Tabela 6, 90.7%, leva em consideração etiquetas. Esse resultado (para o inglês) representa o estado da arte para análise sintática de dependência.

Ambos os modelos, onde um é para análise por constituência (SOCHER et al., 2013) e outro de dependência (CHEN; MANNING, 2014), representam os principais modelos de análise sintática com modelos neurais existentes. Ambos os modelos representam o quão importante é a utilização de modelos simples de aprendizado e ilustram a grande vantagem de utilizar dados não rotulados provenientes da Web como a Wikipédia. Modelos com arquiteturas *deep* tornam-se cada vez mais necessários pela sua simplicidade e automatização na criação de atributos complexos.

O NNParser, diferentemente da proposta de Chen e Manning (2014), utiliza uma modelagem mais gulosa para a camada *soft-max* onde, através de uma modelagem *one-hot*, o modelo é capaz de realizar decisões gulosas. Também, a utilização do maior no número de camadas escondidas é uma diferença entre o modelo de Chen e Manning (2014) e o NNParser.

2.6 Análise de dependência baseada em transições

Sistemas de análise de dependência baseados em transição possibilitam a descrição simples de estados de análise sintática ao mesmo passo que especificam fortemente tipos de transições entre diferentes estados do histórico de decisões de análise sintática.

Em um analisador sintático baseado em transições, uma configuração é descrita como:

Stack $\mathbf{S} = [\mathbf{Null}]$

1. A pilha (*stack*) possui elementos que serão analisados e podem ser adicionados ou extraídos;
2. Esses elementos são palavras analisadas da sentença ou seja palavras que sofreram alguma operação.

Buffer $\mathbf{B} = [\mathbf{0, ..,n\text{-}1}]$

1. Armazena todas as palavras (representadas como nós do grafo) de uma sentença que serão analisadas.

Set of Arcs $\mathbf{A} = \{\}$

1. Conjunto de operações já realizadas durante o processo de análise.

A cada n operações, o sistema extrai elementos do *buffer* e da pilha, atualizando-os e decidindo qual é a melhor operação a ser realizada. Esse processo ocorre até que todos os elementos do *buffer* e da pilha sejam percorridos.

O objetivo final de um analisador sintático é extrair a estrutura sintática da entrada, após as n operações.

Em um analisador sintático baseado em transições, existem dois tipos de operações possíveis: (i) *arc-standard* e (ii) *arc-eager*. Neste trabalho, apenas a *arc-standard* é usada conforme descrito a seguir:

LEFT-arc (l)
1. Adiciona em A, a tupla (S_1, S_2), onde S_n representa os elementos presentes na pilha (S);
2. Remove o elemento S_2 de S;
3. Se $|S| >= 2$, atribui l como rótulo do arco.

RIGHT-arc (l)
1. Se $|S| >= 2$ então,
2. Adiciona em A, a tupla (S_2, S_1);
3. Remove o elemento S_1 de S;

SHIFT
1. Remove B_1 e adiciona no topo de S;

A partir desta definição de estado, pode-se propor a criação de um decisor. Um decisor no contexto de análise baseada em transições é um modelo capaz de predizer operações de análise, a partir de um histórico de decisões de análises sintáticas. Para tal, a construção de um modelo de decisão deve ser fortemente dependente de extração de atributos presentes nos estados.

Nesse sentido, Nivre (2003), Yamada e Matsumoto (2003) propuseram um modelo capaz de representar os diferentes estados de uma sentença durante o processo de análise. Nesse modelo pode-se extrair atributos particulares presentes em cada estado de configuração, os quais estão presentes em (B, S). A tabela 7 apresenta exemplos destes atributos.

Como pode-se perceber estes atributos são da forma n-grama, até tamanho 3. Este tamanho foi constatado por (ZHANG; NIVRE, 2011b) como ideal.

Utilizando um n-grama de tamanho igual à 3 são extraídos 48 atributos de um estado de analise sintática. Por exemplo, no "NNParser" elementos são dados de entrada do decisor neural cada elemento possui uma representação distribuída (MIKOLOV et al., 2013; COLLOBERT et al., 2011). Como demonstra (CHEN; MANNING, 2014) este tipo de representação permite diminuir a esparsidade associada a atributos de um estado de análise por transição. A Tabela 8 demonstra o processo de transição entre estados de análise.

Tabela 7 – Exemplo de atributos que podem ser extraídas de uma configuração.

n-grama de atributos
$s1.w$; $s1.t$; $s1.wt$; $s2.w$; $s2.t$; $s2.wt$; $b1.w$; $b1.t$; $b1.wt$
$s1.wt \circ s2.wt$; $s1.wt \circ s2.w$; $s1.wts2.t$; ...
$s2.t \circ s1.t \circ b1.t$; $s2.t \circ s1.t \circ lc1(s1).t$; ...

Fonte: (CHEN; MANNING, 2014)

Tabela 8 – Descrição das operações de análise entre os $estado_t$ até $estado_{t+1}$, para as aplicações das operações LEFT-arc e SHIFT as quais extrai o segundo elemento do topo da pilha, adicionando uma nova operação em $D = add(l(b_2, s_1))$, e retira um elemento do *buffer* colocando-o no topo de S, respectivamente. A operação RIGHT-arc, similar à operação LEFT-arc que retira o elemento do topo de S.

Buffer | Pilha | Dependency Set | Operação

$estado_t = n$	$estado_t = n + 1$
B=$\{b_1, ..., b_n\}$ S=$\{s_1, s_2\}$ D=$\{r(s_0, b_0)\}$ SHIFT	B=$\{b_2, ..., b_n\}$ S=$\{S_1\ \}$ D=$\{..., l(b_2, s_2)\}$ LEFT-arc

Fonte: próprio autor

Uma vez que os atributos são extraídas para cada sentença, o decisor é inicializado com todas as amostras extraídas[39]. Gerando assim como resultado, ao final do processos de treinamento um sistema e analise sintática baseado em transições.

Em nosso modelo foi adotado um sistema de transições através de um decisor neural. Esta escolha deve-se ao fato de um sistema baseado em transições ser rápido, isto é, ser capaz de manter um processamento linear de acordo com a entrada.

[39] Por exemplo: se cada sentença no córpus tem 20 amostras de 48 elementos cada e, o córpus possuí 1000 sentenças, temos 20000 amostras para o córpus.

3 Trabalhos relacionados

Este capítulo busca oferecer uma visão das pesquisas relacionadas ao tema de análise de dependência neural para sistemas baseados em transições. Sistemas baseados em transição vêm sendo cada vez mais utilizados para a análise sintática por sua alta velocidade e capacidade de analisar sentenças sem um alto custo computacional associado ao tamanho da sentença (ZHANG; NIVRE, 2011b; CHEN; MANNING, 2014). Esta velocidade de processamento está associada ao modo como as amostras de treinamento são extraídas. Mais especificamente, o processo de treinamento extrai amostras baseado no conjunto de operações aceitas no analisador (*arc-standard* ou *arc-eager*) e na forma de funcionamento do processo de análise de dependência (*oracle*).

Com o objetivo de organizar as comparações, o capítulo as divide em quatro seções. A primeira seção comenta os trabalhos que tratam de sistemas baseados em transição que utilizam *oracle* estático. Já a segunda seção, trata de sistemas baseados em transição que utilizam *oracle* dinâmico e outras formas de arquiteturas neurais. A penúltima seção traz as propostas para a análise sintática multilíngue. E, por fim, a última seção apresenta um levantamento de alguns trabalhos relacionados de análise neural de dependência baseada em grafos.

3.1 Análise de dependência baseada em transições e o uso do *oracle* estático em redes neurais profundas

O funcionamento de um *oracle* estático é descrito em (ZHANG; NIVRE, 2011b; HALL; NILSSON, 2006; NIVRE, 2003; YAMADA; MATSUMOTO, 2003) como uma implementação direta de um grafo direcionado. Isto é, todas as amostras extraídas no processo de treinamento descrevem exatamente o grafo da sentença, sem nenhum "desvio" ou sem nenhuma operação extra que seja aceita como operação possível para representar fidedignamente o grafo.

Recentemente, alguns modelos neurais que utilizam sistemas baseados em transição com *oracle* estático foram propostos para a análise sintática de dependência, tais como (CHEN; MANNING, 2014) e (DYER et al., 2015).

A proposta de Chen e Manning (2014) é o tipo de arquitetura básica que definiu o funcionamento de um modelo de análise de dependência neural. Essa arquitetura marcou por apresentar soluções inéditas no uso redes neurais profundas como a utilização de janelas de *embedding* para representar os diferentes atributos extraídos durante o processo de treinamento ou, ainda, a primeira representação distribuída de etiquetas morfossin-

táticas e de arcos de relação. Apesar de ser um modelo neural simples, uma MLP, esta arquitetura obteve resultados interessantes no córpus PTB: 92% de UAS ultrapassando, assim, os 89,9% do tradicional malt-parser (ZHANG; NIVRE, 2011a).

Pode-se dizer que a principal contribuição de Chen e Manning (2014) está em demonstrar que é possível representar etiquetas morfossintáticas e de arcos de relação da mesma maneira que é possível representar de forma distribuída as palavras. Mas, esse trabalho ainda é bastante dependente de atributos extraídos manualmente durante o processo de treinamento: o problema das *hand-crafted features* (CHEN; MANNING, 2014; DYER et al., 2015; BALLESTEROS et al., 2016). Chen e Manning (2014) utilizaram o mesmo modelo proposto por (ZHANG; NIVRE, 2011a; ZHANG; NIVRE, 2011b), que é um modelo de representação de estados de análise muito esparso e específico, o que torna a representação pouco confiável.

Para contornar este problema, Dyer et al. (2015) e suas modificações posteriores (AMMAR et al., 2016a; BALLESTEROS et al., 2016) propuseram a utilização de máquinas de memórias para capturar atributos de cada estado de análise de forma automática. Esta captura se dá pois o modelo neural proposto por Dyer et al. (2015) utiliza uma arquitetura recorrente, onde estados de análise de dependência são armazenados na memória da rede possibilitando que estados futuros utilizem informações de estados anteriores. Essas informações são os atributos da configuração destes estados como: elementos da *pilha*, elementos do *buffer* e elementos do conjunto de operações já realizadas.

Esta modelagem, diferente proposta por Chen e Manning (2014), tem como vantagens extrair a maior quantidade possível[1] de atributos de uma configuração de análise, ao invés de 48 atributos como o utilizado em (CHEN; MANNING, 2014). Possibilitando, assim, ao modelo "adequar" os atributos aos diferentes cenários do treinamento, reduzindo a esparsidade dos dados inerente dos atributos propostos em (ZHANG; NIVRE, 2011a; ZHANG; NIVRE, 2011b).

3.2 Análise de dependência baseada em transições e o uso do *oracle* dinâmico em redes neurais profundas

Outra forma de extração de amostras de treinamento é através do uso de *oracle* dinâmico. Esta forma de extração caracteriza-se por criar amostras "extras" de treinamento com o objetivo de gerar possíveis caminhos no grafo. Como um grafo não determinístico, o *oracle* dinâmico gera possibilidades de transições até que o conjunto de operações correto seja alcançado.

O uso de um *oracle* dinâmico deve vir acompanhado de técnicas de busca para

[1] Este número depende do tamanho das unidades escondidas.

tornar o espaço de busca melhor dimensionado e otimizado. Diversas técnicas de busca foram utilizadas para modelos não neurais, como o *beam search* (GOLDBERG; NIVRE, 2012; BALLESTEROS; NIVRE, 2013).

Goldberg e Nivre (2012) demonstram que o uso de um sistema de *oracle* dinâmico, em conjunto com um sistema de busca, maximiza o desempenho do sistema baseado em transições. A ideia geral é gerar amostras extras de treinamento, com o intuito de aumentar o conjunto de amostras de treinamento, para que, através do uso do algoritmo de busca, este espaço "maior" proporcione melhores decisões de análise.

Alguns trabalhos recentes implementaram estas técnicas de busca em redes neurais profundas tais como (BALLESTEROS et al., 2016), (ANDOR et al., 2016) e (COPPOLA; PETROV, 2015). Os trabalhos de Andor et al. (2016) e Coppola e Petrov (2015) utilizam a arquitetura neural proposta por Chen e Manning (2014) como base, mas adicionam modelos de predição estruturada (*Conditional Randon Fields* e *Structured Perceptron*, respectivamente) na saída da rede neural, agregando mais informações durante o processo de decisão de análise.

Além do uso de predições estruturadas através de redes neurais profundas, Andor et al. (2016) e Coppola e Petrov (2015) usam o algoritmo *beam search* para dimensionar e otimizar a busca das diferentes opções de análise.

3.3 Análise de dependência multilíngue baseada em transições e redes neurais profundas

Modelos de análise de dependência multilíngues, sejam neurais ou não, têm como principal problema as diferenças de domínio. Seja uma diferença em nível de palavras (MCDONALD et al., 2005; MCDONALD; PETROV; HALL, 2011) ou em nível morfossintático ou de relações (NIVRE et al., 2016a; PETROV; DAS; MCDONALD, 2012; GUO et al., 2015; AMMAR et al., 2016a), o tratamento se dá através do uso de um dicionário multilíngue ou da construção de recursos universais.

A utilização de técnicas de mapeamento interlínguas para modelos neurais é debatida em (FARUQUI; DYER, 2014) e (GUO et al., 2015). Estes autores demonstram que a combinação de representações distribuídas multilíngues leva a resultados superiores aos de uma representação monolíngue. Neste sentido, Guo et al. (2015) propõem o uso de representações distribuídas multilíngues como técnica para melhorar o desempenho em línguas morfologicamente ricas. Mas, como Faruqui e Dyer (2014) e Guo et al. (2015) realizam mapeamentos em nível de palavras, o modelo de Guo et al. (2015) carece de tratamento em nível morfossintático ou de relações.

Para contornar este problema, o uso de uma notação universal para córpus se faz

necessária. Nesse sentido, o uso de uma analisador sintático multilíngue para modelos profundos é factível tanto que em (AMMAR et al., 2016a) tal modelo obtém os melhores resultados para o córpus UD. Mais especificamente, a proposta de Ammar et al. (2016a) utiliza técnicas de redes neurais recorrentes e um refinamento no nível morfossintático. Este refinamento, como aponta Tiedeman (2015), é crucial para o bom desempenho de qualquer analisador que utilize como base recursos homogêneos.

Também destaca-se a forma como Ammar et al. (2016a) realizaram o treinamento do modelo via concatenação universal de recursos, isto é, nos experimentos eles demonstram que a combinação de recursos homogêneos durante o treinamento faz com o que o modelo obtenha desempenhos superiores para cada língua em comparação com um treinamento e teste monolíngue.

3.4 Análise de dependência neural baseada em grafos

Um outro tipo de análise de dependência muito utilizado é a análisde de dependência baseada em grafos. Similarmente à análise baseada em transições, na análise baseada em grafos a sentença de entrada também é representada como um grafo direcionado, mas diferentemente do que ocorre na análise baseada em transições, o classificador deve extrair atributos do grafo e não de uma configuração. Essa diferença leva a um maior tempo de computação para extrair os atributos que descrevem exatamente as transições de um grafo. Alguns atributos comumente extraídos são: pré-fixos (se existe alguma transição que chega na palavra alvo), pós-fixos (se existe alguma transição que sai da palavra alvo) e os próprios atributos da palavra alvo (etiqueta morfossintática, se é uma palavra com letra maiúscula, etc).

Esse tipo de análise de dependência, similarmente à análise baseada em transições, vem recebendo grande destaque com o advento das redes neurais profundas (WANG; BAOBAO, 2016b; WANG; BAOBAO, 2016a; ZHANG; ZHAO; QIN, 2016; FONSECA; ALUÍSIO, 2015).

Um dos primeiros trabalhos a realizar análise de dependência baseada em grafos para redes neurais convolucionais foi (FONSECA; ALUÍSIO, 2015). Nesse trabalho, os autores propõem uma arquitetura convolucional com uso de representações distribuídas de palavras pré-treinadas para o treinamento de um classificador capaz de predizer arcos de dependência entre sentenças. Vale destacar que foi um dos primeiros trabalhos a utilizar atributos de distância de palavras como *token* de entrada para a rede neural, tanto a distância para a cabeça da sentença como a distância para outras palavras.

Essas distâncias são referenciadas através de índices para representações distribuídas. Por exemplo, um grafo onde a palavra “a” fica a uma distância de 3 até a cabeça da sentença o índice correspondente será atribuído à representação distribuída para que

indique a correspondência desse índice (*look-up*).

Similarmente a redes estilo *Feed-Foward*, as redes convolucionais também sofrem com o problema de longas dependências: as decisões de análise não levam em conta dependências longas entre palavras. Isso ocorre porque, como o algoritmo "anda" na sentença, as operações anteriores acabam sendo "esquecidas". Para contornar este problema, modelos que utilizam redes recorrentes foram empregados. Mais especificamente, redes LSTM (WANG; BAOBAO, 2016b; WANG; BAOBAO, 2016a; ZHANG; ZHAO; QIN, 2016), as quais têm a capacidade de manter informações importantes que foram aprendidas no passado.

Wang e Baobao (2016a) utilizam uma arquitetura bi-LSTM para capturar longas dependências entre sentenças e diminuir a necessidade de hiperparâmetros a serem aprendidos. Uma arquitetura bi-LSTM tem a capacidade de "andar" na memória para "manter" dependências, ou até mesmo "esquecer" dependências desnecessárias. Vale destacar que (WANG; BAOBAO, 2016a) foi um dos primeiros trabalhos a utilizar atributos de segmento como entrada da rede neural. Isto é, informações de segmento são atributos que descrevem relações de sufixo, infixo e pós-fixo, e este tipo de informação é crucial para um bom funcionamento de um analisador de dependência baseado em grafos. Wang e Baobao (2016a), Wang e Baobao (2016b) realizam a extração destes atributos com base na capacidade de "andar" na memória LSTM.

Wang e Baobao (2016a), Wang e Baobao (2016b) também apresentam formas interessantes e efetivas de se realizar análise de dependência, tanto que obtêm resultados aproximados ao estado da arte para o córpus WSJ $93,51\%$ de UAS e $92,45\%$ de LAS, resultados estes próximos a modelos como (ANDOR et al., 2016; COPPOLA; PETROV, 2015; WEISS et al., 2015).

4 O NNParser

Como mencionado anteriormente, este trabalho tem como objetivo propor, implementar e avaliar um novo modelo neural para análise sintática universal. Isto é, um modelo capaz de analisar sentenças em qualquer linguagem (que tenha um córpus de dependência) e mantenha um desempenho médio para todas as línguas. Para isso, utilizou-se como base o modelo de dependência baseado em transição proposto por Nivre (2003) e aplicado em (CHEN; MANNING, 2014) para redes neurais. Esse tipo de sistema se caracteriza por decisões gulosas sobre estados de configurações. Com base nessas características, propomos o NNParser: um modelo mais guloso e poderoso para análise de dependência no âmbito mono e multilíngue.

Para melhor clareza na apresentação deste trabalho, optou-se por dividi-lo em duas propostas: (i) análise sintática monolíngue e (ii) análise sintática multilíngue. Ambas utilizam córpus de dependência e, ao final, são avaliadas da mesma forma (com base no número de arcos corretos que elas geram).

A principal diferença entre as duas propostas está na forma como os dados são extraídos, ou seja, como a representação distribuída é aprendida. Em ambas as propostas, as representações distribuídas das palavras são induzidas utilizando a ferramenta *word2vec*, mas na análise sintática multilíngue combinam-se características extras de diferentes línguas para produzir a representação gerando, assim, uma representação multilíngue (FARUQUI; DYER, 2014; AMMAR et al., 2016a). As representações multilíngues são geradas com base na técnica de correlações canônicas, na qual dois conjuntos de representações multilíngues (em duas línguas) são combinados por meio de alguma característica em comum (dicionários bilíngues).

Ambas as propostas utilizam técnicas recentes dentro do PLN como *parser* baseado em *deep learning* (CHEN; MANNING, 2014; DYER et al., 2015; WEISS et al., 2015) e representações distribuídas mono e multilíngue (MIKOLOV et al., 2013; PENNINGTON; SOCHER; MANNING, 2014b; FARUQUI; DYER, 2014).

Na próxima seção (4.1) será apresentado o NNParser, o *parser* de dependência baseado em transições implementado neste trabalho. Em seguida (seção 4.2), será explicada a arquitetura multilíngue do NNParser.

4.1 NNParser: um *parser* neural baseado em transições

O NNParser é um sistema de análise (*parser*) de dependência baseado em transições que utiliza técnicas de aprendizado profundo (*deep learning*) em seu núcleo utilizando

representações lexicalizadas. Assim, o NNParser utiliza camadas inferiores ocultas para extrair representações compactadas da entrada com o objetivo de extrair a estrutura de dependência de uma sentença.

Um sistema baseado em transições caracteriza-se por escolhas gulosas de operações. Estas operações atuam sobre um conjunto de estados, onde o estado inicial contém a sentença que pretende-se analisar e o estado final, a estrutura de dependência desta sentença. Ao final, o histórico de operações realizadas é o conjunto de decisões tomadas pelo *parser* para obter a estrutura de dependência da sentença.

Além disso, o NNParser segue o modelo proposto inicialmente por Nivre (2003), e posteriormente aplicado em redes neurais profundas por Chen e Manning (2014) no qual sentenças não-projetivas são descartadas. Esse tipo especial de sentença caracteriza-se pelo cruzamento de arcos de dependência, o que gera um problema exponencial no número de combinações possíveis. Para "contornar" este problema, inúmeras técnicas de pré-processamento foram propostas como: árvores minímas (MCDONALD; PETROV; HALL, 2011) e fatoração (MCDONALD; PETROV; HALL, 2011). Por essa razão, sentenças não-projetivas são descartadas no NNParser.

O NNParser possui uma arquitetura muito simples se comparada à arquitetura dos modelos por recorrência. Ele não é capaz de "memorizar" ações para auxiliá-lo posteriormente em decisões de análise sintática como em (DYER et al., 2015; AMMAR et al., 2016a), mas, similarmente a (WEISS et al., 2015; CHEN; MANNING, 2014), utiliza atributos para descrever os estados.

Uma vez que esses atributos descrevem elementos presentes em cada estado, o NNParser é capaz de escolher as operações de análise sintática com base nos estados gerando, assim, um mapeamento entre operações e estados possíveis da estrutura de dependência da sentença.

Esses atributos descrevem o real estado do analisador sintático, uma vez que representam elementos presentes na pilha e no *buffer*. A escolha por atributos adequados é importante para qualquer tarefa do aprendizado de máquina, uma vez que eles consigam descrever a entrada perfeitamente, o decisor realizará decisões ótimas (NIVRE, 2014). Por este motivo, o NNParser utiliza o mesmo conjunto de atributos propostos por (ZHANG; NIVRE, 2011a).

Zhang e Nivre (2011a) também discutem o número ideal de atributos e concluem que o número ideal é 48. Exemplos desses atributos estão descritos na Tabela 9. O tamanho de 48 atributos, é dito como ideal por (ZHANG; NIVRE, 2011a) pois é capaz de representar com exatidão dependências entre atributos do estado de análise baseada em transições.Zhang e Nivre (2011a) demonstra que não é necessário um tamanho maior que 3 para qualquer tupla de atributos e dão o exemplo de $s2.t \circ s1.t \circ lc1(s1).t$ onde o modelo

proposto por eles consegue representar atributos presentes em elementos da pilha e extrair as respectivas etiquetas sintáticas da relação envolvendo outros elementos já analisados.

Tabela 9 – Exemplo de atributos que podem ser extraídos de uma configuração.

n-grama de atributos
$s1.w$; $s1.t$; $s1.wt$; $s2.w$; $s2.t$; $s2.wt$; $b1.w$; $b1.t$; $b1.wt$
$s1.wt \circ s2.wt$; $s1.wt \circ s2.w$; $s1.wts2.t$; ...
$s2.t \circ s1.t \circ b1.t$; $s2.t \circ s1.t \circ lc1(s1).t$; ...

Fonte: adaptado de (CHEN; MANNING, 2014)

Atributos em analise sintática baseada em transições, descrevem estados de análise sintática. Os possíveis tipos de atributos existentes são:

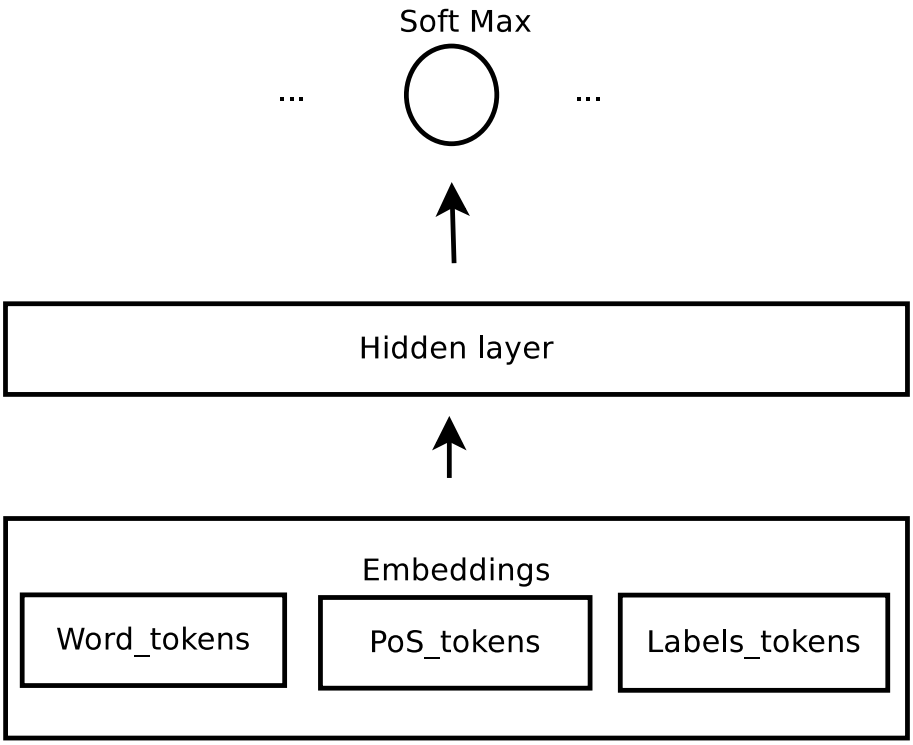
- **atributos de pilha** (s_n) – Elementos encontrados na pilha que podem ser: palavras, etiquetas morfossintáticas e etiquetas de transição.
- **atributos de *buffer*** (b_n) – Elementos encontrados no *buffer* que são: palavras e as respectivas etiquetas morfossintáticas.
- **atributos do histórico de análise sintática** (L-R pais) – Também pode-se extrair elementos presentes no histórico de operações já realizadas como: nós pais de uma palavra alvo, e nó pai do pai (esquerda e direita).

Os atributos de estados têm tamanho igual à 3 (ZHANG; NIVRE, 2011a). Isto é, esses atributos são compostos três à três e atributos não existentes são indicados por "NULL". Por exemplo, o atributo $s1.wt \circ s2.wt \circ b1.wt$ representa, respectivamente: a palavra no topo da pilha ($s1.wt$), a segunda palavra no topo da pilha ($s2.wt$) e a primeira palavra do *buffer* ($b1.wt$).

A Figura 36 representa a arquitetura neural do "NNParser" que descreve as diferentes entradas para cada etapa do processo de treinamento e é composta por atributos que descrevem estados de análise sintática. O objetivo final do modelo é maximizar a entropia cruzada (*cross-entropy*) das operações de análise sintática douradas, isto é, as operações que descrevem um estado de análise baseado em transição.

Mais especificamente, o NNParser utiliza os atributos como entrada de sua rede neural. Para tanto, constrói uma janela de *embeddings* (MIKOLOV et al., 2013; COLLOBERT et al., 2011) responsável pela representação de cada atributo por meio de sua respectiva representação distribuída. Vale ressaltar que o modelo neural utiliza três janelas diferentes para tratar cada tipo de elemento: uma janela para palavras, de tamanho 100;

Figura 36 – Arquitetura neural do NNParser.



Fonte: próprio autor

uma janela para etiquetas morfossintáticas, de tamanho 50; e uma janela para etiquetas sintáticas, de tamanho 50.

A cada iteração do processo de treinamento do modelo neural, camadas escondidas são ativadas para extração de características distribuídas. O processo de extração de características é feito através de regressões lineares.

O conjunto de treinamento é extraído através do uso de oracle estático. Como descreve Nivre (2014), o oracle estático é um processo capaz de mapear todos os possíveis estados e operações para uma sentença presente no conjunto de treinamento. Isto é, um algoritmo que simula o processo de análise sintática a fim de gerar o "conjunto dourado" (*gold standard*) de atributos, operações e os possíveis estados de análise sintática.

Sobre os parâmetros utilizados no modelo neural, para redução do *over-fitting* utilizou-se um *dropout* fixado em 0,5 na camada de representações distribuídas (camada de entrada). O mesmo parâmetro é utilizado em diversas outras tarefas de PLN (DYER et al., 2015; MIKOLOV et al., 2011; CHEN; MANNING, 2014). Como método de convergência utilizou-se o "adagrad" com parâmetro de aprendizado em $1 \times \epsilon^{-9}$, e como função de ativação para as unidades escondidas utilizou-se a função "relu".

Uma vez modeladas as entradas da rede neural e seu funcionamento, o modelo necessita gerar decisões de análise sintática a fim de mapear estas operações para os

respectivos estados no conjunto de estados da análise sintática. Para tanto, utilizou-se uma camada *softmax* de forma gulosa, dividindo as possíveis operações de análise sintática na forma de vetores *one-hot*. Vetores *one-hot* são vetores binários onde cada amostra tem um tamanho igual a 2^N onde N é o número total de classes ou operações possíveis. Estas operações dependem do número de possíveis relações de dependência existentes no córpus de treinamento.

De acordo com Nivre (2003), um sistema de transição da forma *arc-standard* tem como possibilidade $2 \times labels + 1$. A multiplicação por dois significa os dois tipos de relações de dependência existentes: *left-arc* e *right-arc*, e a adição de 1 é a operação de *shift*.

A probabilidade de cada operação é definida pela equação:

$$P(y) = exp\{W_y^T h_i + b_y\}$$

onde, W_y^T é o vetor de pesos da camada escondida i.

Um vetor *one-hot* para a camada *softmax* é uma forma diferente das propostas existentes para sistemas neurais de análise sintática por transição (CHEN; MANNING, 2014; DYER et al., 2015). Normalmente, a modelagem utilizada é um vetor de pesos (representação distribuída) para as classe existente (CHEN; MANNING, 2014; DYER et al., 2015), e esta escolha acarreta em um menor determinismo na escolha da classe, uma vez que é necessária a conversão para obter-se a probabilidade (extrair a exponencial da representação distribuída). Então, optou-se por uma modelagem *one-hot* onde classes são modeladas como vetores binários, o que torna o modelo mais guloso.

Esta escolha gulosa é uma extensão dos métodos propostos anteriormente para decisores não neurais (HALL; NILSSON, 2006; ZHANG; NIVRE, 2011a; NIVRE, 2003) em sistemas baseados em transição. Um método guloso é capaz de manter o tempo de processamento linear em função do tamanho da sentença de entrada. Já a decisão de modelar a camada *soft-max* através de representações *one-hot* se deu por questões puramente de desempenho e de velocidade de processamento.

O modelo apresentado do NNParser foi testado e seus resultados estão expostos no Capítulo 5 de três formas distintas: (i) *baselines* de diferentes configurações, onde os melhores parâmetros para a atual arquitetura são definidos; (ii) monolíngue, onde avalia-se uma língua individualmente (teste e treino); (iii) multilíngue de duas formas – (iii-A) com transferência de linguagem utilizando correlações canônicas e (iii-B) utilizando concatenação de córpus homogêneos.

4.2 NNParser: Mono ou Multilíngue

O modelo exposto na seção 4.1 foi utilizado para análise sintática mono e multilíngue neste trabalho. Um mesmo modelo utilizado para análise multilíngue pode, com algumas modificações, ser utilizado também para análise monolíngue.

Um analisador sintático de dependência deve ser capaz de analisar uma sentença em uma língua e extrair uma representação de dependência para ela. Mas, pela não existência de córpus homogêneos, ou pelas diferenças sintáticas, semânticas entre línguas não similares, um modelo multilíngue deve valer-se de técnicas diferentes de um modelo monolíngue, tais como, transferência de linguagem (MCDONALD; PETROV; HALL, 2011; TACKSTROM; MCDONALD; USZKOREIT, 2012) e uso de dicionários (GUO et al., 2015).

Técnicas para análise sintática multilíngue partem do princípio de que é possível mapear diferentes línguas a fim de combiná-las para aumentar o desempenho do analisador sintático.

Então, para habilitar a forma monolíngue do NNParser é necessária a utilização de um córpus de dependência que siga os moldes de notação sintática propostos por Nivre et al. (2016b), Schuster e Manning (2016). Essa notação sintática caracteriza-se por expressar sentenças de dependência de forma rígida no documento (de forma padronizada), universalidade no conjunto de etiquetas entre línguas (alguns casos existem etiquetas específicas, por exemplo classes de verbo muito específicas do português), e fácil acesso e possibilidade de comparação com diversos modelos do estado da arte (BALLESTEROS et al., 2016; AMMAR et al., 2016a; DYER et al., 2015). Também pode-se utilizar um córpus com notação por constituência desde que seja utilizada alguma técnica de conversão para os modelos universais citados, como a ferramenta Penn2Malt[1].

Uma vez que o córpus esteja no formato aceito, o NNParser é capaz de extrair amostras de treinamento em qualquer língua presente no conjunto de línguas do córpus. Estas amostras estão na forma de atributos como descrito na Tabela 9. Como o modelo de atributos é o mesmo proposto por Zhang e Nivre (2011a), é possível extrair um total de 48 atributos por estado de análise sintática.

O processo de treinamento de um modelo, seja ele mono ou multilíngue no NNParser, é descrito pelo *oracle*. Ele é capaz de extrair a melhor operação de análise sintática (operação dourada) para qualquer estado possível de análise sintática.

Uma vez que o processo de treinamento é iterativo para todos os grafos presentes no conjunto de treinamento, é possível extrair todas as operações douradas de uma sentença e possibilitar o mapeamento entre atributos que descrevem um estado de análise sintática e a

[1] <http://stp.lingfil.uu.se/~nivre/research/Penn2Malt.html>

respectiva operação dourada. Então, este mapeamento gera uma amostra de treinamento para cada estado possível da sentença. Como exemplo, a Tabela 10 descreve 3 amostras para uma sentença qualquer.

Tabela 10 – Exemplo de amostras de treinamento extraídas de uma sentença qualquer.

Nro. da amostra	**N-grama de atributos**	**Operação dourada**
1	$s1.w$; $s1.t$; $s1.wt$; $s2.w$; $s2.t$; $s2.wt$; $b1.w$; $b1.t$; $b1.wt$	*Shift*
2	$s1.wt \circ s2.wt$; $s1.wt \circ s2.w$; $s1.wt \circ s2.t$; ...	*Left-arc(aux)*
3	$s2.wt \circ s1.wt$; $s1.wt \circ s2.w$; $b1.wt \circ b2.t$; ...	*Right-arc(prop)*

Fonte: próprio autor

O número total de amostras é descrito pelo somatório de todas as sentenças (m) multiplicado pelo número de amostras extraídas pelo *oracle* (n), que é descrito pela função: $f(x) = \sum_{n=1}^{sentencas} n_n \times m_n$, onde x é o córpus.

Uma vez extraídas todas as ações *oracle*, o NNParser alimenta o decisor neural. Como descrito na seção 4.1, janelas de *embeddings* são utilizadas para o mapeamento entre amostras de treinamento e vetores de representações distribuídas (COLLOBERT, 2011; TURIAN; RATINOV; BENGIO, 2010).

Uma vez mapeadas as amostras do treinamento o aprendizado da análise sintática ocorre pelo método de gradiente estocástico, com o objetivo de maximizar a *cross-entropy* das operações douradas.

O processo de treinamento do NNParser é o mesmo descrito anteriormente, seja no modo monolíngue, seja no modo multilíngue. Uma vez que as diferenças entre eles está na forma como as representações distribuídas são induzidas, os processos de análise são feitos das seguintes formas: combinação de representações multilíngues (FARUQUI; DYER, 2014) através de similaridade linguísticas e concatenação de córpus universais.

O processo de representações multilíngue parte do princípio de que línguas irmãs compartilham similaridades. Através dessa similaridade, Tsarfaty et al. (2013) propõem um método de combinação de representações distribuídas através de similaridades morfossintáticas (dicionários bilíngues). O princípio utilizado é: seja y a tradução da palavra x, muito possivelmente y está no mesmo contexto de x pois palavras em diferentes línguas que são tradução uma da outra compartilham um contexto similar.

A indução de dicionários bilíngues foi feita com base no alinhamento de córpus comparáveis. Uma vez que exista uma relação entre as línguas combinam-se estas por meio da técnica de correlações canônicas (FARUQUI; DYER, 2014). Esta técnica combina vetores multilíngues a fim de obter-se uma nova representação, onde ambas as palavras

correlatas transportam suas características para esta nova representação.

Após a obtenção das "novas" representações, o NNParser extrai amostras de treinamento para cada sentença do córpus da língua original e o teste é realizado na língua alvo (projetada) gerando, assim, um modelo capaz de analisar sintaticamente duas línguas, onde as amostras de treinamento são extraídas a partir de uma das línguas combinadas.

A Figura 42 representa o processo de análise sintática multilíngue, onde as representações distribuídas são induzidas através do método de correlações canônicas. Já a Figura 38 representa o processo de análise multilíngue através da concatenação de córpus.

Figura 37 – Modelo utilizando o CCA.

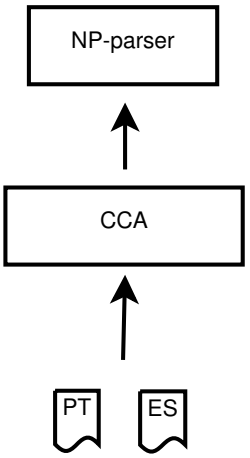
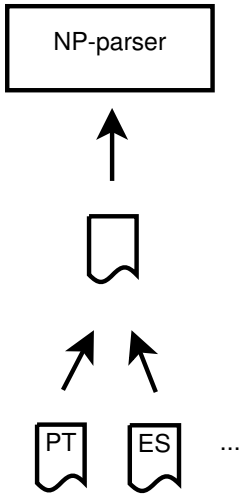


Figura 38 – Modelo de concatenção de córpus.



Fonte: próprio autor

Apesar do aumento de desempenho em diversas tarefas do PLN (FARUQUI; DYER, 2014; GUO et al., 2015), a indução automática de dicionários é uma tarefa custosa, que demanda a construção e pré-processamento de dados. Uma solução interessante e lógica é aplicar a concatenação de córpus homogêneos. A homogeneidade desses recursos permite a fácil combinação deles em um modelo de análise sintática, isto é, sem a necessidade de mapeamentos. Desse modo, a construção de um modelo universal se torna viável. Então, por meio desta proposta de concatenação de recursos torna-se possível que várias línguas sejam treinadas conjuntamente e o teste seja realizado em qualquer língua desse conjunto, tendo assim um analisador sintático multilíngue.

Mais especificadamente, a proposta da Figura 38 especifica que uma vez extraídas todas as representações distribuídas das línguas envolvidas no treinamento, o processo de *oracle* fará o mapeamento de todas as operações douradas com seus respectivos estados de análise sintática para cada língua. Ao final existirá um conjunto de amostras de todas as línguas. Assim, permite-se ao modelo um maior "refinamento" de suas crenças, isto é, uma maior quantidade de amostras para cada estado de análise sintática.

Os modelos para análise sintática mono e multilíngue treinados para o NNParser foram testados quanto à taxa de acertos (veja seção 5.2), sendo seus resultados expostos e discutidos no Capítulo 5.

5 Experimentos

O presente capítulo explora e analisa diferentes configurações para o sistema de análise sintática universal NNParser. As próximas seções descrevem os recursos (seção 5.1) e as formas de avaliação (seção 5.2) utilizados neste trabalho, seguidas pelos experimentos realizados e seus respectivos resultados em relação ao *baseline* (seção 5.3) e as versões monolíngue (seção 5.4) e multilíngue (seção 5.5) do NNParser. As diferentes línguas selecionadas para os experimentos foram português, inglês e espanhol.

Os experimentos descritos na seção 5.3 têm como objetivo demonstrar quais são os melhores parâmetros para o modelo apresentado e discutido nas seções 4.1 e 4.2. Uma vez encontrada a melhor configuração de parâmetros, as seções 5.4 e 5.5 avaliam o desempenho do NNParser para as tarefas de análise mono e multilíngue.

Mais especificamente, a seção 5.5 apresenta os resultados selecionados para o córpus *universal dependencies* versão 1.2. Estes resultados multilíngues são comparados com o *baseline* individual de cada língua e o estado da arte atual para a tarefa. A apresentação dos resultados está dividida em: (i) transferência de linguagem com representações distribuídas e (ii) concatenação de córpus. Para a tarefa monolíngue, foi utilizado o córpus *penn treebank* convertido para a tarefa de análise sintática de dependência. Também, fez-se um pequeno refinamento das diferentes entradas e saídas dos modelos multilíngue (seção 5.6).

5.1 Córpus utilizados

Para a construção e indução dos modelos é necessária a utilização de córpus. Para fins de explicação, a seguir os córpus são divididos de duas formas: córpus sem anotação linguística e córpus com anotação linguística.

A Tabela 11 traz um levantamento de alguns córpus com anotação sintática para análise por constituência e de dependência. Todos os córpus listados na tabela são de domínio público.

Para os experimentos descritos neste documento, utilizou-se o córpus *universal dependencies* na versão 1.2 (UD 1.2). As especificações de cada língua do UD 1.2 utilizadas nos experimentos são apresentadas na Tabela 12.

Para a geração das representações distribuídas é necessária a utilização de córpus. Os córpus utilizados, bem como suas informações de tamanho, estão listados na Tabela 13. A principal diferença desta tabela com a anterior é a quantidade de texto disponível.

[1] UD 1.2

Tabela 11 – Córpus com anotação sintática.

Nome	Forma de análise	Línguas	Qtd. sentenças
universal dependencies córpus[1]	Dependência	Português, Inglês,...	103.000
Bosque	Dependência	Português	9.368
Ancora	Dependência	Espanhol	17,376
WSJ-penntreebank	Constituência	Inglês	45.000
Tycho Brahe	Constituência	Português	42.000

Fonte: próprio autor

Tabela 12 – Número de sentenças em cada língua no córpus UD 1.2.

Língua	Treino	Teste
Português	9.600	1.205
Inglês	39.832	2.416
Espanhol	14.138	300

Fonte: próprio autor

Tabela 13 – Córpus sem anotação sintática utilizados neste trabalho.

Nome	Línguas	Tamanho em Gigabytes
Wikipédia português[2]	Português	1,5
Wikipédia espanhol[3]	Espanhol	3,0
Wikipédia inglês[4]	Inglês	13,0

Fonte: próprio autor

A utilização de córpus não rotulados provenientes da Wikipédia é algo normal, principalmente na indução de representações distribuídas (MIKOLOV et al., 2013; PENNINGTON; SOCHER; MANNING, 2014b).

5.2 Métricas

Uma forma de medir a quantidade de acertos em analisadores sintáticos de dependência é através de medidas como *Labeled Attachment Score* (LAS) e *Unlabeled Attachment Score* (UAS).

[2] http://dumps.wikimedia.org/ptwiki/latest/
[3] http://dumps.wikimedia.org/eswiki/latest/
[4] http://dumps.wikimedia.org/enwiki/latest/

Estas medidas são estimadas através de um córpus de teste[5]:

$$UAS = \frac{Numero\,de\,cabecas\,corretas \times 100}{Soma\,de\,Arcos\,totais}$$

$$LAS = \frac{Numero\,de\,arcos\,corretos \times 100}{Soma\,de\,Arcos\,totais}$$

Como exemplo de cálculo dessas medidas considere o grafo de dependência dourado da Figura 39, à esquerda, e o respectivo grafo hipotético gerado pelo NNParser, à direita, na Figura 40.

Figura 39 – Produções dourada.

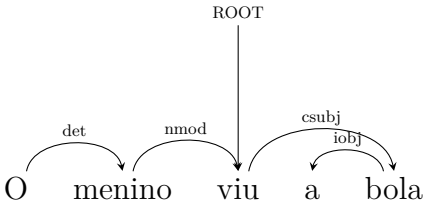
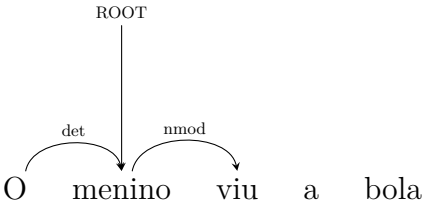


Figura 40 – Saída do NNParser.



Fonte: próprio autor

Os valores das UAS e LAS, para o exemplo descrito, são: para um número total de 5 arcos de transição e um total de 2 cabeças certas tem-se a aplicação da fórmula de $UAS = \frac{2\times100}{5} = 40$; e para um total de 2 arcos corretos e um total de 5 arcos tem-se de $LAS = \frac{2\times100}{5} = 40$.

5.3 *Baseline* para análise sintática monolíngue

Diversos experimentos foram realizados com o intuito de encontrar a melhor configuração do modelo do NNParser. As diferentes configurações testadas para a arquitetura proposta nas seções 4.1 e 4.2 estão descritas na Tabela 14.

Tabela 14 – Diferentes configurações testadas para o NNParser.

Configuração i	**Configuração ii**	**Configuração iii**
Duas camadas escondidas 200 neurônios cada	Duas camadas escondidas 400 neurônios cada	Três camadas escondidas 400 neurônios cada

Fonte: próprio autor

Estas configurações foram testadas com o córpus do inglês do *Universal Dependencies 1.2* (UD 1.2), com o objetivo inicial de encontrar a melhor configuração para a arquitetura neural.

[5] Córpus de teste é um córpus de referência, que não é utilizado no processo de treinamento do modelo. Em aprendizado de máquina geralmente divide-se um conjunto de dados em teste, treinamento e validação. O córpus de teste tem a função de verificar a capacidade de generalização de um modelo probabilístico, através de medidas claras durante o processo de teste.

O córpus do inglês do UD 1.2 é um córpus compilado de notícias provenientes do Google. Esse córpus se caracteriza por ser homogêneo em relação ao conjunto de etiquetas e por ser um *treebank*.

Como mencionado previamente, o NNParser não é capaz de tratar sentenças não projetivas, por isso, foram pré-selecionadas sentenças projetivas (em torno de 40.000 sentenças) e utilizou-se a divisão padrão original para treino e teste: 39.832 sentenças foram usadas para treino e 2.416 sentenças, para teste.

As representações distribuídas foram induzidas usando a ferramenta word2vec[6], que é a mais utilizada atualmente para a indução de representações distribuídas. Assim, utilizou-se uma inicialização randômica das representações distribuídas de etiquetas morfossintáticas/sintáticas com pesos no intervalo [-0.01, 0.01], similar a (CHEN; MANNING, 2014). Utilizou-se inicialização randômica também para "NULL" e "UNKNOWN". "NULL" é usado para representar símbolos não existentes no estado de análise sintática, e "UNKNOWN" para palavras não existentes no vocabulário das representações distribuídas previamente induzidas.

As configurações (i, ii, iii) da Tabela 14 foram testadas com diferentes parâmetros, descritos na Tabela 15.

Tabela 15 – Análise de otimizadores de método para *arc-base* para a tarefa de análise sintática do inglês no córpus PennTreebank.

Otimizador	**Fator de aprendizado**	**% de UAS**
Adagrad	$1 \times \epsilon^{-9}$	**55,35**
SGD	$0,01$	$54,35$
SGD	$1 \times \epsilon^{-9}$	$53,28$

Fonte: próprio autor

A escolha por um otimizador é crucial no processo de aprendizado automático. Uma escolha correta reduz o tempo de treinamento, levando à convergência rápida do algoritmo. Inicialmente, optou-se pelo otimizador `adagrad` e, seguindo (CHEN; MANNING, 2014), também resolveu-se comparar com o desempenho do tradicional otimizador *Stochastic Descendent Gradient* (SGD) proposto por Rumelhart David E.; Hinton (1986).

Para a análise da Tabela 15 testou-se uma forma diferente de aplicar operações. Ao invés do tradicional *arc-standard*. Optou-se em realizar operações de arco que retirem elementos do *buffer* ao invés de retirar elementos da pilha nomeado de *arc-base*, como um *baseline* para comparação de desempenhos.

[6] <https://code.google.com/archive/p/word2vec/>

Os parâmetros para os diferentes otimizadores testados na Tabela 15 foram iterados por 20.000 épocas, com um tamanho de *mini-batch* igual a 10.000 já o processo de *oracle* estático gerou em torno de 350.000 amostras de treinamento.

Ao final desse processo, concluiu-se que o melhor otimizador é o `adagrad` e a função de ativação *relu*[7]. Vale ressaltar que utilizou-se o tamanho de 50 para as representações distribuídas, similar ao proposto em (CHEN; MANNING, 2014).

A Tabela 16 traz os resultados para a forma *arc-standard* com o otimizador `adagrad` para as diferentes configurações explicadas na Tabela 14. Com base nos resultados dos experimentos constata-se a importância do aumento do número de camadas no desempenho do modelo, bem como do uso de um *oracle* guloso como o *arc-standard*.

Tabela 16 – Análise das diferentes arquiteturas propostas, aplicadas na forma *arc-base*.

Configuração	**pré-treinadas?**	% **de LAS/UAS**
i	Não	82, 18/84, 78
i	Sim	83, 85/86, 26
ii	Sim	85, 05/87, 30
iii[8]	Sim	**85,76/87,94**

Fonte: próprio autor

Em números tem-se um ganho de 32, 59 pontos percentuais (UAS) do modelo *arc-standard* em relação ao *arc-base*.

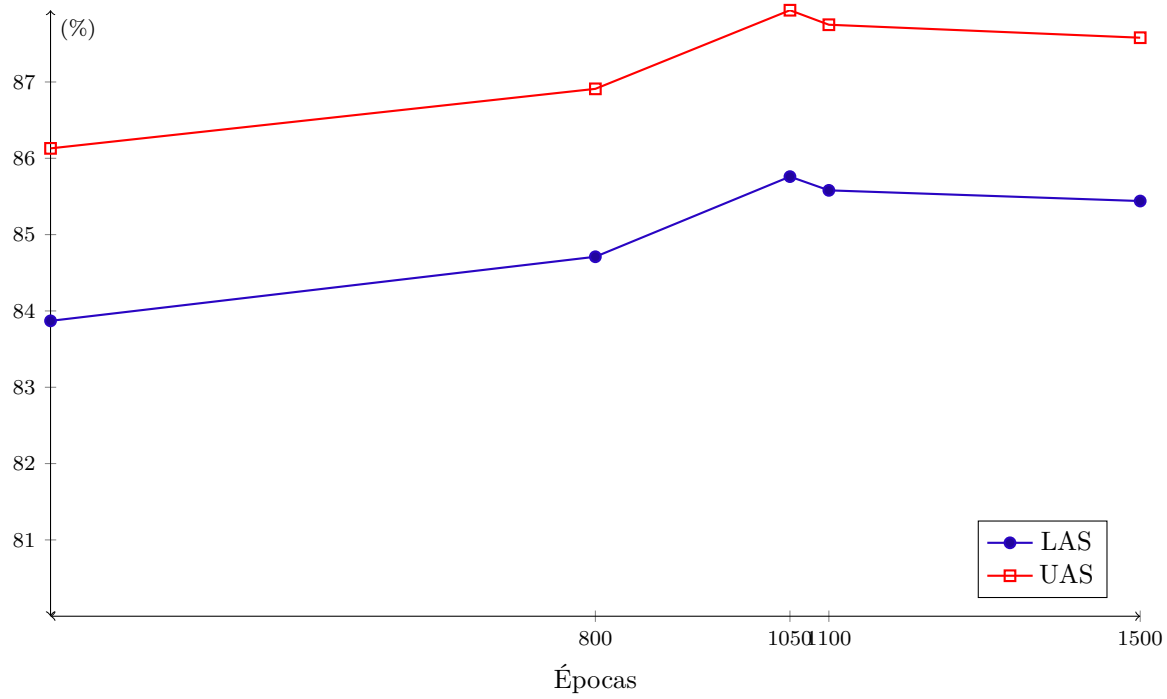
Por meio da análise dos dados apresentados nas Tabelas 16 e 15, observa-se que a melhor configuração para o NNParser é: três camadas escondidas com 400 neurônios cada, com o otimizador `adagrad` e representações de palavras distribuídas pré-treinadas com tamanho 50. Também nota-se, pela análise do gráfico da Figura 41, que o melhor número de épocas foi igual a 1.050.

Vale ressaltar que foi utilizada uma penalização nos valores das matrizes durante a fase de *backpropagation*: a l_2. A penalização l_2 foi utilizada com valor igual a 1×10^{-6} para todas as três camadas escondidas, com o objetivo de reduzir o *over-fitting* (CHEN; MANNING, 2014).

[7] A função de ativação `relu` foi demonstrada por Weiss et al. (2015) como uma forma eficiente de extração de *features* para a análise sintática de dependência. A função *relu* funciona como um método de "retificar" o aprendizado, permitindo um melhor funcionamento do método de *back-propagation*.

[8] Com 4 ou mais *hidden layers* os ganhos foram tão insignificantes que não foram computados.

Figura 41 – Impacto no número de épocas nos resultados da análise sintática monolíngue para o inglês no córpus PennTreebank.



Fonte: próprio autor

5.4 Experimentos para a análise sintática monolíngue

Para a tarefa de análise sintática monolíngue, foram realizados experimentos com o córpus *penn treebank*. Este córpus caracteriza-se por ser um compêndio de notícias jornalísticas do *Wall Streeet Jornal* (WSJ). Para treino e teste da configuração definida na seção 5.3 utilizou-se as tradicionais divisões do córpus: seções 02-21 para treino e seção 23 para teste.

Para a utilização do NNParser foi necessária a geração de representações distribuídas das palavras usando o `word2vec` com o córpus da Wikipédia, com tamanho de dimensões em 100. Já para as etiquetas sintáticas/morfossintáticas, utilizou-se uma inicialização randômica em $[-0,01, 0,01]$ com um tamanho de 50 dimensões.

Para conversão do córpus Penn Treebank para o formato de dependências utilizou-se a ferramenta `Penn2Malt`[9], com as configurações propostas em (CHEN; MANNING, 2014). Vale ressaltar as seguintes configurações para a ferramenta: *head finder* proposto em (MAGERMAA, 1995; MAGERMAA, 1996) com o descarte dos símbolos de pontuação.

Após o treinamento e o teste do NNParser nos moldes definidos por Chen e Manning (2014), foram obtidos os resultados apresentados na Tabela 17. Analisando esses

[9] <http://stp.lingfil.uu.se/~nivre/research/Penn2Malt.html>

resultados percebe-se que o NNParser é similar ao atual estado da arte (DYER et al., 2015) para a forma de extração de operações douradas de *oracle* estático.

Tabela 17 – Resultados da avaliação do NNParser na análise sintática monolíngue do inglês no córpus Penn Treebank.

Modelo	**UAS**	**LAS**
Stack-LSTM (DYER et al., 2015)	93,04	90,87
C-M (CHEN; MANNING, 2014)	91,80	89,60
NNParser	**93,08**	90,74
LSTM+Exploration+DynamicOracle (BALLESTEROS et al., 2016)	93,56	91,42
FFnetwork+Beam search+DynamicOracle+CRF (ANDOR et al., 2016)	**94,61**	**92,79**

Fonte: próprio autor

O modelo do NNParser é uma rede multi camadas e, por esse motivo, não tem a capacidade de aprender palavras fora do vocabulário uma vez que a capacidade de aprender representações distribuídas de novas palavras está associada à capacidade de "memória", como proposto em Dyer et al. (2015) (Stack-LSTM) e Ballesteros et al. (2016) (LSTM+Exploration+DynamicOracle).

Com base nos dados apresentados, pode-se afirmar que o NNParser é um modelo similar em desempenho a (DYER et al., 2015), apesar de não empregar técnicas sofisticadas em sua construção como tratamento para palavras fora do vocabulário e refinamentos específicos durante o treinamento recorrente para redes neurais.

Pode-se afirmar, também, pela comparação com o trabalho de Chen e Manning (2014), que a opção por uma modelagem *one-hot* permite um ganho real de $1,28$ pontos percentuais para a medida UAS e de $1,14$ pontos percentuais para a medida LAS (C-M).

Optou-se por não comparar diretamente os resultados obtidos pelo NNParser com o modelo proposto por Andor et al. (2016) ou Ballesteros et al. (2016), devido a estes modelos não utilizarem operações *arc-standard*, pois todo o processo de extração de operações é diferente.

Vale destacar que o resultado do modelo do NNParser para a análise monolíngue é muito bom considerando-se que ele utiliza técnicas muito simples. Então, as futuras versões do modelo devem levar em consideração a incorporação de novas características tais como treinamento dinâmico, decisões mais prolongadas, combinação de métodos de busca a fim de maximizar as possibilidade de operações de análise sintática e o tratamento de sentenças projetivas.

A incorporação de algumas dessas características demanda a modificação do processo de treinamento, uma vez que o modelo de *oracle* estático utiliza um mecanismo determinístico de treinamento. Este processo pode ser convertido em um não-determinístico pela simples adição de decisões "extras" durante o processo de treinamento permitindo,

assim, que o modelo ganhe "mais" amostras para as sentenças durante o processo de treinamento.

A combinação de um *oracle* dinâmico e métodos de busca foi proposta por Zhang e Nivre (2011a), onde a simples adição dessa técnica permite ao decisor expandir o espaço de busca, habilitando-o a uma escolha melhor e mais completa de operações douradas.

A técnica apresentada em (MCDONALD et al., 2005) prevê o uso de um algoritmo de "árvore geradora mínima" onde um grafo é aglutinado em grafos menores, não influenciando na capacidade de representação do grafo (sem inter-dependências).

Ainda sobre o processo de treinamento, uma forma menos gulosa de operação de estados de análise sintática foi proposta em (ZHANG; NIVRE, 2011b): a *arc-eager*. De acordo com tal proposta, a adição de uma operação específica, a operação *reduce*, permitiu ao modelo explorar dependências longas. Isto é, o modelo se tornou capaz de prolongar operações de arco de dependência para explorar dependências mais internas, que provavelmente não seriam encontradas em um modelo clássico de operações gulosas como o *arc-standard*.

5.5 Experimentos para a análise sintática multilíngue

Para os experimentos com análise sintática multilíngue, utilizou-se como *baseline* os resultados individuais em cada língua, que foram obtidos através do treinamento do NNParser no córpus *universal dependencies* para as línguas: inglês, espanhol e português. Como configuração para os experimentos mono e multilíngues utilizou-se a proposta (iii) descrita na seção 5.3.

Optou-se por dividir os experimentos da seguinte forma: experimentos de *baseline* (resultados apresentados na Tabela 18); experimentos utilizando transferência de linguagem através de correlações canônicas (resultados apresentados na primeira metade da Tabela 19); experimentos de concatenação de línguas (resultados apresentados na segunda metade da Tabela 19).

Para indução das representações distribuídas utilizadas em todos os experimentos, usou-se o córpus da Wikipédia com 100[10] dimensões para representações distribuídas das palavras. Já para etiquetas sintáticas e morfossintáticas, utilizou-se uma inicialização de 50 dimensões com distribuições randômicas entre o intervalo $[-0,01, 0,01]$.

A Tabela 18 apresenta os *baselines* individuais para cada língua do córpus *universal dependencies*, em comparação ao atual estado da arte para análise sintática multilíngue: o Stack LSTM (AMMAR et al., 2016a). Os melhores resultados para cada língua aparecem

[10] Para os experimentos de transferência de linguagem através de correlações canônicas, após a etapa de pré-processamento, o NNParser é treinado com o tamanho de dimensão igual à 50 para palavras.

em negrito na tabela. Analisando em detalhe os experimentos monolíngue expostos na Tabela 18, comparando-se o modelo NNParser com o modelo considerado o estado da arte, Stack LSTM, quando ambos são treinados de forma monolíngue, obtêm-se os seguintes resultados para medidas UAS/LAS: 87,94/85,76 X 88,70/85,90 para o inglês, 89,35/87,01 X 87,50/83,70 para o espanhol e 89,52/87,95 X 89,10/85,70 para o português. Percebe-se que o modelo NNParser obtém melhores resultados para as línguas de português e espanhol.

Tabela 18 – *Baselines* monolíngues utilizando o córpus UD 1.2.

Modelos	Línguas	**UAS**	**LAS**
NNParser	Inglês	87,94	85,76
	Espanhol	**89,35**	**87,01**
	Português	**89,52**	**87,95**
Média	-	88,93	86,90
Stack LSTM	Inglês	**88,70**	**85,90**
	Espanhol	87,50	83,70
	Português	89,10	85,70
Média	-	88,43	85,10

Fonte: próprio autor

O melhor desempenho para as línguas portuguesa e espanhola dos experimentos na Tabela 18, deve-se a utilização de um conjunto especifico de atributos pelo modelo "NNParser". Em hipótese, as línguas ricas morfologicamente (portuguesa e espanhola) possuem relações linguísticas mais complexas, que línguas pobres morfologicamente o que leva a necessidade de atributos mais "específicos".

Diferentemente do modelo "NNParser", o modelo "Stack LSTM" utiliza atributos genéricos de pilha e *buffer*, o que pode ter levado o modelo obter baixos desempenhos para as línguas ricas morfologicamente.

Para os experimentos de análise sintática multilíngue, duas estratégias foram investigadas. Uma delas é a estratégia de transferência de linguagem, também conhecida como projeção multilíngue. A utilização de projeções multilíngues é debatida em (GUO et al., 2015; FARUQUI; DYER, 2014). A técnica de projeção multilíngue utilizada neste trabalho é a CCA, que projeta vetores utilizando dicionários bilíngues. Assim, para os experimentos de transferência de linguagem usando correlações canônicas foram utilizados dicionários bilíngues como proposto em (GUO et al., 2015; FARUQUI; DYER, 2014). Estes dicionários foram induzidos por meio da ferramenta `GIZA++` (OCH; NEY, 2003), que realiza o alinhamento entre palavras de duas línguas utilizando um córpus paralelo.

Uma vez descobertas as similaridades das línguas, projetam-se os vetores em um espaço comum. Este novo espaço contém 50% das características de cada língua, isto é,

o espaço final resultará em um vetor com dimensão 50. A partir deste algoritmo, foram obtidas as representações: (i) português e inglês e (ii) português e espanhol.

Mais especificamente, o pré-processamento do modelo é feito através da combinação de representações multilíngues (CCA). A representação distribuída das palavras de duas línguas (linha e coluna na primeira metade da Tabela 19), por exemplo, o português (PT) e o espanhol (ES), gera uma nova representação distribuída para as palavras de ambas as línguas. Desse modo, o processo de treinamento e teste é feito de forma monolíngue na língua da coluna com as projeções da etapa do pré-processamento.

A primeira metade da Tabela 19 apresenta os resultados para as diferentes combinações de línguas durante as etapas de pré-processamento/treino e testes. Utilizou-se para os testes o córpus UD 1.2. A interpretação da primeira metade da Tabela 19 é da seguinte forma: A (linha) combina com B (coluna) e realiza treino e teste em B. Por exemplo, português (PT) combina com espanhol (ES) e realiza treino e teste em espanhol obtendo 86, 14% de LAS como medida.

Tabela 19 – Valores de LAS na análise sintática multilíngue utilizando o córpus UD 1.2.

Línguas	PT	EN	ES	Média
PT	87,95	83,28	86,14	-
EN	86,85	85,76	-	-
ES	86,99	-	87,01	-
PT-ES	**87,93**	-	**86,96**	87,47
PT-EN	87,06	-	-	87,06
ES-EN	-	-	85,95	85,95
ES-EN-PT	87,19	80,72	86,70	84,87
Stack LSTM	86,2	**85,4**	84,3	85,3

Fonte: próprio autor

Como pode-se notar pelos valores da Tabela 19, nenhuma combinação de línguas usando a estratégia de transferência de linguagem apresentou melhores resultados do que o *baseline* (Tabela 18). Esse fato pode ser observado em todos os experimentos comparando-se o melhor resultado de LAS usando transferência de linguagem com o *baseline*, respectivamente: 86,99 × 87,95 para o português, 86,14 × 87,01 para o espanhol e 83,28 × 85,76 para o inglês. O que se pode afirmar a partir desses resultados é que para obtermos bons resultados (e talvez alcançar o *baseline*) é necessário refinar melhor as etiquetas e construir um modelo mais poderoso que realize análise sintática e morfossintática em paralelo.

Também é interessante ressaltar que, nos experimentos com correlações canônicas, a ordem em que as línguas são combinadas parece afetar o resultado. Por exemplo, de acordo com os experimentos da primeira linha da Tabela 19 (PT/ES com testes em espanhol) o valor de LAS foi de 86, 14, enquanto no primeiro teste da terceira linha

(ES/PT com teste em português) o valor de LAS foi de 86, 99. Essa diferença de 0, 85 pontos percentuais pode indicar que existe uma relação entre o desempenho dos métodos de correlação canônica e a ordem na qual as línguas são combinadas.

Para a técnica de concatenação de córpus (segunda metade da Tabela 19), resolveu-se compará-la aos *baselines* individuais de cada língua (Tabela 18) e ao atual estado da arte para análise sintática multilíngue (Stack LSTM (AMMAR et al., 2016a)). Vale mencionar que apenas os valores da medida LAS são apesentados na Tabela 19 porque os modelos que foram utilizados na comparação (AMMAR et al., 2016a) apresentam valores apenas para esta medida.

Uma vez que a homogeneidade é a principal característica do córpus *unviversal dependencies*, a utilização de métodos de projeções não é necessária já que o modelo de anotação permite uma universalidade. Então, propomos a combinação de recursos multilíngues através da homogeneidade dos córpus.

Para os experimentos onde os diferentes córpus são combinados através de uma concatenação, o treino e os testes são multilíngues para línguas pertencentes ao conjunto de treinamento. Mais especificamente, a concatenação de córpus prevê a combinação de diferentes amostras de treinamento de diferentes línguas. Esta concatenação tem o objetivo de aumentar a quantidade de recursos para o treinamento do modelo, afim de cobrir uma maior quantidade de exemplos que possivelmente não seriam vistos em um treinamento monolíngue.

Nesse sentido, vale destacar, pelos valores apresentados na segunda parte da Tabela 19, que os experimentos com concatenação de córpus obtiveram resultados de LAS muito próximos aos obtidos pelo *baseline* quando línguas irmãs PT-ES foram combinadas: 87,93 para português (o *baseline* obteve 87,95) e 86,96 para o espanhol (o *baseline* obteve 87,01).

Já para os experimentos através de concatenação de línguas (segunda metade da Tabela 19) a partir dos valores médios obtidos para LAS – 87, 47 quando concatenamos português com espanhol, 87, 06 quando concatenamos português com inglês, 85, 95 quando concatenamos espanhol com inglês e 84, 87 quando concatenamos português com inglês e espanhol – é possível concluir que o melhor desempenho médio é quando línguas irmãs – português e espanhol – são combinadas. Quando analisamos os resultados do NNParser para português e espanhol (87, 47) e comparado à concatenação de português e inglês (87, 06) o modelo obtém uma diferença de 0, 41 pontos percentuais em favor da teoria de línguas irmãs.

Também vale destacar a importância da combinação por proximidades linguísticas, evidenciada no quarto experimento da segunda metade da Tabela 19. Nesse experimento, a combinação de duas línguas irmãs (PT-ES) com uma terceira não irmã (EN) acarreta

em uma perda de desempenho médio do modelo NNParser de $2,6$ pontos percentuais: de $87,47$ (PT-ES) para $84,87$ (PT-ES-EN).

Quando comparamos especificamente estes resultados com outros trabalhos tem-se que a combinação de português e espanhol, em média, para as línguas de teste tem-se $87,47\%$ de LAS, superando a combinação de línguas não irmãs proposta por Ammar et al. (2016a)[11] ($85,30\%$) em $2,17$ pontos percentuais. Esta evidência corrobora e demonstra que combinar recursos através de similaridades linguísticas é de suma importância para análise multilíngue.

Com os resultados apresentados, fica evidenciada a importância da combinação de recursos de linguagens irmãs confirmando, assim, a hipótese desse trabalho. Também atingiu-se o objetivo de obter um modelo que tenha um desempenho médio ótimo para todas as línguas utilizadas do córpus *universal dependencies*, com um desempenho médio de $86,90\%$ (LAS) para avaliações individuais (*baseline*, Tabela 18). Se este resultado for comparado com o modelo de Ammar et al. (2016a), para o mesmo ambiente de treino e teste, obtém-se um ganho médio de $1,8$ pontos percentuais para uma média de $85,1\%$ (LAS).

Em um âmbito geral para os resultados multilíngues, o NNParser apresenta resultados superiores em média ao modelo comparado (Stack LSTM). Afirma-se também, que a utilização de várias camadas escondidas combinada com uma modelagem *one-hot* permitiu explorar com mais robustez o modelo de análise de dependência. Apesar de o NNParser não atingir os *baselines* individuais, o modelo é capaz de obter resultados comparáveis ao estado da arte e foi capaz de demonstrar a importância da utilização de recursos de línguas irmãs.

Por fim, vale mencionar que o NNParser não foi capaz de superar os *baselines* individuais de cada língua para nenhuma técnica multilíngue: transferência de linguagem ou concatenação de córpus. Esse fraco desempenho deve-se, em parte, à falta de um refinamento de etiquetas morfossintáticas.

Como aponta Tiedeman (2015) e posteriormente re-afirmam Ammar et al. (2016a), a omissão de um refinamento de etiquetas morfossintáticas na construção de um analisador sintático multilíngue de dependência pode ser crucial para o bom desempenho do modelo. Esta omissão pode prejudicar o bom desempenho de um analisador sintático de dependência, como prejudicou o NNParser especialmente em experimentos multilíngues (Tabela 19).

A não indicação ou não utilização de um modelo paralelo (ou como etapa de pré-processamento) para etiquetagem morfossintática prejudica o desempenho do analisador sintático (TIEDEMAN, 2015). Isso ocorre uma vez que o não tratamento específico

[11] Estes experimentos seguem a metodologia proposta por Ammar et al. (2016a) quando é testado o ambientes onde "existem árvores na língua alvo". Eles denominam ambiente $L^o = L^t$.

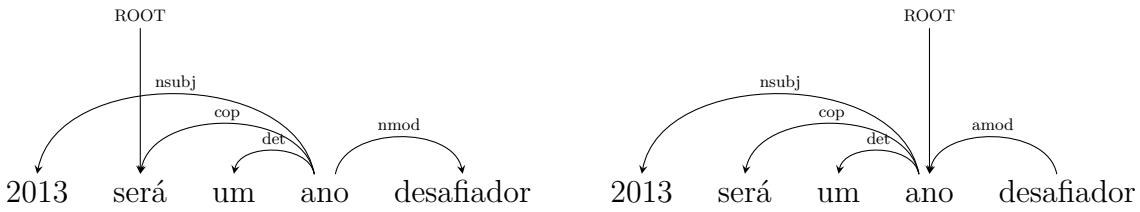
para etiquetas morfossintáticas acarreta em uma total generalização no nível morfológico, forçando o modelo a basear decisões de análise em etiquetas morfossintáticas douradas, isto é, etiquetas que somente foram vistas durante o processo de treinamento.

5.6 Análise das saídas do NNParser multilíngue

Para ilustrar o processo de análise sintática multilíngue do NNParser, a seguir apresenta-se um detalhamento desse processo através da análise das saídas produzidas para a sentença de teste: "2013 será um ano desafiador.". Mais especificamente, a seguir são apresentadas as saídas produzidas pelo NNParser, treinado/testado como descrito na seção 5.5, para os experimentos de transferência de linguagem através de correlações canônicas e para os experimentos de concatenação de linguagem.

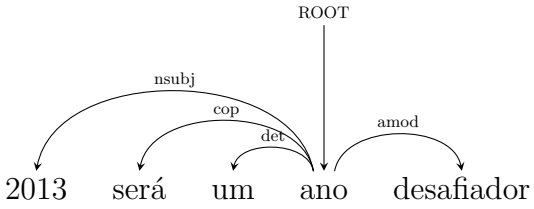
Para a técnica de transferência de linguagem através de correlações canônicas, apresenta-se a saída do experimento ES/PT para um treinamento em português e teste em português (i). Já para o experimento de concatenação de linguagens, ilustra-se a saída do experimento PT-ES com o teste em português (ii). As estruturas de dependência geradas pelo NNParser seguindo essas duas abordagens são apresentadas na Figura 42: à esquerda (i) e à direita (ii). A estrutura correta (sentença dourada) é mostrada na Figura 43.

Figura 42 – Saída do NNParser para o experimento ES/PT (i), à esquerda, e para o experimento ES-PT (ii), à direita.



Fonte: próprio autor

Figura 43 – Amostra dourada da sentença: "2013 será um ano desafiador".



Fonte: próprio autor

Percebe-se pela comparação de ambas as saídas, que as saídas da Figura 42 obtém o mesmo desempenho de UAS (80%), e para uma comparação de 80% LAS para o experimento i, e 80% LAS para o experimento ii.

O refinamento reflete os resultados obtidos na seção 5.5, onde o experimento ii $(87,93)$ é superior ao experimento i $(86,99)$ para o teste do português.

A Tabela 20 apresenta as principais etiquetas sintáticas para as línguas inglesa, portuguesa e espanhola no córpus UD 1.2. Para o refinamento proposto anteriormente, a diferença que destacamos entre os experimentos i e ii. No contexto do experimento i, a etiqueta sintática "nmod" que fora atribuída ao invés da etiqueta "amod" (português 5900) para a relação "ano" e "desafiador", deve-se à baixa frequência de "amod" em relação a "nmod" (português 23456).

Tabela 20 – Top 10 etiquetas sintáticas e suas frequências nos córpus de treino UD 1.2.

Língua	Etiqueta sintática	Frequência
Português	nmod	23465
	case	13212
	root	9308
	punct	8222
	det	6207
	amod	5900
	conj	5033
	dobj	4098
	appos	3402
	name	2524
Inglês	root	11038
	nmod	10032
	conj	5062
	det	4985
	compound	4729
	case	4591
	dobj	4172
	amod	4047
	advmod	2486
	advcl	2164
Espanhol	nmod	28576
	det	23295
	case	14155
	root	13797
	punct	12427
	amod	12012
	dobj	11966
	conj	8944
	acl	7308
	name	5774

Fonte: próprio autor

Já para o experimento ii, o universo gerado pela concatenação PT-ES possibilitou que 17912 amostras de treinamento para a etiqueta "amod" fossem consideradas, tornando esta uma escolha mais provável do que a etiqueta "nmod", se comparado com a frequência de "amod" no experimento i[12], mas não conseguiu corrigir a direção do arco

[12] Essa é uma evidência fraca, já que a decisão probabilística de qual é a melhor etiqueta sintática

de dependência o que acarretou uma perda de acurácia de LAS.

depende de vários atributos extraídos de uma configuração de análise sintática.

6 Conclusão

Nesta dissertação, apresentamos o NNParser, um modelo de análise sintática multilíngue capaz de manter um desempenho médio ótimo em diferentes línguas. O NNParser utiliza-se dos mais avançados recursos para PLN, como aprendizado profundo e representações distribuídas (TURIAN; RATINOV; BENGIO, 2010; MIKOLOV et al., 2013; CHEN; MANNING, 2014); transferência e combinação de linguagens (GUO et al., 2015; AMMAR et al., 2016a); e análise sintática por transição (NIVRE, 2003; BALLESTEROS et al., 2016; DYER et al., 2015).

Neste trabalho demonstrou-se que o NNParser é capaz de obter resultados similares ao estado da arte para a forma *statical oracle* monolíngue (DYER et al., 2015), e para a tarefa de análise sintática multilíngue obteve resultados superiores à (AMMAR et al., 2016a) para o português e o espanhol. Demonstrou-se, também, que a combinação de recursos multilíngues provenientes de línguas irmãs é superior que a simples combinação de recursos que não possuam similaridades linguísticas (veja seção 5.5). Esta constatação está embasada no fato de que existem diferenças sintáticas/fonéticas/morfossintáticas entre línguas não irmãs e irmãs.

O NNParser apresenta bons resultados para análise mono e multilíngue, por vezes, superando o modelo do estado da arte em comparação: o Stack LSTM. Apesar de línguas nem sempre compartilharem similaridades morfossintáticas/sintáticas/fonéticas o NNParser obtém resultados similares para todo o conjunto de línguas testado, o que é um dos objetivos deste trabalho. Além disso, outros objetivos foram atingidos: demonstrar a importância em combinar recursos por similaridades linguísticas (línguas irmãs) e desenvolver um modelo capaz de igualar o estado da arte.

O NNParser está disponível para download[1] e utilização. Ele foi construído utilizando a linguagem Python versão 3.4 em conjunto com a biblioteca Keras. Gostaríamos de deixar um agradecimento especial ao professor Laurent Basacier por disponibilizar a utilização dos recursos do *Laboratoire d'Informatique de Grenoble (LIG)*, bem como a oportunidade de estágio apoiado pela FAPESP dentro do projeto AIM-WEST (FAPESP 2013/50757-0).

6.1 Trabalhos Futuros

Apesar do NNParser ter uma arquitetura simples em comparação a (AMMAR et al., 2016a; DYER et al., 2015), foram obtidos resultados muito bons em diversas tarefas.

[1] <https://bitbucket.org/pablocosta/parser/src>

Contudo, diversas melhorias ainda podem ser realizadas, como: (i) modelagem de palavras fora do vocabulário; (ii) suavização de diferenças sintáticas/morfossintáticas entre línguas não irmãs por meio da modelagem informativa da linguagem a ser analisada (NASEEM; BARZILAY; GLOBERSON, 2012), possibilitando assim a extração de atributos específicos para a linguagem; (iii) utilização de atributos léxicos específicos para cada linguagem; e (iv) um maior refinamento de etiquetas morfossintáticas (AMMAR et al., 2016a; TIEDEMAN, 2015).

Para a modelagem de (i), analisando todas as propostas do estado da arte atual, estas são construídas a partir de modelos de redes neurais recorrentes (DYER et al., 2015; AMMAR et al., 2016a; BALLESTEROS et al., 2016). Modelos recorrentes são uma opção, pois eles têm á capacidade de induzir dados condicionados através da análise temporal da entrada.

Para os trabalhos futuros (i e ii) em redes neurais será necessária a capacidade de análise temporal que somente redes recursivas são capazes de mapear, podendo assim, induzir características específicas por língua ou para uma palavra específica sem a necessidade de utilizar o símbolo (UNKNOWN) para o tratamento de palavras fora do vocabulário.

Para a proposta (iii) de trabalho futuro podemos utilizar redes recursivas como foi proposto por Ammar et al. (2016a), ou talvez a forma mais simples de prover essa mudança seja utilizando uma nova janela de *embedding* para atributos léxicos. Atributos léxicos específicos por língua já estão presentes no córpus *universal depdendencies* (NIVRE et al., 2016b), podendo ser: nominais, verbais, etc.

Tiedeman (2015) demonstra que a não utilização de refinamento de etiquetas morfossintáticas, promove o decaimento do desempenho para modelos de análise sintática. Uma forma de promover o refinamento de etiquetas morfossintáticas (proposta de trabalho futuro iv) é através da utilização de um etiquetador morfológico que seja treinado em conjunto com o modelo de análise sintática, sendo assim, capaz de promover uma análise morfossintática "clara" de elementos presentes no estado de análise sintática (pilha e *buffer*).

Predições estruturadas são uma forma eficiente de realizar análise de dependência (ZHANG; NIVRE, 2011a; COLLINS, 2000). Recentemente, Weiss et al. (2015) propuseram a utilização desta técnica para análise neural de dependência. Mais especificamente, Weiss et al. (2015) demonstram que é possível realizar decisões de análise sintática através da concatenação de diferentes analisadores sintáticos onde um realiza decisões de análise sintática, e outro combina as diferentes soluções produzidas e as explora a fim de expandir o estado de busca, expandindo decisões ótimas através do tamanho da busca.

Referências

AMMAR, W. et al. Many languages, one parser. In: *TACL*. [S.l.: s.n.], 2016. Citado 12 vezes nas páginas 23, 66, 67, 68, 71, 72, 76, 88, 91, 92, 97 e 98.

AMMAR, W. et al. Massively multilingual word embeddings. *CoRR*, abs/1602.01925, 2016. Disponível em: <http://arxiv.org/abs/1602.01925>. Citado na página 23.

ANDOR, D. et al. Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042, 2016. Disponível em: <http://arxiv.org/abs/1603.06042>. Citado 3 vezes nas páginas 67, 69 e 87.

ANDRE. Redes neurais artificiais. In: *<http://www.icmc.usp.br/~andre/research/neural/>*. [S.l.: s.n.], 2014. Citado na página 41.

BALLESTEROS, M. et al. Training with exploration improves a greedy stack-lstm parser. In: *acl*. [S.l.: s.n.], 2016. Citado 6 vezes nas páginas 66, 67, 76, 87, 97 e 98.

BALLESTEROS, M.; NIVRE, J. Going to the roots of dependency parsing. *Computational Linguistics*, MIT Press, v. 39, n. 1, p. 5–13, 2013. Citado na página 67.

BECK, D. E. *Tradução automática estatística baseada em sintaxe e linguagens de árvores*. Dissertação (Mestrado) — UNIVERSIDADE FEDERAL DE SAO CARLOS, CENTRO DE CIENCIAS EXATAS E DE TECNOLOGIA, PROGRAMA DE POS-GRADUACÃO EM CIÊNCIA DA COMPUTAÇÃO, São Carlos, 2012. Citado 3 vezes nas páginas 25, 26 e 32.

BENGIO, Y. et al. A neural probabilistic language model. In: *Journal of Machine Learning Research*. [S.l.: s.n.], 2003. Citado na página 50.

BLUNSOM, P.; COHN, T. Unsupervised induction of tree substitution grammars for dependency parsing. In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. [S.l.: s.n.], 2010. Citado na página 35.

BOHNET, B. et al. Joint morphological and syntactic analysis for richly inflected languages. In: *ACL*. [S.l.: s.n.], 2013. Citado 3 vezes nas páginas 7, 9 e 36.

BOHNET, T. et al. Joint morphological and syntactic analysis for richly inflected languages. In: *transacl*. [S.l.: s.n.], 2013. Citado na página 40.

BROWN, P. et al. *Method and system for natural language translation*. Google Patents, 1995. US Patent 5,477,451. Disponível em: <https://www.google.com/patents/US5477451>. Citado 2 vezes nas páginas 7 e 9.

BROWN, P. F.; SOUZA, P. V. de; MERCER, R. L. Class-based n-gram models of natural language. In: *ACL*. [S.l.: s.n.], 1992. Citado 2 vezes nas páginas 49 e 50.

CHARNIAK, E.; JOHNSON, M. Coarse-to-fine n-best parsing and maxent discriminative reranking. In: *Proceedings of the 43rd Annual Meeting of the ACL*. [S.l.: s.n.], 2005. Citado 2 vezes nas páginas 35 e 36.

CHEN, D.; MANNING, C. D. A fast and accurate dependency parser using neural networks. In: *emnlp*. [S.l.: s.n.], 2014. Citado 26 vezes nas páginas 7, 9, 12, 20, 47, 49, 56, 57, 59, 60, 61, 62, 63, 65, 66, 67, 71, 72, 73, 74, 75, 84, 85, 86, 87 e 97.

COHEN, S. B.; DAS, D.; SMITH, N. A. Unsupervised structure prediction with non-parallel multilingual guidance. In: *EMNLP*. [S.l.: s.n.], 2011. Citado 2 vezes nas páginas 7 e 9.

COLLINS, M. Discriminative reranking for natural anguage parsing. In: *In Machine Learning: Proceedings f the Seventeenth International Conference ICML*. [S.l.: s.n.], 2000. Citado na página 98.

COLLINS, M. Head-driven statistical models for natural language parsing. *Computational linguistics*, MIT Press, v. 29, n. 4, p. 589–637, 2003. Citado na página 17.

COLLOBERT, R. Deep learning for efficient discriminative parsing. In: *AISTATS*. [S.l.: s.n.], 2011. Citado 3 vezes nas páginas 20, 56 e 77.

COLLOBERT, R. et al. Natural language processing (almost) from scratch. In: *Journal of Machine Learning Research*. [S.l.: s.n.], 2011. Citado 6 vezes nas páginas 51, 52, 53, 54, 62 e 73.

COPPOLA, C. A. D. W. G.; PETROV, S. Improved transition-based parsing and tagging with neural networks. 2015. Citado 2 vezes nas páginas 67 e 69.

DYER, C. et al. Transition-based dependeny parsing with stack long short-term memory. In: *Proc. ACL*. [S.l.: s.n.], 2015. Citado 11 vezes nas páginas 20, 65, 66, 71, 72, 74, 75, 76, 87, 97 e 98.

FARUQUI, M.; DYER, C. Improving vector space word representations using multilingual correlation. In: *Proceedings of EACL*. [S.l.: s.n.], 2014. Citado 6 vezes nas páginas 23, 67, 71, 77, 78 e 89.

FONSECA, E. R.; ALUÍSIO, S. M. A deep architecture for non-projective dependency parsing. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS-ACL. *Conference of the North American Chapter of the Association for Computational Linguistics-Human Language Technologies; Workshop on Vector Space Modeling for Natural Language Processing, I*. [S.l.], 2015. Citado na página 68.

GILLENWATER, J. et al. Sparsity in dependency grammar induction. In: *acl10*. [S.l.: s.n.], 2010. Citado na página 35.

GOLDBERG, Y.; ELHADAD, M. Word segmentation, unknown-word resolution, and morphological agreement in a hebrew parsing system. In: *mitpressjournals*. [S.l.: s.n.], 2013. Citado na página 39.

GOLDBERG, Y.; NIVRE, J. A dynamic oracle for arc-eager dependency parsing. In: . [S.l.: s.n.], 2012. Citado na página 67.

GUO, J. et al. Cross-lingual dependency parsing based on distributed representations. In: *ACL*. [S.l.: s.n.], 2015. Citado 7 vezes nas páginas 22, 23, 67, 76, 78, 89 e 97.

HAJIC, J. Morphological tagging: Data vs. dictionaries. In: *acl*. [S.l.: s.n.], 2000. Citado na página 39.

HALL, J. N. andJohan; NILSSON, J. Maltparser: A data-driven parser-generator for dependency parsing. In: *LREC*. [S.l.: s.n.], 2006. Citado 5 vezes nas páginas 18, 47, 59, 65 e 75.

HINTOM. Types of neural network architectures. In: *<https://www.youtube.com/watch?v=w7CZqul0CcM&list=PLnnr1O8OWc6ZL5a3d8KaaqGHIMQagJKyg>*. [S.l.: s.n.], 2014. Citado na página 44.

IEL-UNICAMP; IME-USP. Córpus histórico do português anotado tycho brahe. url: http://www.tycho.iel.unicamp.br/. In: *IEL-UNICAMP e IME-USP*. [S.l.: s.n.], 2010. Citado na página 27.

JOULIN, A. et al. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759, 2016. Disponível em: <http://arxiv.org/abs/1607.01759>. Citado na página 56.

JURGENS. Measuring relation similarity. In: *SemEval*. [S.l.: s.n.], 2012. Citado na página 54.

KARLEN, M. et al. Large scale manifold transduction. In: *In International Conference on Machine Learning, ICML*. [S.l.: s.n.], 2008. Citado na página 50.

KLEIN, D.; MANNING, C. D. Accurate unlexicalized parsing. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. [S.l.], 2003. p. 423–430. Citado na página 17.

KLEIN, D.; MANNING, C. D. Corpus-based induction of syntactic structure: Models of dependency and constituency. In: *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. [S.l.: s.n.], 2004. Citado na página 35.

KOO, T.; CARRERAS, X.; COLLINS, M. Simple semi-supervised dependency parsing. In: *acl*. [S.l.: s.n.], 2008. Citado 2 vezes nas páginas 39 e 49.

KOVACS, Z. *Redes Neurais Artificiais*. LIVRARIA DA FISICA, 2002. ISBN 9788588325142. Disponível em: <https://books.google.com.br/books?id=O0nLxR67wmUC>. Citado 2 vezes nas páginas 42 e 43.

LEE, J.; NARADOWSKY, J.; SMITH, D. A. A discriminative model for joint morphological disambiguation and dependency parsing. In: *ACL*. [S.l.: s.n.], 2011. Citado na página 40.

LING, W. et al. Two/too simple adaptations of word2vec for syntax problems. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. [S.l.]: Association for Computational Linguistics, 2015. Citado na página 55.

MAGERMAA, D. Statistical decision-tree models for parsing. In: *the 33rd Annual Meeting of the Association for Computational Linguistics*. [S.l.: s.n.], 1995. Citado na página 86.

MAGERMAA, D. A new statistical parser based on bigram lexical dependencies. In: *the 34th Annual Meeting of the Association for Computational Linguistics*. [S.l.: s.n.], 1996. Citado na página 86.

MANNING, C. D.; SCHÜTZE, H. *Foundations of Statistical Natural Language Processing*. 1th. ed. [S.l.]: The MIT Press, 1999. Citado 6 vezes nas páginas 26, 27, 28, 30, 31 e 33.

MARCUS MITCHELL P., B. S.; MARCINKIEWICZ, M. A. Building a large annotated corpus of english: the penn treebank. In: *Association for Computational Linguistics 19.2, pp. 313–330*. [S.l.: s.n.], 1994. Citado na página 27.

MCCLOSKY, D.; CHARNIAK, E.; JOHNSON, M. Effective self-training for parsing. In: *Proceedings of HLT/NAACL*. [S.l.: s.n.], 2006. Citado 2 vezes nas páginas 35 e 36.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. In: *Department of Psychiatry at the Illinois Neuropsychiatric Institute*. [S.l.: s.n.], 1943. Citado 2 vezes nas páginas 41 e 43.

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 2 vezes nas páginas 67 e 88.

MCDONALD, R.; PETROV, S.; HALL, K. Multi-source transfer of delexicalized dependency parsers. In: *NACL*. [S.l.: s.n.], 2011. Citado 9 vezes nas páginas 7, 9, 37, 38, 39, 40, 67, 72 e 76.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. In: *ICLR*. [S.l.: s.n.], 2013. Citado 12 vezes nas páginas 20, 21, 45, 53, 54, 55, 56, 62, 71, 73, 82 e 97.

MIKOLOV, T. et al. Extensions of recurrent neural network language model. In: *Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2011. Citado 3 vezes nas páginas 7, 9 e 74.

MIKOLOV, T.; LE, Q. V.; SUTSKEVER, I. Exploiting similarities among languages for machine translation. In: *arXiv*. [S.l.: s.n.], 2013. Citado 2 vezes nas páginas 20 e 49.

MIKOLOV, T.; YIH, W. tau; ZWEIG, G. Linguistic regularities in continuous space word representations. In: *NAACL*. [S.l.: s.n.], 2013. Citado na página 59.

NASEEM, T.; BARZILAY, R.; GLOBERSON, A. Selective sharing for multilingual dependency parsing. In: *ACL*. [S.l.: s.n.], 2012. Citado na página 98.

NG, S. D. e V. Discriminative models for semi-supervised natural langua ge learning. In: *Proceedings of the NAACL HLT Workshop on Semi-Supervised Learning for Natural Language Processing*. [S.l.: s.n.], 2009. Citado 2 vezes nas páginas 35 e 36.

NIVRE, J. An efficient algorithm for projective dependency parsing. In: *In Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*. [S.l.: s.n.], 2003. Citado 7 vezes nas páginas 18, 62, 65, 71, 72, 75 e 97.

NIVRE, J. Adventures in multilingual parsing. In: *lingfil*. [S.l.: s.n.], 2014. Citado 6 vezes nas páginas 18, 23, 31, 40, 72 e 74.

NIVRE, J. et al. *Universal Dependencies 1.3*. 2016. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague. Disponível em: <http://hdl.handle.net/11234/1-1699>. Citado 4 vezes nas páginas 7, 9, 29 e 67.

NIVRE, J. et al. *Universal Dependencies 1.3*. 2016. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague. Disponível em: <http://hdl.handle.net/11234/1-1699>. Citado 3 vezes nas páginas 19, 76 e 98.

OCH, F. J.; NEY, H. A systematic comparison of various statistical alignment models. *Computational Linguistics*, v. 29, n. 1, p. 19–51, 2003. Citado na página 89.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: *EMNLP*. [S.l.: s.n.], 2014. v. 14, p. 1532–1543. Citado 2 vezes nas páginas 20 e 55.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: *Stanford Press*. [S.l.: s.n.], 2014. Citado 2 vezes nas páginas 71 e 82.

PETROV, S.; DAS, D.; MCDONALD, R. A universal part-of-speech tagset. In: *LREC*. [S.l.: s.n.], 2012. Citado 3 vezes nas páginas 19, 23 e 67.

ROSENBLATT. Perceptron. In: *Department of Psychiatry at the Illinois Neuropsychiatric Institute*. [S.l.: s.n.], 1958. Citado 2 vezes nas páginas 41 e 43.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation. In: *Stanford*. [S.l.: s.n.], 1986. Citado 2 vezes nas páginas 43 e 44.

RUMELHART DAVID E.; HINTON, G. E. W. R. J. Learning representations by back-propagating errors - (533–536). In: *Nature*. [S.l.: s.n.], 1986. Citado na página 84.

SCHUSTER, S.; MANNING, C. D. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In: *Proceedings of the 10th International Conference on Language Resources and Evaluation*. [S.l.: s.n.], 2016. Citado na página 76.

SNYDER, B.; NASEEM, T.; BARZILAY, R. Unsupervised multilingual grammar induction. In: *ACL*. [S.l.: s.n.], 2009. Citado 2 vezes nas páginas 37 e 38.

SOCHER, R. et al. Parsing with compositional vector grammars. In: *ACL*. [S.l.: s.n.], 2013. Citado 10 vezes nas páginas 12, 17, 20, 49, 53, 56, 57, 58, 59 e 61.

SOCHER, R.; MANNING, C.; BENGIO, Y. Deep learning for nlp (without magic). In: *ACL 2012 + NAACL 2013 Tutorial*. [S.l.: s.n.], 2013. Citado 6 vezes nas páginas 44, 46, 47, 48, 49 e 55.

TACKSTROM, O.; MCDONALD, R.; USZKOREIT, J. Cross-lingual word clusters for direct transfer of linguistic structure. In: *NAACL*. [S.l.: s.n.], 2012. Citado 6 vezes nas páginas 7, 9, 37, 39, 40 e 76.

TIEDEMAN, J. Cross-lingual dependency parsing with universal dependencies and predicted pos labels. In: *Proc. of Depling*. [S.l.: s.n.], 2015. Citado 3 vezes nas páginas 68, 92 e 98.

TITOV, I.; HENDERSON, J. Fast and robust multilingual dependency parsing with a generative latent variable model. In: *conll*. [S.l.: s.n.], 2007. Citado 2 vezes nas páginas 7 e 9.

TSARFATY, R. et al. Parsing morphologically rich languages: Introduction to the special issue. In: *ACL*. [S.l.: s.n.], 2013. Citado 6 vezes nas páginas 7, 9, 18, 27, 36 e 77.

TURIAN, J.; RATINOV, L.; BENGIO, Y. Word representations: A simple and general method for semi-supervised learning. In: *ACL*. [S.l.: s.n.], 2010. Citado 3 vezes nas páginas 50, 77 e 97.

WANG; BAOBAO. Graph-based dependency parsing with bidirectional lstm. In: . [S.l.: s.n.], 2016. Citado 2 vezes nas páginas 68 e 69.

WANG; BAOBAO. Improved graph-based dependency parsing via hierarchical lstm networks. In: SPRINGER. *China National Conference on Chinese Computational Linguistics*. [S.l.], 2016. p. 25–32. Citado 2 vezes nas páginas 68 e 69.

WEISS, D. et al. Structured training for neural network transition-based parsing. 2015. Citado 5 vezes nas páginas 69, 71, 72, 85 e 98.

YAMADA, H.; MATSUMOTO, Y. Statistical dependency analysis with support vector machines. In: *IWPT*. [S.l.: s.n.], 2003. Citado 3 vezes nas páginas 18, 62 e 65.

YOUNGER, D. m. Recognition and parsing of context-free languages. In: *Seventh Annual Symposium on Circuit Theory Switching and Logical Design*. [S.l.: s.n.], 1967. Citado na página 32.

ZHANG; ZHAO; QIN. Probabilistic graph-based dependency parsing with convolutional neural network. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. [S.l.: s.n.], 2016. p. 1382–1392. Citado 2 vezes nas páginas 68 e 69.

ZHANG, Y.; NIVRE, J. ransition-based dependency parsing with rich non-local features. In: *ACL*. [S.l.: s.n.], 2011. Citado 7 vezes nas páginas 66, 72, 73, 75, 76, 88 e 98.

ZHANG, Y.; NIVRE, J. Transition-based dependency parsing with rich non-local features. In: *ACL*. [S.l.: s.n.], 2011. Citado 5 vezes nas páginas 18, 62, 65, 66 e 88.