Pontifícia Universidade Católica
do Rio de Janeiro

Miguel Mendes de Brito

Aprendizado Profundo aplicado à Segmentação de Texto

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio.

Orientador: Prof. Sérgio Colcher

Rio de Janeiro
Janeiro de 2019

Pontifícia Universidade Católica
do Rio de Janeiro

Miguel Mendes de Brito

Aprendizado Profundo aplicado à Segmentação de Texto

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Sérgio Colcher
Orientador
Departamento de Informática – PUC-Rio

Prof. Marco Serpa Molinaro
Departamento de Informática – PUC-Rio

Prof. Leandro Guimarães Marques Alvim
Departamento de Ciência da Computação – UFRRJ

Prof. Ruy Luiz Milidiú
Departamento de Informática – PUC-Rio

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Departamento de Informática – PUC-Rio

Rio de Janeiro, 21 de Janeiro de 2019

Miguel Mendes de Brito

Graduado em Ciência da Computação pela Universidade Federal Rural do Rio de Janeiro.

Ficha Catalográfica

Brito, Miguel Mendes de

Aprendizado Profundo aplicado à Segmentação de Texto / Miguel Mendes de Brito; orientador: Sérgio Colcher. – Rio de janeiro: PUC-Rio, Departamento de Informática, 2019.

v., 65 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Informática – Teses. 2. Procesamento de Linguagem Natural;. 3. Aprendizado Profundo;. 4. Aprendizado de Máquinas;. 5. Segmentação Textual.. I. Colcher, Sérgio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Eu gostaria de agradecer a Deus por ter me dado toda a estrutura que eu precisei para realizar este trabalho. Sem a saúde e a força por Ele me dada, nada disso seria possível.

Agradeço enormemente a minha família, em especial, meus pais pois se cheguei até aqui foi porque me incentivaram e forneceram todo o apoio para que eu pudesse ter uma boa educação. Também sou grato a minha irmã Lidiane, por todos esses anos de amizade. Amo vocês!

Também sou imensamente grato a minha noiva Paula. Seu amor, apoio, carinho e compreensão foram fundamentais durante essa jornada. Seu apoio se estende muito além do incentivo motivacional e dos puxões de orelha necessários, sendo ajuda fundamental na revisão gramatical deste trabalho. Serei eternamente grato, meu bem.

Agradeço ao meu orientador Ruy, por todas as discussões, ensinamentos e conselhos em diversos momentos. Agradeço também pela confiança a mim depositada.

Agradeço também aos professores Marco Molinaro, Sérgio Colcher e Leandro Alvim por aceitarem participar desta banca e contribuírem para a melhoria deste trabalho.

Eu gostaria de agradecer a todos os meus amigos da UFRRJ por todos esses anos de amizade que dura até hoje. Por último, agradeço aos meus amigos do LEARN, pelas conversas descontraídas, pelo aprendizado e as dicas valiosas de quem já passou por essa fase. Em especial, gostaria de agradecer ao Guilherme Varela (Bardo), pela ajuda prestada a mim nesse tempo e ao João Forny pelos esclarecimentos de diversas dúvidas que tive ao longo do ano.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Brito, Miguel Mendes de; Colcher, Sérgio. **Aprendizado Profundo aplicado à Segmentação de Texto**. Rio de Janeiro, 2019. 65p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O Processamento de Linguagem natural é uma área de pesquisa que explora como computadores podem entender e manipular textos em linguagem natural. Dentre as tarefas mais conhecidas em PLN está a de rotular sequências de texto. O problema de segmentação de texto em sintagmas é um dos problemas que pode ser abordado como rotulagem de sequências. Para isto, classificamos quais palavras pertencem a um sintagma, onde cada sintagma representa um grupo disjunto de palavras sintaticamente correlacionadas. Este tipo de segmentação possui importantes aplicações em tarefas mais complexas de processamento de linguagem natural, como análise de dependências, tradução automática, anotação de papéis semânticos, identificação de orações e outras. O objetivo deste trabalho é apresentar uma arquitetura de rede neural profunda para o problema de segmentação textual em sintagmas para a língua portuguesa. O corpus usado nos experimentos é o Bosque, do projeto Floresta Sintá(c)tica. Baseado em trabalhos recentes na área, nossa abordagem supera o estado-da-arte para o português ao alcançar um $F_{\beta=1}$ de 90,51, que corresponde a um aumento de 2,56 em comparação com o trabalho anterior. Além disso, como forma de comprovar a qualidade do segmentador, usamos os rótulos obtidos pelo nosso sistema como um dos atributos de entrada para a tarefa de análise de dependências. Esses atributos melhoraram a acurácia do analisador em 0,87.

Palavras-chave

Procesamento de Linguagem Natural; Aprendizado Profundo; Aprendizado de Máquinas; Segmentação Textual.

Abstract

Brito, Miguel Mendes de; Colcher, Sérgio (Advisor). **Deep Learning applied to Text Chunking**. Rio de Janeiro, 2019. 65p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Natural Language Processing is a research field that explores how computers can understand and manipulate natural language texts. Sequence tagging is amongst the most well-known tasks in NLP. Text Chunking is one of the problems that can be approached as a sequence tagging problem. Thus, we classify which words belong to a chunk, where each chunk represents a disjoint group of syntactically correlated words. This type of chunking has important applications in more complex tasks of natural language processing, such as dependency parsing, machine translation, semantic role labeling, clause identification and much more. The goal of this work is to present a deep neural network archtecture for the Portuguese text chunking problem. The corpus used in the experiments is the *Bosque*, from the *Floresta Sintá(c)tica* project. Based on recent work in the field, our approach surpass the state-of-the-art for Portuguese by achieving a $F_{\beta=1}$ of 90.51, which corresponds to an increase of 2.56 in comparison with the previous work. In addition, in order to attest the chunker effectiveness we use the tags obtained by our system as feature for the depedency parsing task. These features improved the accuracy of the parser by 0.87.

Keywords

Natural Language Processing; Deep Learning; Machine Learning; Text Chunking.

Sumário

Lista de figuras

Lista de tabelas

Lista de Abreviaturas

PLN – Processamento de Linguagem Natural
AM – Aprendizado de Máquina
NLP – *Natural Language Processing*
NP – *Noun Phrase*
VP – *Verb Phrase*
PP – *Prepositional Phrase*
ADJP – *Adjectival Phrase*
ADVP – *Adverbial Phrase*
POS – *Part-of-Speech*
CoNLL – *Conference on Computational Natural Language Learning*
TBL – *Transformation-Based Learning*
CRF – *Conditional Random Fields*
SVM – *Support Vector Machines*
RNN – *Recurrent Neural Network*

1
Introdução

1.1
Visão Geral

O surgimento da internet e, consequentemente, sua adoção massiva ao redor do mundo, trouxe consigo um aumento exponencial na produção de dados. Um estudo (11) apresentado em 2012 estima que mais de 90% cento dos dados em circulação no ano de 2011 foram produzidos nos dois anos anteriores. Segundo a Domo (12), somente a rede social *Twitter* produziu uma média de 473.400 mensagens por minuto em 2018. Mais espantosa ainda é a quantidade de mensagens de texto enviadas, com uma média de 12.986.111 por minuto.

O processamento de grandes quantidades de dados, por si só, é um grande desafio devido à necessidade de recursos computacionais adequados. Tão desafiador quanto, é a necessidade de filtrar, categorizar e extrair conhecimento a partir de dados não estruturados, como os textos produzidos em redes sociais e aplicativos de mensagem instantânea. O Processamento de Linguagem Natural (PLN) é a área de pesquisa que se preocupa, justamente, em como tornar textos em linguagem natural manipuláveis e entendíveis por computadores.

Os problemas estudados em PLN estendem-se desde os problemas linguísticos até os de aplicação mais direta, como os sistemas de perguntas e respostas e sumarização de textos. Dentro do conjunto dos problemas linguísticos, existe uma classe de tarefas que se propõe a classificar sequências de palavras[1]. Exemplos de tais tarefas são a análise morfossintática[2], reconhecimento de entidades nomeadas[3], anotação de papéis semânticos[4] e a segmentação textual em sintagmas[5]. Cada uma dessas tarefas varia em níveis de complexidade. Por exemplo, a tarefa de anotação de papéis semânticos é muito mais complexa que a análise morfossintática.

O problema de segmentação de texto consiste em particionar uma sentença em grupos de palavras sintaticamente relacionadas. Além disso, tais gru-

[1] *Sequence Tagging*
[2] *Part-of-speech Tagging*
[3] *Named Entity Recognition*
[4] *Semantic Role Labeling*
[5] *Text Chunking*

pos são disjuntos, logo, uma palavra não pode pertencer a mais de um grupo simultaneamente. É muito comum em PLN que tarefas mais complexas sejam resolvidas em etapas encadeadas. Fazem parte dessas etapas problemas menos complexos, que, por sua vez, fornecem informações linguísticas importantes. Por exemplo, o problema de segmentação textual pode se beneficiar de informação de análise morfossintática, assim como o problema de anotação de papéis semânticos pode se beneficiar de informações de segmentos de texto, que em inglês é conhecido por *text chunking*, ou apenas *chunking*.

Os primeiros sistemas de Processamento de Linguagem Natural eram baseados em regras formais, que se propunham a descrever as características linguísticas do problema. Todavia, isso nem sempre é possível, pois uma língua humana não é plenamente formal. Além disso, pessoas estão sempre "burlando" as regras para atender suas necessidades de comunicação (13). Dessa forma, faz-se necessário atacar o problema de segmentação textual de outra maneira, como por meio de abordagens centradas em dados, com o uso de técnicas de Aprendizado de Máquina.

O Aprendizado de Máquina (AM) é uma área de pesquisa que se propõe a estudar algoritmos capazes de aprender automaticamente, ou seja, sem serem explicitamente programados (14). Uma grande vantagem desses algoritmos é que seu objetivo é otimizar um critério de desempenho usando dados como exemplo (15). Em outras palavras, o aprendizado é o processo de otimização de um modelo, definido por parâmetros, a partir de dados usados para o treinamento. Como resultado, a abordagem de problemas de PLN por meio de algoritmos de AM mostrou-se um grande sucesso e vem sendo usado até hoje.

Uma outra vantagem de se aplicar algoritmos de AM para problemas linguísticos é o fato dos modelos serem agnósticos ao idioma da tarefa. Isso implica que uma técnica pode ser adaptada com poucas modificações para outros idiomas, bastando apenas um conjunto de dados anotados para a etapa de treinamento. Como forma de incentivar a avanço de pesquisas em Processamento de Linguagem Natural, em especial, por meio de Aprendizado de Máquina, em 1997 é lançada a *Conference on Computational Natural Language Learning* (CoNLL). A CoNLL é uma competição anual onde uma determinada tarefa linguística é o tema central. No ano 2000, seu tema central foi a tarefa de segmentação textual[6] para a língua inglesa. A partir dessa competição, várias abordagens por meio de Aprendizado de Máquina foram propostas para esse problema e a tarefa tornou-se um *benchmark*. Além disso, tarefas subsequentes passaram a incluir informações de *chunk* como atributos

[6] *text chunking*

em seus conjuntos de dados, evidenciando a sua importância.

Apesar da CoNLL-2000 ser a referência em pesquisas sobre segmentação de texto em inglês, a língua portuguesa, por muitos anos, careceu de estudo semelhante. A maioria dos estudos sobre *chunking* focavam em apenas uma classe de sintagma, a nominal. Em 2011, Ferreira (1) propôs uma heurística para extrair segmentos do corpus Bosque a partir de informações sintáticas completas. Seu objetivo era o de analisar o impacto que suas definições de *chunk* teriam em tarefas mais complexas, como análise de dependências[7] e identificação de orações[8]. Além disso, em seu trabalho, Ferreira propôs dois modelos de Aprendizado de Máquina para a extração automática de segmentos de textos. Até onde sabemos, este trabalho é o mais semelhante ao realizado em 2000 para o inglês.

Com o avanço das pesquisas, recentemente, a área de Aprendizado de Máquina sofreu uma grande mudança de paradigma. Até poucos anos, técnicas convencionais de AM dominavam o estado-da-arte em diversas tarefas. Apesar dos bons resultados até então, tais métodos possuem uma limitação que os impede de generalizar em problemas caracterizados por dados de alta dimensão (16). Não somente isso, esses métodos possuem grande dificuldade de descobrir estruturas latentes a partir de dados brutos. A solução para essa limitação veio por meio do que se conhece hoje como Aprendizado Profundo.

Em PLN, era muito comum que uma engenharia de atributos fosse feita de antemão para se empregar algoritmos tradicionais de AM. Tais atributos incluíam particularidades linguísticas do idioma alvo. Em 2011, Collobert et al. (17) investigou o uso de redes neurais profundas para problemas de rotulamento de sequências, entre eles a segmentação de texto. Seu objetivo era o de avaliar a efetividade dessas redes a partir de dados com pouca ou nenhuma informação linguística. Apesar de resultados promissores, o trabalho mostrou que ainda não é possível resolver tarefas linguísticas mais complexas sem nenhuma engenharia de atributos.

Mesmo as redes profundas obtendo resultados melhores que técnicas convencionais, atributos como *POS tags* e segmentação são capazes de melhorar bastante o desempenho de redes neurais. Apesar disso, seu trabalho inspirou muitos pesquisadores a usarem técnicas de Aprendizado Profundo para problemas de PLN, que, por sua vez, vieram a bater o estado-da-arte em diversos problemas.

[7] dependency parsing
[8] clause identification

1.2 Proposta

O objetivo deste trabalho é investigar técnicas de Aprendizado Profundo para o problema de segmentação de texto em português. Até onde sabemos, não existe nenhum outro trabalho que empregue tais técnicas para o português. Para isso, inspirado nos trabalhos de segmentação em inglês, propomos uma arquitetura neural e comparamos os resultados obtidos com o estado-da-arte tanto para o português quanto para o inglês. Além disso, aplicados os segmentos obtidos pela nossa rede e avaliamos seu impacto na tarefa de análise de dependências.

1.3 Contribuições

Em nossos experimentos, nossa arquitetura foi capaz de melhorar o $F_{\beta=1}$ em um dos *datasets* em até 5,96 pontos. Por último, como forma de atestar a qualidade do segmentador, usamos as anotações obtidas para enriquecer os conjuntos de dados da tarefa de análise de dependências, que foi o tema da CoNLL-X, em 2006. Em nossos experimentos, os *chunks* obtidos pelo segmentador foram capazes de melhorar o desempenho dessa tarefa em até 0,87 pontos UAS.

1.4 Organização

Este trabalho está organizado da seguinte forma: no Capítulo 2 definimos a tarefa de segmentação de texto, apresentamos sua motivação, discutimos as principais abordagens para a segmentação em inglês e, também, em português; por último, apresentamos o corpus usado e também a heurística proposta por Ferreira (1) para a extração dos *chunks*. No Capítulo 3, apresentamos os conceitos teóricos em que este trabalho se embasa. Discutimos os conceitos do Aprendizado de Máquina, do Aprendizado Profundo e também debatemos sobre redes neurais profundas. O último tópico desse capítulo discute sobre a importância da representação de textos por meio de vetores de palavras. No Capítulo 4, relatamos a metodologia aplicada para a resolução da tarefa. Nele, é dada uma visão geral sobre os conjuntos de dados empregados, discutimos a representação textual adotada e a arquitetura proposta. O próximo capítulo discute os experimentos conduzidos e os resultados obtidos, tanto para a tarefa de segmentação quanto para a análise de dependências. Por último, no Capítulo 6 concluímos a dissertação e apresentamos possíveis direções a serem seguidas para a melhoria deste trabalho.

2
Segmentação de Texto

Neste capítulo, apresentamos uma definição formal para o problema de segmentação de texto, sua motivação e os trabalhos relacionados que empregam técnicas de Aprendizado de Máquina para a segmentação automática de textos.

2.1
Definição

A tarefa de **segmentação de texto** consiste em dividir um texto em conjuntos disjuntos, e não recursivos, de palavras sintaticamente correlacionadas (18). Isso significa que uma palavra não pode pertencer a dois grupos, ou *chunks*, diferentes. Com isso, devido a esta característica não recursiva, segmentos classificados como sintagmas nominais, por exemplo, podem ser usados com propósito de geração de índices (19). A Figura 2.1 exemplifica a divisão de uma sentença em segmentos.

Esse tipo de segmentação, que também é conhecida como *parsing* superficial[1], possui importantes aplicações em tarefas mais complexas de Processamento de Linguagem Natural. Abney (20) considera esta tarefa como uma precursora do *parsing* completo[2], pois fornece uma base para outros níveis de análise como a identificação de estruturas do tipo verbo-argumento.

2.2
Motivação

Tarefas de Processamento de Linguagem Natural têm ganhado cada vez mais importância em nossa sociedade e inclui uma diversidade de tópicos que, por sua vez, se preocupam em processar e entender línguas humanas (21). De maneira geral, Kalita et al. (21) classificam as tarefas de PLN em dois grupos, ou subáreas. De todas as tarefas envolvendo PLN, algumas podem não ter um distinção clara sobre qual subárea elas pertencem. A primeira subárea está mais ligada à linguística e ataca problemas como a modelagem de linguagem, que estuda as associações entre palavras; processamentos morfológicos, sintáticos e semânticos. A segunda subárea envolve aplicações mais diretas, como

[1] *shallow parsing*
[2] *full parsing*

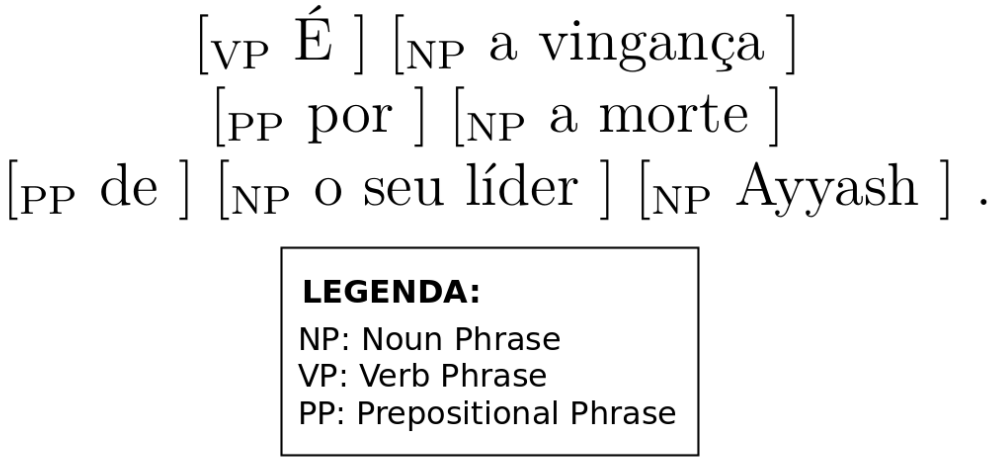

Figura 2.1: Divisão em segmentos.

extração de entidades nomeadas (22), tradução de textos (23), sumarização automática (24), classificação de documentos e outras, como agrupamento de documentos textuais. Em muitos casos, o sucesso das aplicações mais diretas vai depender de informações linguísticas que são atacados pela primeira subárea.

Segundo Hammerton et al. (25) nem todas as aplicações de PLN necessitam informações sintáticas completas e detalhadas. Na maioria dos casos, um *parser* completo provê mais informações que o necessário, em alguns casos pode ser insuficiente. Exemplos de aplicações que pouco se beneficiam das informações sintáticas completas são as tarefas de recuperação da informação, sumarização automática e sistemas de perguntas e respostas. A razão disto é que nessas tarefas é importante saber informações sintático-semânticas como agentes, objetos, localização e suas relações. Dessa maneira, é necessário um *parser* capaz de fornecer informações sintáticas que não seja tão detalhado como o *parser* completo.

A tarefa de segmentação de texto, também conhecida por *parsing* superficial, é justamente o tipo de tarefa intermediária que fornece informações sintáticas de maneira limitada, porém, muito útil para aplicações mais complexas. Collins (26) mostra que um *parser* superficial é capaz de reduzir o espaço de busca em *parsers* profundos. Outra aplicação da tarefa é em sistemas de perguntas e respostas, para processar eficientemente grandes quantidades de textos potencialmente mal-formados (27).

A utilidade da segmentação de texto sintagmas se estende além das tarefas de aplicação. Problemas de linguística também se beneficiam dos atributos

extraídos de um *parser* superficial. Fernandes et al. (28) mostram que a tarefa de identificação de orações por meio de Aprendizado de Máquinas apresenta melhores resultados quando usam atributos de *chunk* como entrada. Já Crestana et al. (29) usam-os para melhorar a tarefa de análise de dependências. Mais recentemente, Lacroix (30) usou sintagmas nominais como atributos para a tarefa de *Universal Dependencies* (31) obtendo bons resultados.

2.3 Segmentação de texto inglês

Um dos primeiros trabalhos a empregar técnicas de Aprendizado de Máquina para a tarefa de segmentação de texto em sintagmas[3] foi o de Ramshaw e Marcus (32). Nesse trabalho, os autores criaram um extrator automático de sintagmas usando a técnica de *Transformation-Based Learning* (TBL), inicialmente proposta por Brill (33), para o inglês. Para usar a TBL de forma efetiva, eles relacionaram cada palavra do corpus com sua etiqueta morfossintática [4] correspondente, assim como o tipo do *chunk*. Com isso, torna-se possível abordar o problema de segmentação como um problema de classificação. Para treinar o extrator, Ramshaw e Marcus usaram o corpus do *Wall Street Journal* que compõe o *Penn Treebank* (34).

O problema de segmentação de texto ganhou notoriedade a partir da CoNLL-2000 (18). Nesta conferência, pesquisadores foram estimulados a desenvolver algoritmos para resolver este problema para a língua inglesa. Para isso, os organizadores definiram uma tarefa compartilhada provendo conjuntos de treino e teste para os participantes. Os vencedores desta competição foram Kudoh e Matsumoto (35), ao proporem um modelo baseado em Máquinas de Vetores de Suporte.

Até meados de 2011, os melhores resultados para o problema de *chunking* utilizavam métodos estatísticos como Modelos Escondidos de Markov, Modelos de Entropia Máxima (36) e *Conditional Random Fields* (CRF) (37). Em 2011, Collobert et al. (17) propuseram um modelo baseado em Redes Neurais Convolutivas, sendo a última camada um CRF, para quatro problemas de rotulamento de sequências, dentre eles o problema de *chunking*. Esta abordagem foi a primeira a atacar o problema por meio do que se conhece hoje como Aprendizagem Profunda, ou Aprendizado Profundo. Nesse trabalho, os autores reportam um $F_{\beta=1}$ de 94,32% para a tarefa, esta métrica corresponde à uma média harmônica entre a precisão e a sensibilidade. Apesar de não su-

[3] *text chunking*
[4] *part-of-speech tag*

perar o estado-da-arte, esse trabalho inspirou outros pesquisadores a atacarem problemas de rotulamento de sequências por meio de redes neurais profundas.

2.4 Segmentação de texto em português

Apesar de ser um tópico bastante estudado na língua inglesa, sendo a tarefa da CoNLL-2000 o *benchmark* principal em diversos trabalhos sobre segmentação de texto, poucos trabalhos sobre esse tópico existem para a língua portuguesa. Destes, a maioria concentra-se em estudar a segmentação em sintagmas nominais (38), (39). Em (38), o autor aplica a técnica de *Transformation Based Learning* para segmentação de texto português. Outro trabalho nessa linha é o de dos Santos et al.(40), nele os autores constroem um corpus anotado para sintagmas nominais chamados sintagma nominal reduzido para, então, usar TBL para extração automática. Já (39) aproveita as características da TBL e a combina com Cadeias Escondidas de Markov e técnicas semi-supervisionadas para extrair sintagmas nominais a partir de poucos exemplos rotulados. Milidiú et al. (39), assim como dos Santos (41), usa a técnica de *Entropy Guided Learning* para segmentar sintagmas nominais.

Com isso, o trabalho mais próximo ao da CoNLL-2000 para a língua portuguesa é o de Ferreira (1). Nesse trabalho, o autor propõe uma heurística capaz de extrair três tipos de sintagmas diferentes a partir do corpus Bosque, do projeto Floresta Sintá(c)tica. Para avaliar a qualidade de sua heurística, o autor usa seu modelo como uma etapa anterior em dois *pipelines* de tarefas mais complexas, a saber: análise de dependências e identificação de orações. Seus resultados mostram que o modelo é capaz de melhorar a qualidade das duas tarefas citadas. Até onde sabemos, não existe nenhum trabalho que emprega redes neurais profundas para o problema de segmentação de texto em português.

2.5 Corpus Bosque

Nesta seção, descreveremos o corpus utilizado neste trabalho. Ele é proveniente do projeto Floresta Sintá(c)tica que, por sua vez, trata-se de um *treebank* público para a língua portuguesa construído em parceria entre a Linguateca 1 e o projeto VISL 2 (42) nos anos 2000. Atualmente, o projeto possui quatro corpora provenientes de diversas fontes, são eles: Bosque, Selva, Amazônia e Floresta Virgem.

O primeiro deles, o Bosque (43), é tido como o mais correto, pois é passado por processos de revisão constante desde de 2007, onde inconsistências

são corrigidas e novas etiquetas adicionadas. Segundo os responsáveis pelo projeto, este corpus é aconselhado para pesquisas que priorizam a qualidade dos resultados (43). Ele é composto por uma variedade de textos escritos tanto em português brasileiro como na variante europeia, originários de diferentes fontes. Todos estes textos possuem anotação sintática obtida automaticamente pelo *parser* PALAVRAS (44).

No total, o Bosque é composto por 9.368 frases construídas a partir dos primeiros 1000 extratos dos corpora CETENFolha, com textos escritos em português brasileiro; e CETENPúblico, que possui textos em português europeu. O corpora CETENFolha foi criado a partir de textos obtidos do jornal Folha de São Paulo. Enquanto o CETENPúblico foi construído a partir de textos extraídos do jornal PÚBLICO. A Tabela 2.1 descreve as estatísticas do Bosque.

Número de tokens	226.758
Número de sentenças	9.368
Média de tokens por sentença	24,2
Número de sintagmas nominais (NP)	61.537
Número de sintagmas verbais (VP)	21.747
Número de sintagmas preposicionais (PP)	32.949
Número de sintagmas adjetivais (ADJP)	9.608
Número de sintagmas adverbiais (ADVP)	6.505

Tabela 2.1: Estatística do corpus Bosque

O Bosque, assim como os demais corpora do projeto Floresta Sintá(c)tica, está disponível em diversos formatos, a saber: árvores deitadas (AD), formato CG, formato *PennTreeBank*, SQL, *SimTreeML* e outros. Além dos arquivos nos formatos citados, o projeto fornece uma documentação detalhada sobre cada um deles. As Figuras 2.2 e 2.3 exemplificam a frase **"O 7 e Meio é um ex-libris da noite algarvia"** codificada nos formatos AD e *PennTreeBank*, respectivamente.

```
A1
STA:fcl
=SUBJ:np
==>N:art('o' <artd> M S) O
==H:prop('7_e_Meio' M S) 7_e_Meio
=P:vp
==MV:v-fin('ser' PR 3S IND) é
=SC:np
==>N:art('um' <arti> M S) um
==>N:ec('ex-') ex-
==H:n('libris' M P) libris
==N<:pp
===H:prp('de' <sam->) de
===P<:np
====>N:art('o' <-sam> <artd> S) a
====H:n('noite' <np-def> F S) noite
====N<:adjp
=====H:adj('algarvio' F S) algarvia
=.
```

Figura 2.2: Exemplo de uma sentença no formato Árvore Deitada (AD)

```
(FRASE CP1-2 (STA:fcl (SUBJ:np (>N:art:o:M_S::artd: O)
                               (H:prop:7_e_Meio:M_S: 7_e_Meio))
                      (P:vp (MV:v-fin:ser:PR_3S_IND::: é))
                      (SC:np (>N:art:um:M_S::arti: um)
                             (>N:ec:ex-:::: ex-)
                             (H:n:libris:M_P::: libris)
                             (N<:pp (H:prp:de:::: de+)
                                    (P<:np
                                      (>N:art:o:S::artd: a)
                                      (H:n:noite:F_S::np-def: noite)
                                      (N<:adjp
                                        (H:adj:algarvio:F_S::: algarvia)))))
                      (.)))
```

Figura 2.3: Exemplo de uma sentença no formato *PennTreeBank*

2.5.1 Codificação dos tipos de sintagmas

A proposta de Ramshaw e Marcus (32) de atribuir a cada *token* um rótulo correspondente ao seu tipo de sintagma tornou possível tratar o problema de segmentação de texto como um problema de classificação. Conforme descrito

nas seções anteriores, ao associar um rótulo a um *token*, é possível criar um modelo de Aprendizado de Máquina que tem como objetivo predizer o sintagma do qual um *token* faz parte.

Para que essa abordagem funcione corretamente, é necessário não somente designar ao *token* o rótulo do seu sintagma mas também informações sobre as fronteiras de cada sintagma. Por exemplo, é possível que exista uma sequência de *tokens* que pertencem a sintagmas diferentes porém possuem tipos iguais. Seria impossível, então, distinguir os dois grupos sem que fosse adicionada alguma informação sobre os limiares dos mesmos. Para solucionar esse problema, Ramshaw e Marcus definiram uma forma de codificar os rótulos de maneira a distinguir sintagmas consecutivos (32). Os autores chamaram essa codificação de IOB, e ela foi usada em seu trabalho sobre segmentação de sintagmas nominais.

A codificação proposta por Ramshaw e Marcus funciona da seguinte maneira: todos os *tokens* pertencentes a um sintagma são rotulados como I-CHUNK, onde CHUNK é um tipo de sintagma como NP, VP e etc. Todos os símbolos que não possuem classificação sintagmática, como pontuação, são rotulados como O. Por último, sempre que dois sintagmas da mesma classe ocorrem seguidamente, o primeiro *token* do segundo grupo em diante recebe o rótulo B. Devido à sua flexibilidade, essa abordagem funciona independente do tipo do sintagma.

Ao longo dos anos, outras codificações surgiram, como a IOB2, IOE1, IOE2 e IOBES. A primeira foi utilizada pela primeira vez na tese de doutorado de Ratnaparkhi (45) e posteriormente escolhida como codificação oficial da CoNLL-2000. Essa codificação é muito semelhante à IOB1, se diferenciando pela forma como determina as fronteiras entre os segmentos. Nela, todo *chunk* se inicia com o rótulo B e os demais *tokens* pertencentes ao *chunk* recebem o rótulo I. Assim como a IOB1, palavras que não pertencem a nenhum sintagma são classificados como O.

A segunda e a terceira codificação se assemelham bastante à IOB1 e IOB2. Elas são apresentadas no trabalho de (46) e tem como objetivo determinar o final de cada *chunk*. Ou seja, na codificação IOE1, a última palavra de um sintagma que precede outro sintagma do mesmo tipo recebe a marcação E. Já na IOE2, toda última palavra de um *chunk* recebe o rótulo E.

Por último, a codificação IOBES, usada no trabalho de (17), usa a marcação S para determinar sintagmas que possuem apenas uma palavra; B para demarcar o início de um sintagma composto de múltiplas palavras; I para marcar os *tokens* no interior de um sintagma e; por fim, E para marcar o fim de um sintagma. Para exemplificar essas codificações, a Tabela 2.2 mostra a

sentença **"É a vingança por a morte de o seu líder Ayyash."** rotulada usando todas as codificações citadas.

Token	**POS**	**IOB1**	**IOB2**	**IOE1**	**IOE2**	**IOBES**
É	V	I-VP	B-VP	I-VP	E-VP	S-VP
a	ART	I-NP	B-NP	I-NP	I-NP	B-NP
vingaça	N	I-NP	I-NP	I-NP	E-NP	E-NP
por	PREP	I-PP	B-PP	I-PP	E-PP	S-PP
a	ART	I-NP	B-NP	I-NP	I-NP	B-NP
morte	N	I-NP	I-NP	I-NP	E-NP	E-NP
de	PREP	I-PP	B-PP	I-PP	E-PP	S-PP
o	ART	I-NP	B-NP	I-NP	I-NP	B-NP
seu	PROADJ	I-NP	I-NP	I-NP	I-NP	I-NP
líder	N	I-NP	I-NP	E-NP	E-NP	E-NP
Ayyash	NPROP	B-NP	B-NP	E-NP	E-NP	S-NP
.	.	O	O	O	O	O

Tabela 2.2: Exemplo dos estilos de marcação IOB1, IOB2, IOE1, IOE2 e IOBES.

2.5.2 Heurística para extração de sintagmas

Em 2011, Ferreira (1) propôs uma heurística para extração de *chunks* do corpus Bosque, pois o mesmo não provê uma anotação de *chunk* para cada *token*. Para isso, o autor inspirou-se na mesma técnica empregada na tarefa compartilhada da CoNLL-2000. A estratégia consistia em determinar que um segmento, ou *chunk*, é composto por todos os *tokens* consecutivos dentro do mesmo nó mais profundo da árvore sintática (1). Como todos os *tokens* são nós terminais da árvore, um *token* pertence a um *chunk* derivado do seu sintagma pai mais próximo. A Figura 2.4 ilustra um exemplo semelhante aos exemplos presentes no Bosque.

Nesta figura, o sintagma preposicional "de a escola" gera dois segmentos distintos. O primeiro deles, "de", é do tipo PP, pois o nó sintagmático mais profundo da árvore que inclui este *token* é preposicional. Já o segundo é um segmento NP, que é composto pelas palavras "a escola", pois o nó mais profundo é um sintagma nominal.

2.5.3 Definições de segmentos

O objetivo de Ferreira (1) com a extração dos segmentos a partir do corpus Bosque era o de melhorar o desempenho de tarefas mais complexas, mesmo que essas definições não fossem exatamente corretas do ponto de vista

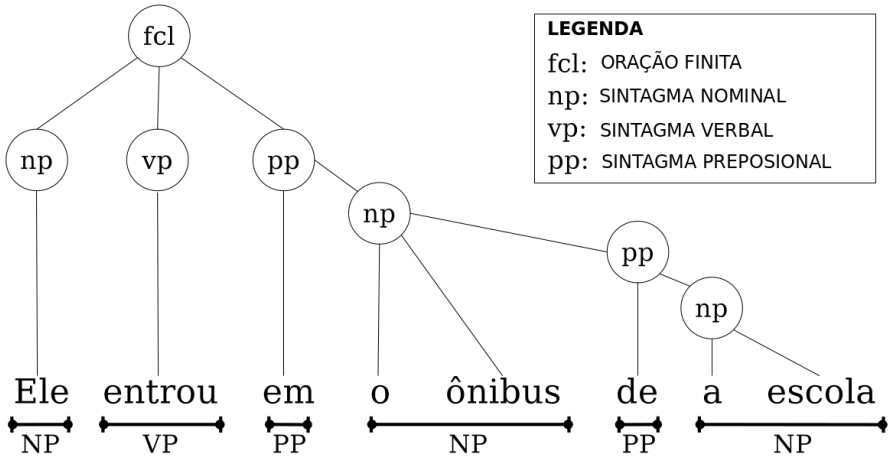

Figura 2.4: Saída obtida pela heurística extratora de segmentos. Adaptado de Ferreira (1).

linguístico. Com isso, uma de suas contribuições foi a de definir diferentes definições de *chunks* para o Bosque. Em seu trabalho, foi determinado que uma definição se caracterizava por um subconjunto de tipos de segmentos obtidos a partir de tipos de sintagmas selecionados. Por exemplo, a definição (NP, VP) só considera os sintagmas nominais e sintagmas verbais. Dessa forma, uma sentença extraída do corpus que possua um nó na árvore rotulado como PP terá esse rótulo desconsiderado. Assim, o *chunk* receberá o rótulo do sintagma ancestral mais próximo. Para ficar mais claro, a Tabela 2.3 ilustra a extração de segmentos usando duas definições, (NP, VP) e (NP, VP, PP). Observe que na definição (NP, VP), o *token* "de", que tem como nó pai um sintagma PP, recebe o rótulo do ancestral mais próximo diferente de PP, ou seja, o rótulo NP.

Palavra	**POS**	**(NP, VP)**	**(NP, VP, PP)**
Ele	pron-pers	B-NP	B-NP
entrou	v-fin	B-VP	B-VP
em	prp	O	B-PP
o	art	B-NP	B-NP
ônibus	n	I-NP	I-NP
de	prp	I-NP	B-PP
a	art	B-NP	B-NP
escola	n	I-NP	I-NP
.	.	O	O

Tabela 2.3: Sequências de *chunks* obtidos pela heurística para duas definições (1).

O corpus Bosque possui anotações de *parsing* completo de cinco tipos de sintagmas: Sintagmas Nominais (NP), Sintagmas Verbais (VP), Sintagmas

Preposicionais (PP), Sintagmas Adjetivais (ADJP) e Sintagmas Adverbiais (ADVP). Com isso, Ferreira propôs três diferentes definições:

1. (NP, VP)
2. (NP, VP, PP)
3. (NP, VP, PP, ADJP, ADVP)

A Tabela 2.4 exibe as estatísticas para cada uma definição obtida pela heurística.

Definição	**NP**	**VP**	**PP**	**ADJP**	**ADVP**
(NP, VP)	74.104	21.236	-	-	-
(NP, VP, PP)	68.166	21.232	34.321	-	-
(NP, VP, PP, ADJP, ADVP)	68.536	21.235	34.129	9.586	6.440

Tabela 2.4: Sequências de *chunks* obtidos pela heurística para duas definições (1).

3
Conceitos Teóricos

Neste capítulo descreveremos os conceitos teóricos em que se baseia este trabalho. Apresentaremos o Aprendizado de Máquina e um dos seus subcampos, o Aprendizado Profundo. Este último vem ganhando bastante atenção recentemente devido aos seus resultados extraordinários, superando drasticamente o estado-da-arte em diversas tarefas. Todo esse sucesso não é por menos, uma das características responsáveis é a capacidade que modelos compostos de múltiplas camadas têm de aprender múltiplos níveis de abstração (16).

Além dos conceitos mais gerais, serão discutidos em mais detalhe as redes neurais, que são exemplos de modelos de múltiplas camadas. Algumas variações dessas redes são especialmente úteis para problemas envolvendo sequências, como é o caso das Redes Neurais Recorrentes. Por último, será apresentado o conceito de Representação Textual e como a forma como o texto é modelado ajuda a melhorar os resultados nos problemas abragendo textos.

3.1
Aprendizado de Máquina

O Aprendizado de Máquina é uma sub-área da Inteligência Artificial que preocupa-se em construir programas de computador que aprendem automaticamente com a experiência. Formalmente, diz-se que um programa de computador aprende com a experiência **E** com relação a alguma classe de tarefas **T** e medida de desempenho **P**, se seu desempenho em tarefas em **T**, como medido por **P**, melhora com a experiência **E** (47).

Essa tecnologia já é parte integrante do dia-a-dia da sociedade moderna, estando presente em agentes conversacionais (48), nos filtros de *spam* (49), nos sistemas de recomendação (50), e muitos outros. É tão versátil que está inclusive na palma de nossas mãos por meio dos *smartphones*.

Suas aplicações são das mais variadas e têm sido usadas para tradução de textos (51), reconhecimento de objetos (52), sistemas de perguntas e respostas (53), predição de séries temporais (54) e outros mais. Todavia, técnicas convencionais de Aprendizado de Máquina, como SVMs, K-Means, Árvores de Decisão e etc, possuem limitações importantes. Apesar de serem

estudadas há anos, essas técnicas não obtêm bons resultados sem que haja uma engenharia de atributos manual para o treinamento desses algoritmos (16).

Em outras palavras, técnicas convencionais de AM têm dificuldade de aprender atributos automaticamente a partir de dados brutos. Esse tipo de aprendizado é conhecido como Aprendizado de Representações[1] (55) e é possível por meio de técnicas mais complexas, como as redes neurais.

3.2 Aprendizado Profundo

Os métodos convencionais de Aprendizado de Máquina possuem outras limitações além da incapacidade de aprender características automaticamente. Eles não generalizam bem quando aplicados a determinados problemas como reconhecimento de falas, identificação de objetos e classificação de imagens. Isto porque generalizar novos exemplos onde os dados são de alta dimensão é uma tarefa exponencialmente difícil (56).

Nesse cenário, os mecanismos empregados pelos métodos tradicionais de AM para generalizar novos exemplos são incapazes de aprender funções complexas em espaços de alta dimensão. O desenvolvimento do Aprendizado Profundo surgiu, então, com a motivação de superar tais limitações. Isso é possível pois modelos compostos de diversas camadas podem aprender representações de dados com múltiplas camadas de abstração (16).

O que torna esse tipo de aprendizado especial é o fato das técnicas de Aprendizado Profundo conseguirem descobrir estruturas complexas em grandes conjuntos de dados por meio do algoritmo *backpropagation* (16). O objetivo do *backpropagation* é auxiliar no ajuste dos parâmetros internos das redes neurais. Tais parâmetros são usados na computação das representações em cada camada a partir das camadas anteriores.

3.3 Redes Neurais Artificiais

Com origem nos trabalhos de McCulloch e Pitts no ano 1943 (57), as redes neurais surgiram como uma modelagem matemática do comportamento de um neurônio biológico. Nessa modelagem, um neurônio possui dois estados possíveis, o de ativo e inativo. O neurônio torna-se ativo quando sua saída supera um limiar estabelecido. A partir da evolução desse conceito outras modelagens foram surgindo, como as Redes Neurais *Feedforward*, também conhecidas como *Perceptron* de Múltiplas Camadas [2]

[1] Representation Learning, ou Feature Learning.
[2] Multi-layer Perceptron.

Esse tipo de rede neural é a base para diversas aplicações comerciais, que por sua vez se utilizam de diversos tipos especializados desse conceito. Exemplos clássicos são as Redes Neurais Convolutivas, muito utilizadas no reconhecimento de objetos; e as Redes Recorrentes, usadas em aplicações de processamento de linguagem natural (56).

O objetivo de uma rede *feedforward* é aproximar uma função f (55). Em uma tarefa de classificação, por exemplo, a função $f(\boldsymbol{x}) = y$ mapeia a entrada $\boldsymbol{x}$ para a categoria y. Dessa maneira, objetivo é aprender parâmetros $\boldsymbol{\theta}$ que resultam na melhor aproximação de $f(\boldsymbol{x}; \theta) = y$. Esse tipo de rede recebe esse nome pois a informação de entrada $\boldsymbol{x}$ flui através de f, tendo como saída a classificação $\boldsymbol{y}$. Outra propriedade importante é o encadeamento dessas funções por composição, isto é, uma rede de múltiplas camadas é representada por $f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$, onde $f^{(1)}$ é a primeira camada, $f^{(2)}$ é a segunda camada e $f^{(3)}$ a terceira. A imagem 3.1 exibe uma Rede *Feedforward* com quatro camadas, sendo uma de entrada, uma de saída e duas camadas escondidas.

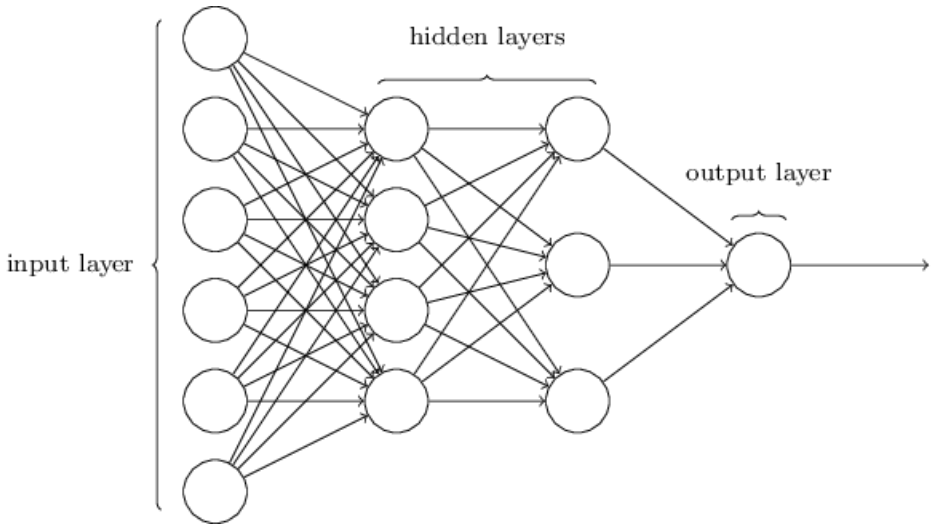

Figura 3.1: Exemplo de *Perceptron* de Múltiplas Camadas (2).

3.3.1 Redes Neurais Recorrentes

As Redes Neurais Recorrentes, ou simplesmente RNNs, são um tipo de especializado de rede *feedforward*, como discutido na seção anterior, especialmente usadas em problemas de natureza sequencial (56), como dados textuais e séries temporais. Devido à sua versatilidade, as RNNs conseguem lidar com sequências de valores $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$ fixos e também variáveis, inclusive para sequências muito longas.

A característica central das RNN é que suas conexões possibilitam reter a memória das entradas anteriores, que permanecem no estado interno da rede e influenciam os resultados da saída, como exemplifica a imagem 3.2. Isto é especialmente importante quando os dados de entrada têm relações de dependência entre si. Existem diversas variações de redes recorrentes como as RNNs tradicionais, Redes Elman (58), Redes Jordan (59), LSTM[3] (60), e GRU[4] (23).

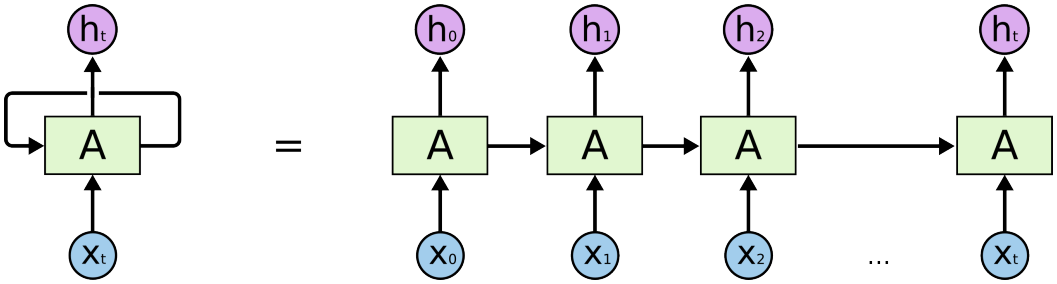

Figura 3.2: Exemplo de uma rede recorrente "desenrolada". (3).

Quando comparadas ao *Perceptron* de Múltiplas Camadas, a principal diferença é que nesse último os vetores de entrada são mapeados para vetores de saída. Tais vetores são ambos de tamanho fixo, ou seja, dado um vetor de entrada com tamanho fixo, a saída da rede será também um vetor de tamanho fixo. Já uma RNN pode mapear um histórico inteiro de entradas anteriores para cada saída (5).

A Figura 3.3 deixa mais claras as diferenças entre as RNNs e os MLPs. O primeiro diagrama demonstra um problema onde a entrada e a saída são fixas, como o problema de classificação de imagens. O segundo representa um problema onde a entrada é fixa e a saída possui tamanho variável, um exemplo de problema nessa linha é o de legendagem de imagens. O próximo diagrama representa um problema onde a entrada e a saída possuem tamanho variável, como o problema de tradução de textos. Por último, os problemas onde a entrada e a saída são sequências fixas, como na classificação de vídeos onde cada quadro é rotulado.

Formalmente, Chung et al. (61) definem uma RNN tradicional como uma rede que recebe como entrada uma sequência $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$ e possui um estado escondido h_t, que é atualizado periodicamente de acordo com a equação (3-1), onde ϕ é uma função não-linear que pode ser obtida através da composição da função *sigmoid* com uma transformação afim. A saída da RNN

[3] *Long Short-Term Memory*
[4] *Gated Recurrent Unit*

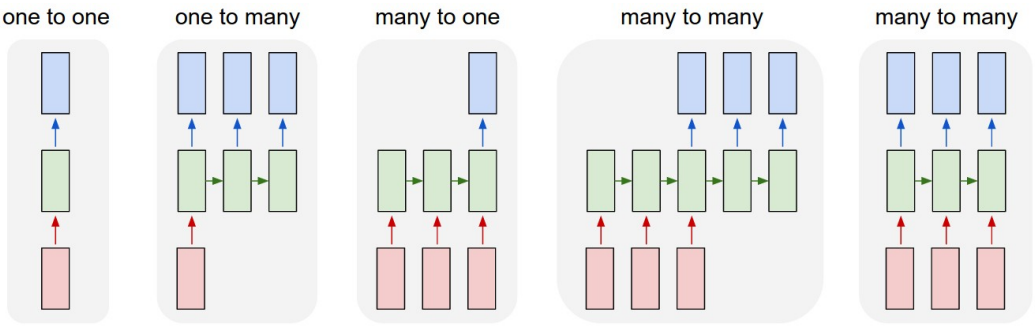

Figura 3.3: Problemas envolvendo dados sequenciais. (4).

é uma sequência de tamanho variável definida por $\mathbf{y} = (y_1, y_2, \ldots, y_T)$.

$$\mathbf{h}_t = \begin{cases} 0, & t = 0 \\ \phi\left(\mathbf{h}_{t-1}, \mathbf{x}_t\right), & \text{caso contrário} \end{cases} \tag{3-1}$$

A atualização dos estados escondidos recorrentes é implementada como

$$\mathbf{h}_t = g\left(W\mathbf{x}_t + U\mathbf{h}_{t-1}\right), \tag{3-2}$$

onde g é uma função suave e limitada, como uma função sigmóide logística ou uma função tangente hiperbólica. A imagem 3.4 descreve a estrutura interna de uma RNN.

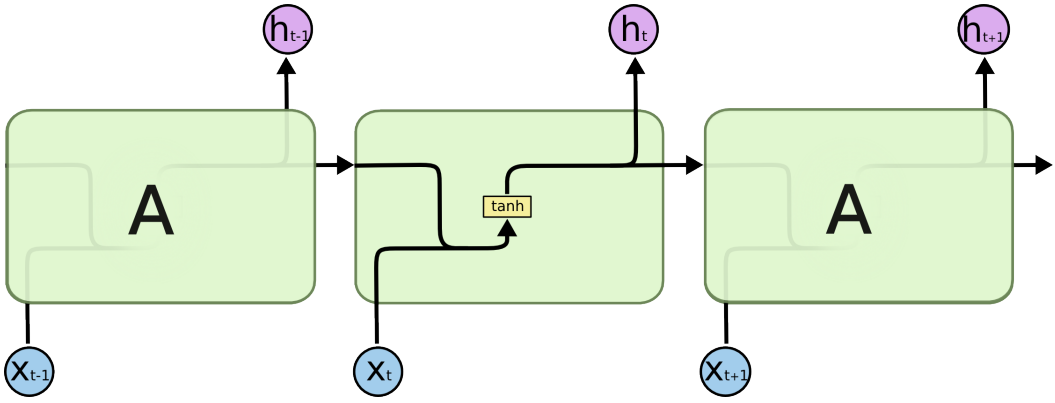

Figura 3.4: Exemplo de uma célula de uma RNN. (3).

Apesar dessa interessante característica de modelar sequências de tamanho ilimitado, a RNN tradicional possui uma limitação importante que, por sua vez, dificulta sua capacidade de aprender. Em problemas onde a entrada é caracterizada por longas dependências, o algoritmo usado para a correção dos parâmetros internos não se comporta bem. A razão para isso é que a RNN mantém um vetor de ativação para cada *timestep* e, ao propagar o resultado, o cálculo do gradiente desses valores tende a explodir ou esvair (62).

A imagem 3.5 descreve o processo do esvaimento do gradiente[5] (5). A tonalidade dos nós da rede indica a sensibilidade da entrada ao longo do tempo. Os tons mais escuros apontam uma sensibilidade maior para aquela entrada. A medida em que o tempo passa, essa sensibilidade diminui, pois as novas entradas sobrescrevem as ativações da camada escondida. Dessa maneira, a rede perde a memória das primeiras entradas.

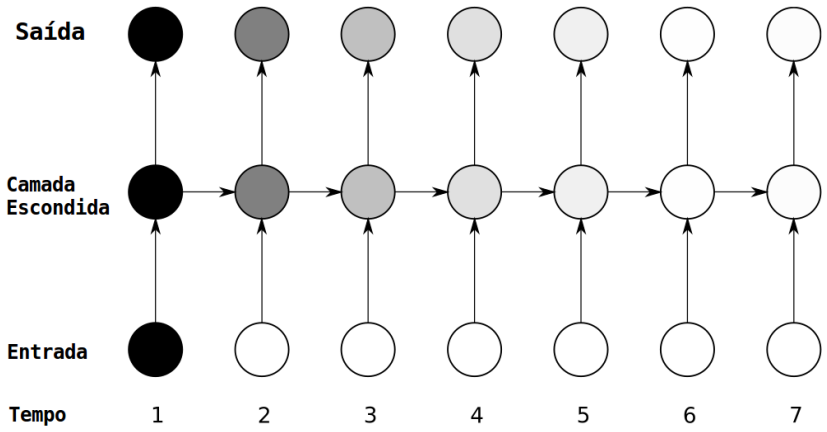

Figura 3.5: O problema do esvaimento do gradiente. Adaptado de Graves, 2012 (5).

3.3.2 Long Short-Term Memory

Para mitigar o problema de esvaimento do gradiente, Sepp Hochreiter e Jürgen Schmidhuber (60) proporam a rede *Long Short-Term Memory*. A LSTM é considerada como um tipo de RNN, pois é bastante semelhante à RNN tradicional, com notáveis diferenças que a torna mais apropriada para problemas onde a entrada apresenta longas dependências.

Uma diferença importante se dá no conceito de blocos de memória conectados. Cada um desses blocos possui um ou mais blocos de memória auto conectados, assim como três unidades multiplicativas (5). Essas unidades permitem que as células de memória armazene e acesse informações de períodos passados muito distantes. Dessa maneira, as novas entradas não sobrescrevem o estado atual da célula quando a porta de entrada está fechada, o que torna essa informação disponível mais adiante.

A Figura 3.6 descreve o processo de preservação do gradiente. Como na Figura 3.5, os nós mais escuros indicam maior sensibilidade, enquanto os mais claros indicam menor sensibilidade. Na imagem, as portas que regulam a propagação de informação são representadas pelos símbolos **O** e __. O primeiro

[5]Vanishing Gradient Problem

símbolo indica que a porta está aberta, ou seja, a informação proveniente dessa entrada será propagada. O segundo símbolo indica que a informação não será propagada. Dizemos que uma porta está fechada quando sua saída é próxima de zero; e aberta quando sua saída é igual a 1. Com isso, a sensibilidade dos blocos podem ser reguladas sem afetar o estado atual da célula, o que faz com que a LSTM seja mais apropriada para problemas de longas dependências.

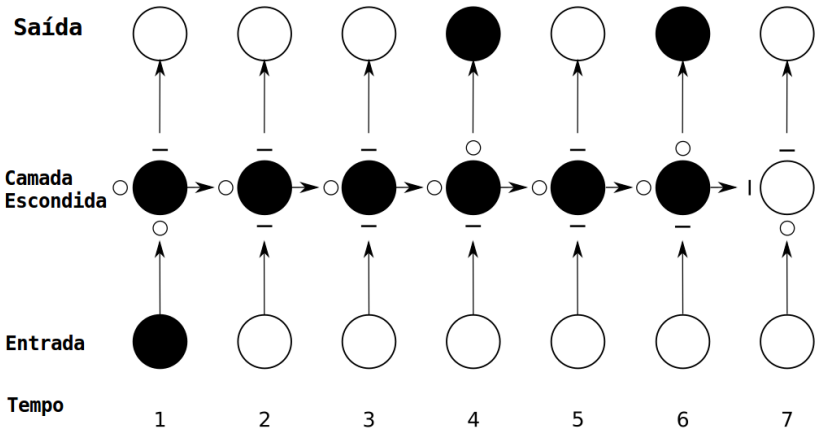

Figura 3.6: Preservação do gradiente em uma rede LSTM. Adaptado de Graves, 2012 (5).

Chung et al. (61) definem formalmente uma LSTM da seguinte forma: cada j-th unidade de uma rede LSTM possui uma memória c_t^j no tempo t; diferentemente da RNN tradicional, que calcula uma soma ponderada a partir de cada entrada e aplica uma função não-linear sobre o resultado. A saída de cada unidade é representada por h_t^j, e a unidade como um todo é definida como

$$h_t^j = o_t^j \tanh\left(c_t^j\right),$$

onde o_t^j é a porta que regula a quantidade de informação contida na memória que será exposta.

A porta de saida é computada como

$$o_t^j = \sigma\left(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c}_t\right)^j,$$

onde σ representa a função sigmoid.

A célula de memória c_t^j é atualizada por meio do "esquecimento" parcial da memória existente e pela adição de novas informações provenientes de $\tilde{c}_t^j$:

$$c_t^j = f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j, \tag{3-3}$$

Já $\tilde{c}_t^j$ é obtido através de:

$$\tilde{c}_t^j = \tanh\left(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1}\right)^j .$$

A quantidade de memória atual a ser esquecida é modulada pela "porta de esquecimento"[6] f_t^j; já a quantidade de memória nova a ser adicionada é regulada pela "porta de entrada"[7] i_t^j. Todos estes cálculos são obtidos por meio das equações:

$$\begin{aligned} f_t^j &= \sigma\left(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_f \mathbf{c}_{t-1}\right)^j , \\ i_t^j &= \sigma\left(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c}_{t-1}\right)^j . \end{aligned}$$

A Figura 3.7 descreve um bloco da rede LSTM.

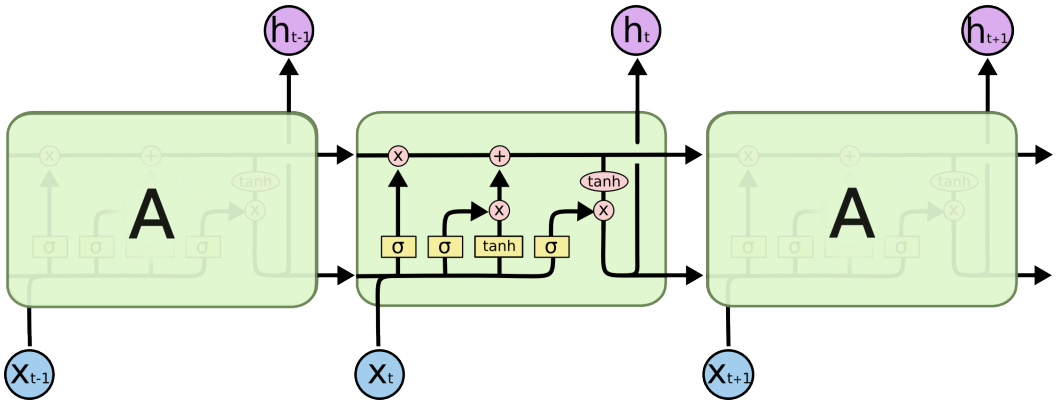

Figura 3.7: Bloco de uma rede LSTM. (3).

3.3.3 Gated Recurrent Unit

Apesar do sucesso das LSTMs em solucionar o problema do esvaimento do gradiente, algumas variações foram propostas após a publicação de Hochreiter e Schmidhuber, como a proposta de Gers et. al. (63). Essas variações possuem poucas mudanças em relação à LSTM de Hochreiter e Schmidhuber, mantendo praticamente a mesma complexidade em relação ao número de portas e consumo de recursos computacionais. Como forma de mitigar esses problemas, mas mantendo suas boas características, Cho et. al. (23) propuseram as *Gated Recurrent Units*, ou apenas GRUs, onde os autores demonstraram sua aplicabilidade no problema de tradução textual.

A motivação dos autores com a GRU, além da melhor eficiência e redução da complexidade em relação às LSTMs, era de fazer com que cada unidade

[6] forget gate
[7] input gate

recorrente capturasse dependências adaptativamente de diferentes escalas de tempo. Assim como na LSTM, as GRU possuem portas que modulam o fluxo de informação dentro de uma unidade. Sua implementação, porém, é mais simples. A diferença, então, se dá pela não existência de células de memória separadas (23).

Seguindo novamente a definição formal apresentada por Chung et al. (61), uma unidade GRU é composta de uma camada de ativação h_t^j no tempo t. Esta camada é uma interpolação linear entre a ativação anterior, denotada por h_{t-1}^j e a ativação candidata, representada por $\tilde{h}_t^j$:

$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \tilde{h}_t^j, \tag{3-4}$$

onde a porta de entrada z_t^j tem a função de decidir o quanto a unidade vai atualizar sua ativação ou conteúdo. A atualização é calculada por meio da equação (3-5).

$$z_t^j = \sigma \left(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1} \right)^j . \tag{3-5}$$

De maneira semelhante a uma célula LSTM, as equações anteriores fazem uma soma linear entre o estado existente e o estado recentemente computado. A exceção, porém, se dá no controle do grau em que o estado é exposto. No caso da GRU, ela não possui um mecanismo para esse controle, com isso, expõe o estado completo sempre.

A ativação candidata $\tilde{h}_t^j$ é calculada de maneira similar a de uma unidade recorrente tradicional, onde $\mathbf{r}_t$ é um conjunto de *reset gates* e $\odot$ é uma multiplicação de elemento por elemento.

$$\tilde{h}_t^j = \tanh \left(W \mathbf{x}_t + U \left(\mathbf{r}_t \odot \mathbf{h}_{t-1} \right) \right)^j ,$$

Quando a porta r_t^j é próxima de zero, ela permite que a unidade esqueça o estado computado anteriormente. Em suma, é como se a unidade estivesse lendo o primeiro símbolo da sequência de entrada. Essa operação é semelhante à operação realizada pela porta de atualização, sendo r_t^j definida como:

$$r_t^j = \sigma \left(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1} \right)^j .$$

Por fim, a Figura 3.8 descreve uma unidade GRU.

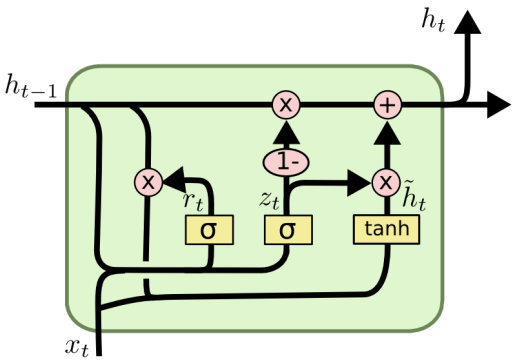

Figura 3.8: Unidade de uma rede GRU (3).

3.4 Conditional Random Fields

Como discutido anteriormente, problemas envolvendo sequências de texto, como análise sintática e segmentação de texto, eram abordados, em geral, de duas maneiras distintas. A primeira consistia na construção de modelos de aprendizado de máquina generativos, como as cadeias escondidas de Markov. Já a segunda abordagem via os problemas de rotulamento de sequências como uma sequência de problemas de classificação, ou seja, o rótulo de uma palavra pode depender das classificações anteriores (64).

Nos modelos generativos, o objetivo é maximizar a probabilidade conjunta no conjunto de dados de treinamento. Um problema associado a isso é que este objetivo não está diretamente ligado às métricas de interesse da tarefa. Já os modelos sequenciais são treinados para minimizar uma função de custo associada ao erro no rotulamento que, por sua vez, acarreta em erros cada vez menores a medida em que mais dados de treinamento são usados.

Cada uma dessa abordagem possui prós e contras, ideal seria um modelo que extraísse o melhor de cada uma. Na prática, esse modelo existe e é conhecido como *Conditional Random Fields* (CRFs). Os CRFs surgiram em 2001, por meio do trabalho de Lafferty et al. (37), que mostrou que eles superavam os modelos baseados em HMM para a tarefa de análise sintática. Desde então, os CRFs foram aplicados para outras tarefas de rotulamento de sequências, como o *parsing* superficial (64), com grande sucesso.

Não demorou muito até os CRFs serem usados em conjunto com as redes neurais profundas para problemas de rotulamento de senquências. Até onde sabemos, o primeiro trabalho a combinar uma rede neural com um CRF como camada de saída foi o trabalho de Collobert et al. (17). Nesses cenários, os CRFs são usados para modelar as dependências entre os rótulos a partir da saída da camada escondida **h** anterior. A ideia de usar CRF para este fim se dá pela característica do problema de *chunking*, onde um rótulo específico possui

forte dependência do anterior. Por exemplo, em uma sequência, não é possível que um rótulo I-NP seja precedido por I-VP usando-se o sistema IOB2. Já um rótulo B-NP tem grande chance de ser precedido de I-NP.

Formalmente, definimos essa relação da seguinte forma: seja $\mathcal{Y}(\mathbf{h})$ o espaço das sequências de rótulos para a camada $\mathbf{h}$. A probabilidade condicional de uma sequência de rótulos $\mathbf{y}$, dado um estado escondido $\mathbf{h}$ é definida como

$$p(\mathbf{y}|\mathbf{h}) = \frac{\exp f(\mathbf{h}, \mathbf{y})}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{h})} \exp f(\mathbf{h}, \mathbf{y}')}, \tag{3-6}$$

onde f é a função que designa uma pontuação para cada par $\mathbf{h}$ e $\mathbf{y}$ (9).

A função $f(\mathbf{h}, \mathbf{y})$ é definida para cada posição t, através da multiplicação do estado escondido $\mathbf{h}_t^w$ com um vetor de pesos $\mathbf{w}_{y_t}$. Este vetor, por sua vez, é indexado pelo rótulo y_t. O objetivo dessa operação é obter a pontuação da designação do rótulo y_t na posição t. Com isso, $f(\mathbf{h}, \mathbf{y})$ é definida formalmente como

$$f(\mathbf{h}, \mathbf{y}) = \sum_{t=1}^{T} \mathbf{w}_{y_t}^T \mathbf{h}_t^w + \sum_{t=1}^{T} A_{y_{t-1}, y_t},$$

Note que $f(\mathbf{h}, \mathbf{y})$ também é definida em termos da matriz A. Seu objetivo é considerar a correlação entre os rótulos. Dessa maneira, definimos a matriz A como uma matriz de similaridade de pontuação entre os pares de rótulos. O treinamento corresponde à maximização da probabilidade condicional logarítmica

$$\log p(\mathbf{y}|\mathbf{h}) = f(\mathbf{h}, \mathbf{y}) - \log \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{h})} \exp f(\mathbf{h}, \mathbf{y}'), \tag{3-7}$$

Já a determinação do conjunto de rótulos de saída é definida da seguinte forma:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}' \in \mathcal{Y}(\mathbf{h})} f(\mathbf{h}, \mathbf{y}'). \tag{3-8}$$

A equação 3-8 é calculada eficientemente através do algoritmo de Viterbi (9).

3.5 Representação Textual

Para que possam produzir resultados satisfatórios, os dados de entrada dos algoritmos de Aprendizado de Máquina precisam ter uma representação bem definida. No caso de problemas que envolvem imagens, a entrada pode ser um vetor com os valores de intensidade dos pixels, que é considerada uma

representação densa dos dados. Problemas que envolvem texto, por outro lado, eram, até a algum tempo atrás, costumeiramente representados por meio dos modelos de *Bag-of-Words* e Frequência do Termo–inverso da Frequência nos Documentos[8].

Representações textuais, como a BOW e TF-IDF, possuem algumas grandes limitações. Uma delas é o consumo de memória para representar textos com grande vocabulário. Por exemplo, no modelo *bag-of-words* tem-se um vocabulário fixo de tamanho N e um texto é representado por meio da codificação "*one-hot*" por um vetor de tamanho N. Essa representação trata cada palavra como um símbolo atômico, sem a noção de contexto presente na representação. Outra limitação, e mais importante em tarefas de linguística computacional, é o fato desses modelos não capturarem informações mais ricas do texto, como informações semânticas e outras informações de contexto.

Como forma de encontrar uma melhor representação que considerasse, entre outras coisas, o contexto, Bengio et al. (65) propuseram um modelo neural distribucional de vetores de palavras[9]. A hipótese dos autores baseava-se na característica que as palavras têm de compartilhar significado semântico quando ocorrem simultaneamente no mesmo contexto. Nesse trabalho, os autores representam cada palavra como um vetor no espaço $\mathbb{R}^n$, onde n é parametrizável, podendo assumir valores como 30, 60, 100 e etc.

Para obter essa representação densa das palavras, os autores usaram uma Rede Neural Multi-camadas. O objetivo é prever a ocorrência de uma palavra dadas as anteriores. O modelo proposto por Bengio et al. generaliza bem pois é esperado que palavras similares possuam representações vetoriais semelhantes. Apesar de bastante inovadora, a idéia não pode ser explorada em sua máxima capacidade pois necessitava de grande quantidade de textos que, por sua vez, requeriam grande poder computacional não disponível na época para processá-los.

3.5.1 Word2vec

Seguindo a linha dos trabalhos anteriores sobre representação distribuída de palavras em espaço vetorial, Mikolov et al. (6) propuseram dois modelos capazes de aprender representações de palavras de maneira eficiente. O primeiro deles, chamado de *Continuous Bag of Words*[10], tem como objetivo prever uma palavra alvo dado o contexto onde ela se encontra. Já o segundo modelo, chamado de *skip-gram* funciona da maneira inversa, ou seja, dada uma palavra

[8] term frequency–inverse document frequency, ou TF-IDF
[9] distributional neural word embedding model
[10] CBOW

o modelo tenta prever o contexto no qual ela está inserida. A obtenção dos *word embeddings* é feita, então, por meio de uma rede neural simples com uma camada escondida. Uma vez treinada, os pesos da camada escondida são usados como os vetores de palavras. A dimensão desses vetores é justamente o tamanho escolhido para a camada escondida. As Figuras 3.9 e 3.10 descrevem os modelos CBOW e *skip-gram*, respectivamente.

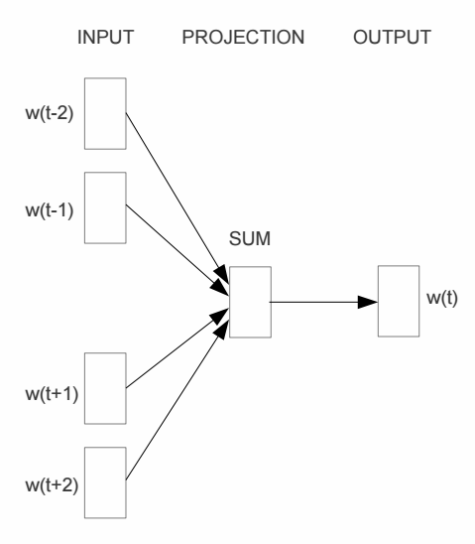

Figura 3.9: Modelo CBOW (6).

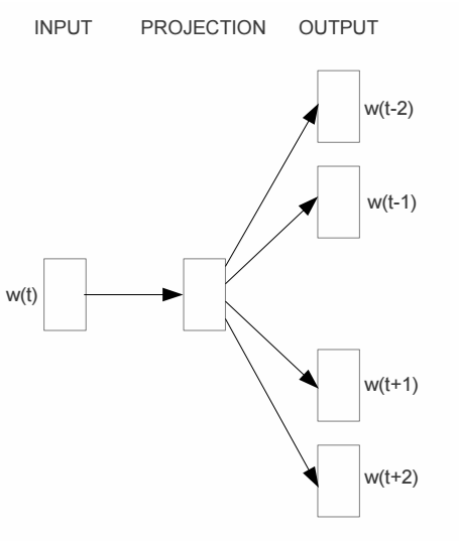

Figura 3.10: Modelo *skip-gram* (6).

Além das características linguísticas capturadas pelos modelos, treiná-los era uma tarefa bem eficiente. De acordo com os autores, na época em que foram publicados, tais modelos eram capazes de serem treinados em mais de 100 bilhões de palavras em apenas um dia. Apesar de bem eficiente, no artigo subsequente (66), os autores propuseram uma série de otimizações que não só melhoram o desempenho como a qualidade dos vetores de palavras. Uma dessas otimizações é a Amostragem Negativa[11].

A motivação por trás das otimizações propostas se baseia em algumas características do modelo inicial. Por exemplo, o modelo *skip-gram* se utiliza da função *softmax* para o cálculo da distribuição de probabilidade das palavras. Uma das premissas do *word2vec* é que quanto maior a quantidade de dados usada no treinamento, melhor será o conjunto de vetores gerado. Todavia, a medida em que o número de palavras no vocabulário aumenta, torna-se impraticável o uso da *softmax* padrão. Para tanto, os autores consideraram como alternativa o uso das técnicas *Hierarchical Softmax* e *Negative Sampling*.

Em mais detalhes, a *Hierarchical Softmax* é uma aproximação da *softmax* tradicional proposta por Morin e Bengio (67). Sua grande vantagem é a

[11] Negative Sampling

avaliação de apenas $\log_2(W)$ dos nós de saída da rede neural para a obtenção da distribuição de probabilidade. Já na Amostragem Negativa, a ideia é avaliar apenas a palavra que faz parte do contexto da palavra de entrada e selecionar aleatoriamente k palavras que não fazem parte do contexto para a obtenção das probabilidades. Ou seja, no total apenas $k+1$ nós de saída da rede neural serão usados na correção dos pesos por meio do *backpropagation*.

Além da melhora no desempenho e na qualidade do algoritmo, em seu trabalho, Mikolov et al. apresentam algumas características interessantes do *word2vec*. Uma delas é a noção clara da captura de informação rica do texto. Por exemplo, ao realizarmos operações aritméticas com os vetores de palavras obtidos, é possível notar alguns resultados como: $vec("Madrid") - vec("Spain") + vec("France") \approx vec("Paris")$. Esse resultado, sem dúvidas, demonstra o poder dos *word embeddings* para aplicações modernas de Processamento Natural de Linguagens.

3.5.2 Wang2vec

Os *word embeddings* representaram um grande avanço na área de Processamento de Linguagem Natural, sendo o *word2vec* uma das implementações que mais ganhou atenção nos últimos anos. Apesar de todas as suas vantagens, como simplicidade e eficiência, esse método possui algumas limitações que impactam determinadas tarefas linguísticas. Uma das limitações é o fato do *word2vec* não levar em consideração a ordem em que as palavras aparecem em uma sentença, o que acarreta em um resultado sub-ótimo em tarefas que envolvem sintaxe, como *part-of-speech* e análise de dependências (7).

Como forma de superar essas deficiências, Wang et al. (7) propuseram uma modificação no *word2vec* para levar em consideração a ordem das palavras. A alteração proposta foi chamada de *Structured Word2Vec*, também chamada de *wang2vec* em outros trabalhos. Em seu trabalho, os autores mostraram que essas modificações resultaram em melhoras tanto para *part-of-speech*, quanto para análise de dependências.

No caso do modelo *skip-gram* do *word2vec*, cujo objetivo é prever uma janela de palavras de tamanho c que tem como centro palavra w_0, o modelo usa uma matriz $O \in \mathbb{R}^{|V| \times d}$ para prever todas as $w_{-c}, ..., w_{-1}, w_{-1}, ..., w_c$ palavras. A proposta de Wang et al. (7) é definir um conjunto de $c \times 2$ matrizes $O_{-c}, ..., O_{-1}, O_{-1}, ..., O_c$ com tamanho $O \in \mathbb{R}^{|V| \times d}$.

O objetivo de cada uma dessas matrizes é prever a saída para uma posição relativa específica em relação a palavra que está no centro do contexto. A matriz O_{o-i} mais apropriada é selecionada na hora de realizar a predição

$p(w_o|w_i)$. O número de operações aritméticas desse modelo é igual ao do *word2vec*, pois a única diferença é a escolha da camada O para cada índice de palavras. A Figura 3.11 ilustra o modelo *skip-gram* do *wang2vec*.

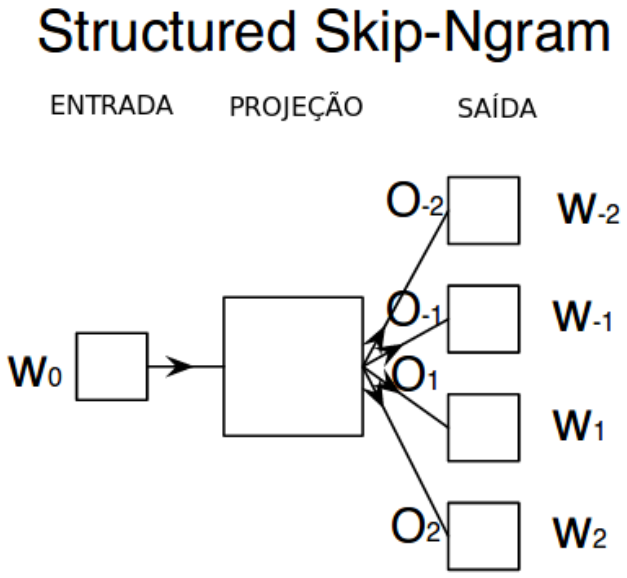

Figura 3.11: Exemplo do modelo *skip-gram*. Adaptado de Wang, 2015 (7).

4
Metodologia

Neste capítulo apresentamos a metodologia usada na construção do segmentador de textos. Nele detalhamos o conjunto de dados usado no treinamento e avaliação do modelo, a representação de textos usada e, por último, a arquitetura do segmentador.

4.1
Conjunto de Dados

Como já descrito no capítulo 2, o *dataset* usado neste trabalho é o mesmo concebido e empregado no trabalho de Ferreira (1). Ao todo, três conjuntos foram gerados a partir do corpus Bosque, sendo cada um correspondente a uma definição de *chunk* descrita no capítulo 2. Para esta geração, o autor preocupou-se em seguir a mesma metodologia aplicada na construção dos conjuntos de dados da CoNLL-2000. Cada conjunto apresenta-se em três colunas, sendo a primeira correspondente ao *token*, a segunda corresponde à etiqueta morfossintática e a última é a etiqueta correspondente ao tipo de *chunk* no formato IOB2. A Figura 4.1 ilustra uma sentença retirada do conjunto de treino.

```
O           ART     B-NP
segundo     ADJ     I-NP
dia         N       I-NP
terá        V       B-VP
a           ART     B-NP
economia    N       I-NP
como        PREP    O
tema        N       B-NP
comum       ADJ     B-ADJP
a           PREP    B-PP
as          ART     B-NP
três        NUM     I-NP
sessões     N       I-NP
.           .       O
```

Figura 4.1: Exemplo de uma sentença no formato IOB2 extraída do conjunto de dados.

Ao contrário da CoNLL-2000, onde somente os conjuntos de dados de treino e teste são disponibilizados, o *dataset* em português está dividido em três subconjuntos. O primeiro deles é o conjunto de treino, usado na etapa de aprendizado. O segundo é o conjunto de validação, que é usado no desenvolvimento do modelo e tem extrema importância: é a partir do conjunto de validação que os melhores hiperparâmetros do modelo são escolhidos, além disso permite a comparação entre diferentes arquiteturas. Por último, temos o conjunto de teste, que é usado apenas uma vez na coleta das estatísticas finais do modelo e estimação de seu poder de generalização em dados jamais vistos.

Cada subconjunto foi gerado de maneira aleatória, ou seja, é feito um embaralhamento de todas as sentenças do conjunto e, em seguida, ele é divido em três partes. A porção destinada ao treino é composta por 70% das sentenças do Bosque, 15% é destinada ao conjunto de validação e as 15% restantes são usadas como conjunto de teste.

Neste trabalho, avaliamos o desempenho da arquitetura proposta nos três *datasets* e os resultados obtidos são usados na comparação entre os modelos.

Por último, como forma de garantir resultados mais próximos aos obtidos em cenários reais, Ferreira não utilizou as etiquetas morfossintáticas presentes no corpus. Para isso, utilizou um etiquetador morfossintático (68) que apresentava resultados compatíveis com o estado-da-arte para gerar todos os conjuntos de dados.

4.2 Representação Textual

Como discutido no Capítulo 3, a escolha de uma boa representação textual pode contribuir consideravelmente para um bom desempenho do modelo. Neste trabalho, representamos o texto usando vetores de palavras pré-treinados gerados para a língua portuguesa. A geração desses *embeddings* se deu a partir de um grande corpus do português brasileiro e europeu, provenientes de fontes e gêneros variados, sendo resultado do trabalho de Hartmann et al. (69). Ao todo, dezessete corpus diferentes foram empregados, totalizando 1.395.926.282 *tokens* e quatro modelos produzidos. Todos esses modelos estão disponíveis gratuitamente para *download*[1].

Os vetores de palavras empregados nesta dissertação foram escolhidos a partir de diversos experimentos. Apesar de termos um conjunto de validação separado, optamos por juntá-lo com o conjunto de treino e realizamos uma validação cruzada. Em nossos experimentos, consideramos todos os quatro modelos pré-treinados e disponíveis, a saber: *word2vec*, *glove*, *FastText* e

[1]http://nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc

wang2vec. Os *embeddings* de melhor resultado na validação foram o modelo CBOW do *wang2vec* com dimensão de tamanho 100, que corrobora os resultados apresentados no trabalho original de Wang et al. (7) para tarefas que envolvem sintaxe.

Além dos vetores de palavras, adicionamos informações sobre a forma dos *tokens* que, ao todo, totalizam oito tipos. Esta estratégia é a mesma utilizada no trabalho de Huang et al. (8), onde os autores mostram que esses atributos são capazes de melhorar o desempenho de seu segmentador. Para isso, criamos um vetor binário de oito posições onde cada posição corresponde a um atributo de forma. A primeira posição indica se o *token* é do tipo numérico, a segunda indica se o *token* é predominantemente numérico, isto é, se a maioria dos caracteres que o compõe é composto por números; o próximo atributo determina se a palavra está totalmente em letras minúsculas e a próxima posição vai indicar se a palavra está totalmente em maiúsculo. As três últimas posições indicam se o *token* contém somente a primeira letra maiúscula, se contém um dígito e, por último, caso a palavra não se encaixe em nenhuma das situações descritas é considerada como *other*. Todas esses atributos de forma, então, são concatenados com os vetores de palavras e enviados para o segmentador.

4.3 Arquitetura da Rede Neural

4.3.1 BiLSTM-CRF

Em 2015, Huang et al. (8) propuseram um modelo baseado em uma *Long Short-Term Memory* bi-direcional para problemas de rotulamento de sequências com um CRF[2] na última camada. Segundo os autores, trata-se do primeiro trabalho a empregar uma BiLSTM-CRF para POS, NER e *Chunking*. Uma grande vantagem desse tipo de modelo é que redes LSTM possuem acesso a contextos passados e futuros (22), o que a torna efetiva para problemas de rotulamento de sequências, pois um rótulo pode depender do resultado do rótulo anterior. LSTM unidirecionais, por sua vez, possuem uma limitação, pois suas camadas escondidas recebem informação apenas do passado, como mostra a Figura 4.2.

[2]Conditional Random Fields

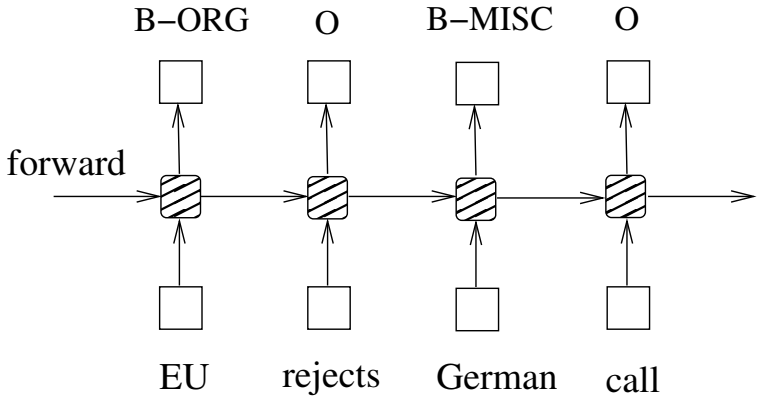

Figura 4.2: Rede LSTM unidirecional aplicada ao problema de NER. (8).

Como forma de mitigar essa limitação, o ideal é que as camadas escondidas tenham acesso a tanto informações de contexto do passado quanto do futuro, que é o caso das redes LSTM bidirecionais. Neste tipo de rede, as camadas escondidas têm acesso à informações de contextos históricos do passado e do futuro, como exemplificado na Figura 4.3.

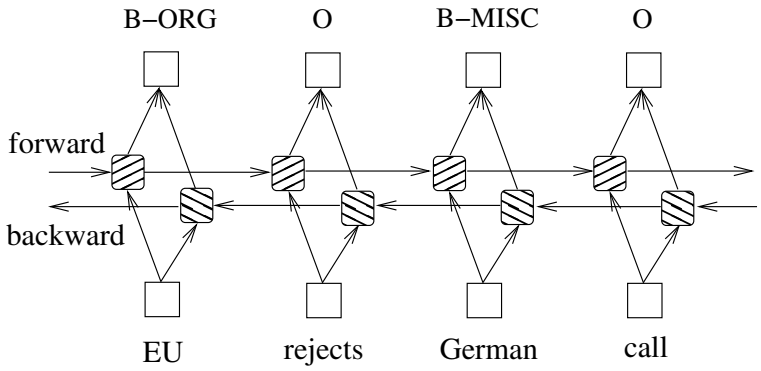

Figura 4.3: Rede BiLSTM aplicada ao problema de NER. (8).

Por último, a saída da rede é composta por um CRF. Problemas de predição estruturada[3], como é o caso do rotulamento de sequências, se beneficiam da correlação entre os rótulos vizinhos. Por exemplo, no problema de *chunking*, é muito mais provável que uma *tag* B-NP seja seguida de I-NP do que I-PP. O papel desta última camada, então, é gerar a sequência com maior probabilidade de ocorrência a partir das informações latentes obtidas pela rede LSTM.

4.3.2 BiGRU-CRF

Em 2016, Yang et al. (9) propuseram uma rede profunda hierárquica para as tarefas de POS, Segmentação de Texto e Reconhecimento de Entidades

[3] *structured prediction*

Nomeadas. O grande diferencial dessa arquitetura para as anteriores é o uso da camada BiGRU no lugar das BiLSTM e o uso de vetores densos de caracter. O objetivo desses vetores é capturar informações morfológicas e de contexto, o que torna o modelo mais robusto e agnóstico a linguagem. Nesse trabalho, os autores conseguiram bater o estado-da-arte nas três tarefas. A Figura 4.4 mostra a arquitetura em detalhes.

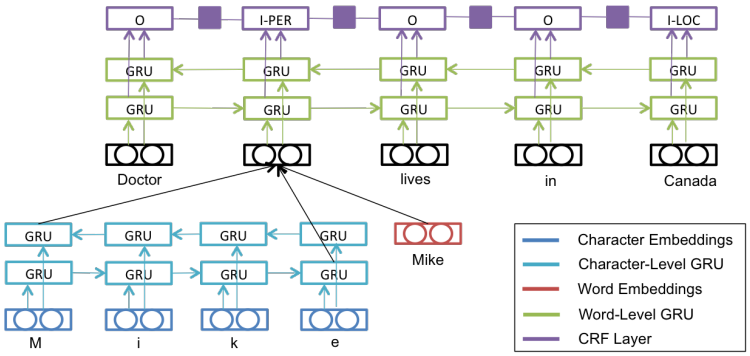

Figura 4.4: BiGRU-CRF (9).

4.3.3 BiLSTM-BiGRU-CRF

A arquitetura da rede proposta neste trabalho foi fortemente baseada nos trabalhos de Huang et al. (8) e Yang et al. (9). Neste último, os autores propuseram uma BiGRU-CRF com informação a nível de caracter concatenados com vetores de palavras como entrada da rede. Nossa proposta combinou essas duas arquiteturas resultando em uma rede neural LSTM bidirecional somada a uma rede GRU bidirecional com a camada final composta de um CRF. A Figura 4.5 ilustra a arquitetura implementada.

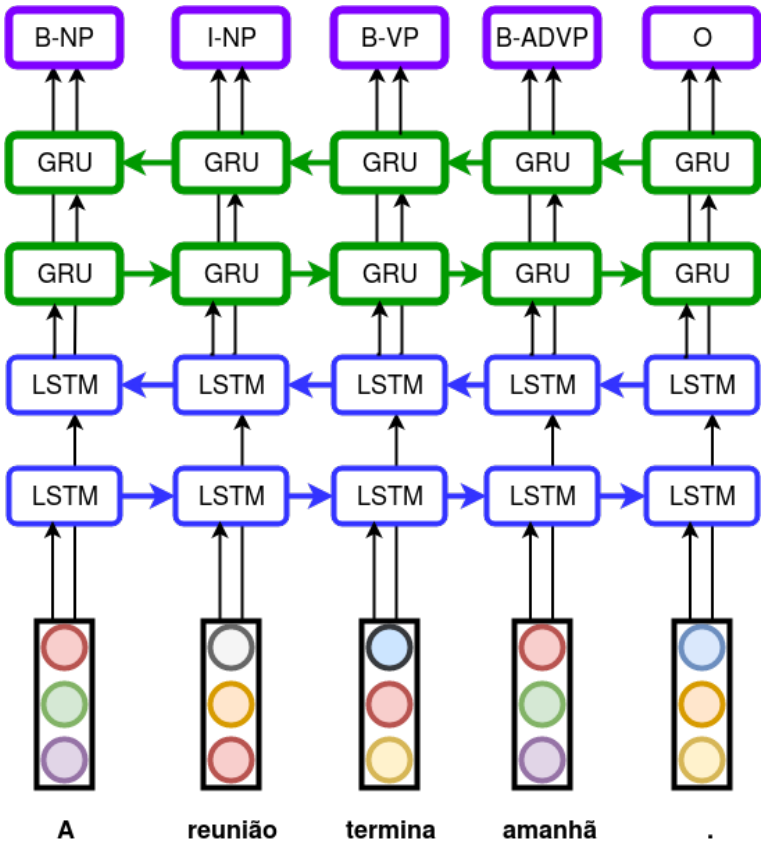

Figura 4.5: BiLSTM-BiGRU-CRF.

4.3.4 Análise de Dependências

A tarefa de Análise de Dependências tem como objetivo identificar as dependências entre palavras dentro de uma sentença (70). Mais precisamente, baseado em uma gramática de dependências, procura-se determinar que palavra governa cada palavra da sentença. A palavra que governa uma outra é chamada de pai e a palavra governada é chamada de dependente. Como uma das formas de avaliar a qualidade do nosso segmentador, implementamos um analisador de dependências semelhante ao proposto por Motta (70). A estratégia para esta análise é treinar o analisador sem informações de *chunk* e compará-lo com os resultados obtidos ao treinarmos com os atributos de *chunk* obtidos pelo nosso segmentador.

O *dataset* usado neste trabalho é o mesmo disponibilizado na competição CoNLL-X para o português, tendo diversos atributos como anotação morfossintática, atributos morfossintáticos adicionais, atributos de relação de dependência com o pai e outros.

5
Experimentos

Neste capítulo apresentamos os experimentos conduzidos e os resultados obtidos com o modelo proposto. Em seguida, os comparamos com o estado-da-arte para os três conjuntos de dados empregados nesta dissertação. Por último, apresentamos os resultados da aplicação dos atributos de *chunk* obtidos através do modelo proposto como atributo de entrada para o problema de análise de dependências.

Todos os experimentos foram executados em um servidor com a seguinte configuração: Intel(R) Core(TM) i7-5960X CPU @ 3.00GHz, 64GB RAM e uma GPU Nvidia Tesla K40c.

5.1
Arquitetura da Rede

Conforme brevemente discutido no Capítulo 4, o modelo proposto é uma rede BiLSTM-BiGRU-CRF, e foi desenvolvido com o auxílio da biblioteca de Aprendizado Profundo Keras (71). Keras é uma biblioteca de alto nível escrita em Python, que utiliza outras bibliotecas como TensorFlow, CNTK, ou Theano como *backend*. Isto é, ela provê uma interface uniforme que abstrai toda a complexidade inerente das bibliotecas de mais baixo nível que fazem parte de seu *backend*. Dessa forma, seu objetivo é proporcionar rapidez de experimentação sem abrir mão do desempenho.

A arquitetura da rede é composta de camadas encadeadas, sendo a primeira delas a camada de *word embeddings*. Essa camada é obtida através da concatenação entre os vetores pré-treinados do *wang2vec* com um vetor binário, que indica os atributos adicionais de forma, ou seja, se a palavra é um dígito, ou se é toda maiúscula e etc. Em seguida, temos a camada bidirecional LSTM, cada uma LSTM possui 100 neurônios na camada escondida, totalizando 200 neurônios. A próxima camada é uma bidirecional GRU, ou simplesmente, BiGRU; também totalizando 200 neurônios na camada escondida, 100 para cada GRU. A última camada da rede é um *Conditional Random Fields*, com dimensão equivalente ao número de classes a serem previstas. Como o *dataset* está no formato IOB2, para cada classe de sintagma tem-se dois rótulos. Por exemplo, o conjunto de dados referente à definição (NP, VP) possui 5 classes de

saída: B-NP, I-NP, B-VP, I-VP e O. Neste caso, a dimensão de saída do CRF será igual a 5. Já o *dataset* correspondente às definições (NP, VP, PP) e (NP, VP, PP, ADJP, ADVP) possuem um total de 7 e 11 classes, respectivamente.

Apesar da existência de um conjunto de validação já separado, optamos por combiná-lo com o conjunto de treino e realizamos uma validação cruzada *k-fold*. Todos os modelos comparados e hiperparâmetros da rede foram escolhidos através deste método. Essa técnica consiste em particionar o conjunto de dados de treinamento em 10 partes mutuamente exclusivas, onde 9 são usadas para treino e a última para validação. No total, são realizados 10 experimentos, cada um usando uma partição de validação diferente. Ao final, calculamos a média dos 10 experimentos. A Figura 5.1 descreve o processo de experimentação.

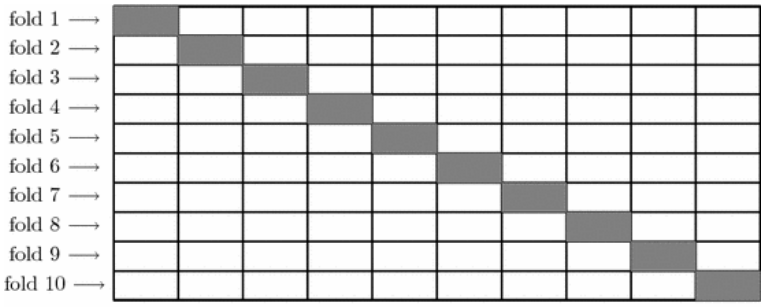

Figura 5.1: O processo de validação cruzada *10-fold* (10).

5.2 Métricas de Avaliação

As métricas de avaliação do segmentador de textos são as mesmas da CoNLL-2000. Para a competição foram estabelecidas três medidas: *precision*, *recall* e $F_{\beta=1}$. A métrica *precision*, ou precisão, é a porcentagem de sintagmas que foram identificados corretamente. A *recall*, ou sensibilidade, é a porcentagem de sintagmas presentes no conjunto de dados que foram identificados pelo segmentador. Por último, a métrica principal é o $F_{\beta=1}$, que corresponde a uma média harmônica entre a precisão e sensibilidade conforme a seguinte fórmula:

$$F_{\beta=1} = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Para garantir a confiabilidade da avaliação, todos os resultados foram obtidos através do *script* oficial da CoNLL-2000.

Já o problema de análise de dependências possui uma métrica diferente, que foi estabelecida para a competição CoNLL-X. Ela foca na identificação dos pais dos *tokens*, e é conhecida como UAS[1]. Essa métrica corresponde

[1] unlabeled attachment score

a porcentagem dos *tokens* que tiveram o pai previsto corretamente pelo classificador. Como trata-se de uma das tarefas da CoNLL-X, usamos o *script* oficial disponibilizado pelos organizadores.

5.3 Resultados

Nesta seção destacamos os resultados obtidos pelo segmentador nos conjuntos de dados em português e comparamos também os resultados no inglês. Por último, destacamos os resultados obtidos para a tarefa de análise de dependências com e seu atributos de *chunk*.

5.3.1 Segmentador de Texto - Português

Os resultados obtidos estão destacados nas Tabelas 5.1, 5.2 e 5.3 e correspondem às definições (NP, VP), (NP, VP, PP) e (NP, VP, PP, ADJP, ADVP), respectivamente. Como *baseline* usamos os mesmos valores descritos no trabalho de Ferreira (1). Também seguindo a linha do trabalho antecessor, optamos por incluir os resultados obtidos através da biblioteca OpenNLP (72) como comparação. O OpenNLP é uma biblioteca de código livre com diversos algoritmos de aprendizado de máquina para o processamento de linguagem natural. Dentre esses algoritmos está um segmentador de texto que utiliza métodos de máxima entropia. Os resultados do OpenNLP foram obtidos por Ferreira usando os parâmetros de configuração: número de iterações igual a 100 e valor de *cutoff* igual a 5.

Modelo	Precision (%)	Recall (%)	$F_{\beta=1}$
Baseline	54,14	54,79	54,47
OpenNLP	80,44	79,32	79,88
ETL (1)	84,50	77,99	81,11
BiLSTM-BiGRU-CRF (Este Trabalho)	86,90	87,25	**87,07**

Tabela 5.1: Desempenho do modelo para o conjunto de dados (**NP** e **VP**).

Modelo	Precision (%)	Recall (%)	$F_{\beta=1}$
Baseline	72,66	75,53	74,07
OpenNLP	86,19	85,66	85,92
ETL (1)	89,61	85,41	87,46
BiLSTM-BiGRU-CRF (Este Trabalho)	89,48	91,07	**90,27**

Tabela 5.2: Desempenho do modelo para o conjunto de dados (**NP**, **VP** e **PP**).

Modelo	Precision (%)	Recall (%)	$F_{\beta=1}$
Baseline	70,04	74,82	72,35
OpenNLP	86,83	86,67	86,75
ETL (1)	89,48	86,47	87,95
BiLSTM-BiGRU-CRF (Este Trabalho)	90,18	90,84	**90,51**

Tabela 5.3: Desempenho do modelo para o conjunto de dados (**NP**, **VP**, **PP**, **ADJP** e **ADVP**).

Os resultados mostram que a rede neural profunda proposta apresenta uma melhoria importante em comparação com técnicas mais clássicas, como o ETL e modelos de Máxima Entropia. Apesar de ser a definição mais difícil, nosso modelo apresentou um ganho de 5,96 pontos para o *dataset* (NP, VP) em relação ao trabalho anterior. O segundo maior ganho foi no conjunto de dados (NP, VP, PP), com um ganho de 2,81 pontos no $F_{\beta=1}$. Por último, para o conjunto de dados (NP, VP, PP, ADJP, ADVP), o modelo foi capaz de superar em 2,56 pontos o melhor resultado disponível na literatura para esse *dataset*. Dessa forma, fica ainda mais evidenciado o poder do Aprendizado Profundo para tarefas de linguística computacional, como a segmentação de texto em sintagmas.

Por último, destacaremos as métricas de desempenho para cada tipo de *chunk* de cada definição obtidas pelo *script* de avaliação da CoNLL-2000.

Tipo	Precision (%)	Recall (%)	$F_{\beta=1}$
NP	84,78	84,96	84,87
VP	94,45	95,46	94,95

Tabela 5.4: Desempenho do modelo por tipo de *chunk* para o conjunto de dados (**NP**, **VP**).

Tipo	Precision (%)	Recall (%)	$F_{\beta=1}$
NP	85,93	88,91	87,40
VP	94,83	95,37	95,10
PP	93,57	92,77	93,17

Tabela 5.5: Desempenho do modelo por tipo de *chunk* para o conjunto de dados (**NP**, **VP**, **PP**).

Estas estatísticas mostram que o modelo apresenta dificuldade em classificar os *chunks* do tipo NP. Dessa maneira, tais erros possuem impacto importante no resultado final do classificador pois os *chunks* do tipo NP são os mais numerosos, como mostra a Tabela 2.1.

Tipo	Precision (%)	Recall (%)	$F_{\beta=1}$
NP	87,77	89,02	88,39
VP	94,32	96,25	95,27
PP	94,91	92,86	93,88
ADJP	85,65	88,77	87,18
ADVP	84,90	84,99	84,95

Tabela 5.6: Desempenho do modelo por tipo de *chunk* para o conjunto de dados (**NP**, **VP**, **PP**, **ADJP**, **ADVP**).

5.3.2 Segmentador de Texto - Inglês

Nesta subseção destacamos os resultados do modelo para o inglês. A única diferença em relação ao modelo utilizado para o português se dá pela camada de *word embeddings*. Para isso usamos o modelo proposto por (73). Conforme destacado na Tabela 5.7, o modelo proposto neste trabalho apresentou um desempenho melhor que os modelos BiLSTM-CRF e BiGRU-CRF.

Modelo	$F_{\beta=1}$
SVM Classifier (35)	93,48
SVM Classifier (74)	93,91
Second order CRF (64)	94,30
HMM + voting scheme (75)	94,01
Conv network tagger (senna) (17)	94,32
BiLSTM-CRF (8)	94,46
BiGRU-CRF (9)	94,66
BiLSTM-BiGRU-CRF (Este Trabalho)	94,68
Pointer Network + LSTM Decoder (76)	94,72
Semi-supervised (77)	96,36
Contextual Embeddings (78)	**96,72**

Tabela 5.7: Desempenho do modelo para o conjunto de dados da CoNLL-2000.

5.3.3 Análise de Dependências

Como forma de atestar a qualidade do segmentador, realizamos também experimentos com a tarefa de análise de dependências, usando como atributos de entrada as informações de *chunk* obtidas pelo nosso segmentador. Para isso, treinamos um modelo sem os rótulos de *chunk*, e outros três com informações obtidas pelo nosso segmentador neural para cada definição de *chunk*. Esse analisador foi implementado por meio de uma Máquina de vetores de suporte

(SVM) e a modelagem baseou-se no trabalho de Crestana (29). A Tabela 5.8 descreve os resultados obtidos.

Modelo	UAS	Melhoria
Sem *Chunk*	87,39	-
(NP, VP)	**88,26**	**0,87**
(NP, VP, PP)	88,02	0,63
(NP, VP, PP, ADJP, ADVP)	87,84	0,45

Tabela 5.8: Impacto no desempenho na tarefa de Análise de Dependências ao adicionarmos o atributo de *chunk*.

Os resultados mostram que o uso de informações de *chunk* melhoram o desempenho desta tarefa independente dos tipos de sintagmas usados. Esses resultados destacam ainda mais a importância de um bom segmentador de texto.

6
Conclusão

O Processamento de Linguagem Natural é um campo de pesquisa que torna-se cada vez mais importante devido à grande quantidade de dados sendo produzidos diariamente. Sua importância é crucial para a facilitação da comunicação entre pessoas como, por exemplo, por meio de sistemas de tradução e agentes conversacionais. Apesar dos estudos iniciais compreenderem técnicas baseadas em regras e lógica matemática, pesquisas em PLN evoluíram para abordagens cada vez mais centradas em dados.

Em cenários onde tem-se abundância de dados, o uso de algoritmos de Aprendizado de Máquina torna-se bastante pertinente. Dessa maneira, técnicas de Aprendizado de Máquina surgiram como uma evolução natural no Processamento de Linguagem Natural. Apesar dos bons resultados, as limitações dos métodos tradicionais de AM impedem o avanço necessário para a adoção em larga escala de sistemas inteligentes. Parte dessas limitações, somente recentemente puderam ser solucionadas graças a técnicas mais sofisticadas, capazes de aprender estruturas intrínsecas a partir de dados de alta dimensão. Tais técnicas fazem parte do campo de pesquisa conhecido como Aprendizado Profundo, nome dado pois seus algoritmos são compostos de muitas camadas.

Apresentando grande sucesso na área de processamento de imagens, o Aprendizado Profundo não demorou muito a ser aplicado a problemas de linguística computacional. Uma das características mais interessantes é a possibilidade dos algoritmos aprenderem características a partir de dado não processado, ou seja, o modelo aprende atributos com pouca ou nenhuma interferência manual. Apesar dessa grande vantagem, certos problemas, em especial os linguísticos, ainda não podem ser solucionados somente a partir de dados não processados. Dessa maneira, o uso de certos atributos linguísticos são essenciais para o desempenho de tarefas mais complexas de PLN.

6.1
Problema

Em geral, a abordagem adotada na resolução de problemas complexos de PLN envolve o encadeamento de tarefas menores em sequência, cada uma agindo como uma etapa de pré-processamento. Cada tarefa contribui, até certo

grau, com informações importantes para a melhoria do desempenho final do problema alvo. Uma das tarefas é a segmentação de texto em sintagmas, cuja importância já foi comprovada em diversos problemas linguísticos. Isso se deve ao fato de que informações sintáticas parciais que são especialmente úteis como atributos de entrada para sistemas de Aprendizado de Máquina. Esse tipo de segmentação se propõe a dividir uma sentença em grupos não disjuntos e não recursivos de palavras sintaticamente correlacionadas.

O uso de algoritmos de Aprendizado de Máquina para o problema de segmentação de texto data do início dos anos 90, com foco na língua inglesa. Inicialmente interessadas em apenas um tipo de segmento, o sintagma nominal, as pesquisas evoluíram para abranger outros tipos de segmentos. Foi durante a CoNLL-2000 que a tarefa evoluiu para um estudo mais completo e, desde então, a segmentação de textos em inglês tornou-se um *benchmark* importante. Com poucas pesquisas no mesmo nível da realizada para o inglês, o estudo desta tarefa para o português limitava-se apenas aos sintagmas mais básicos, como o nominal e o verbal. Até onde sabemos, o trabalho mais próximo ao da CoNLL-2000 é resultado da dissertação de Ferreira (1).

Assim como os demais trabalhos contemporâneos à sua época, Ferreira usou técnicas de Aprendizado de Máquina convencionais para criar um segmentador automático. Até onde sabemos, nenhum outro estudo se propôs a aplicar algoritmos de Aprendizado Profundo para resolver o problema de *chunking* em português.

6.2 Proposta

Neste trabalho, propomos uma rede neural profunda para o problema de segmentação de texto em português. A arquitetura proposta foi baseada em trabalhos similares para o idioma inglês. Como forma de comparação, reportamos os resultados de trabalhos anteriores para os mesmos conjuntos de dados, além disso, comparamos os resultados com o obtido pela ferramenta OpenNLP, que dispõe de segmentador baseado em técnicas de Entropia Máxima. Por último, apresentamos os resultados também para a língua inglesa.

6.3 Contribuições

Com um desempenho muito superior aos trabalhos anteriores, os resultados indicam que a abordagem por Aprendizado Profundo é o caminho a ser seguido para a resolução dessa tarefa. Além dos resultados obtidos tratando a segmentação de texto como tarefa fim, avaliamos seu uso como atributo para

uma tarefa mais complexa. Em nossos experimentos, comprovamos a melhoria no desempenho da tarefa de análise de dependências ao usar informações de *chunk* obtidas pelo nosso segmentador.

6.4 Trabalhos Futuros

Como forma de melhorar ainda mais os resultados desta pesquisa, alguns aperfeiçoamentos podem ser efetuados. O uso de redes recorrentes como as LSTM e GRU foi o responsável pelo estado-da-arte em diversas tarefas de classificação de sequências recentemente. Todavia, tais técnicas possuem algumas limitações, não somente em relação ao desempenho computacional, que as fazem serem preteridas à outras arquiteturas, como as Redes Convolutivas. Além do melhor desempenho computacional, pesquisas recentes indicam que o uso de mecanismos como *self-attention* (79) já são o estado-da-arte em vários problemas de PLN.

Por fim, indicamos como trabalhos futuros para a segmentação de texto em português o uso de arquiteturas mais modernas, como as Redes Convolutivas e mecanismos de atenção. Outra direção é a investigação de outras variantes de vetores de palavras que não foram explorados neste trabalho para o português, como o Bert (80) e ELMo (81). Além disso, sugerimos a avaliação da aplicação de segmentos textuais em outras tarefas complexas de PLN.

7 Referências bibliográficas

[1] HARWOOD, P.. **A machine learning approach for portuguese text chunking**. Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro, 6 2011.

[2] LIFE, C.. **How Neural Networks Work**, 2018 (accessed October 31, 2018). https://chatbotslife.com/how-neural-networks-work-ff4c7ad371f7.

[3] **Understanding lstm networks**. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Último acesso em 31/12/2018.

[4] KARPATHY, A.. **Connecting Images and Natural Language**. PhD thesis, Stanford University, 8 2016.

[5] GRAVES, A.. **Supervised Sequence Labelling with Recurrent Neural Networks**, volumen 385 de **Studies in Computational Intelligence**. Springer, 2012.

[6] MIKOLOV, T.; CHEN, K.; CORRADO, G. ; DEAN, J.. **Efficient estimation of word representations in vector space**. CoRR, abs/1301.3781, 2013.

[7] LING, W.; DYER, C.; BLACK, A. W. ; TRANCOSO, I.. **Two/too simple adaptations of word2vec for syntax problems**. In: NAACL HLT 2015, THE 2015 CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES, DENVER, COLORADO, USA, MAY 31 - JUNE 5, 2015, p. 1299–1304, 2015.

[8] HUANG, Z.; XU, W. ; YU, K.. **Bidirectional LSTM-CRF models for sequence tagging**, 2015.

[9] YANG, Z.; SALAKHUTDINOV, R. ; COHEN, W. W.. **Multi-task cross-lingual sequence tagging from scratch**. CoRR, abs/1603.06270, 2016.

[10] HALEY, M. R.. **K-fold cross validation performance comparisons of six naive portfolio selection rules: how naive can you be and**

still have successful out-of-sample portfolio performance? Annals of Finance, 13(3):341–353, Aug 2017.

[11] **The dawn of big data**. http://www-935.ibm.com/services/uk/en/attachments/pdf/IBM_BOA_Big_Data_General_WEB_v2.pdf. Último acesso em 19/11/2018.

[12] **Data never sleeps 6.0**. https://www.domo.com/assets/downloads/18_domo_data-never-sleeps-6+verticals.pdf. Último acesso em 19/11/2018.

[13] MANNING, C. D.; SCHÜTZE, H.. **Foundations of statistical natural language processing**. MIT Press, 2001.

[14] SIMON, P.. **Too Big to Ignore: The Business Case for Big Data**. Wiley Publishing, 1st edition, 2013.

[15] ALPAYDIN, E.. **Introduction to Machine Learning**. The MIT Press, 2014.

[16] LECUN, Y.; BENGIO, Y. ; HINTON, G. E.. **Deep learning**. Nature, 521(7553):436–444, 2015.

[17] COLLOBERT, R.; WESTON, J.; BOTTOU, L.; KARLEN, M.; KAVUKCUOGLU, K. ; KUKSA, P. P.. **Natural language processing (almost) from scratch**. Journal of Machine Learning Research, 12:2493–2537, 2011.

[18] SANG, E. F. T. K.; BUCHHOLZ, S.. **Introduction to the conll-2000 shared task chunking**. In: FOURTH CONFERENCE ON COMPUTATIONAL NATURAL LANGUAGE LEARNING, CONLL 2000, AND THE SECOND LEARNING LANGUAGE IN LOGIC WORKSHOP, LLL 2000, HELD IN COOPERATION WITH ICGI-2000, LISBON, PORTUGAL, SEPTEMBER 13-14, 2000, p. 127–132, 2000.

[19] ZHAI, C.. **Fast statistical parsing of noun phrases for document indexing**. In: 5TH APPLIED NATURAL LANGUAGE PROCESSING CONFERENCE, ANLP 1997, MARRIOTT HOTEL, WASHINGTON, USA, MARCH 31 - APRIL 3, 1997, p. 312–319, 1997.

[20] ABNEY, S. P.. **Parsing By Chunks**, p. 257–278. Springer Netherlands, Dordrecht, 1992.

[21] OTTER, D. W.; MEDINA, J. R. ; KALITA, J. K.. **A survey of the usages of deep learning in natural language processing**. CoRR, abs/1807.10854, 2018.

[22] LIN, B. Y.; XU, F. F.; LUO, Z. ; ZHU, K. Q.. **Multi-channel bilstm-crf model for emerging named entity recognition in social media**. In: PROCEEDINGS OF THE 3RD WORKSHOP ON NOISY USER-GENERATED TEXT, NUT@EMNLP 2017, COPENHAGEN, DENMARK, SEPTEMBER 7, 2017, p. 160–165, 2017.

[23] CHO, K.; VAN MERRIENBOER, B.; GÜLÇEHRE, Ç.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H. ; BENGIO, Y.. **Learning phrase representations using RNN encoder-decoder for statistical machine translation**. In: PROCEEDINGS OF THE 2014 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, EMNLP 2014, OCTOBER 25-29, 2014, DOHA, QATAR, A MEETING OF SIGDAT, A SPECIAL INTEREST GROUP OF THE ACL, p. 1724–1734, 2014.

[24] HINGU, D.; SHAH, D. ; UDMALE, S. S.. **Automatic text summarization of wikipedia articles**. In: 2015 INTERNATIONAL CONFERENCE ON COMMUNICATION, INFORMATION COMPUTING TECHNOLOGY (ICCICT), p. 1–4, Jan 2015.

[25] HAMMERTON, J.; OSBORNE, M.; ARMSTRONG, S. ; DAELEMANS, W.. **Introduction to special issue on machine learning approaches to shallow parsing**. Journal of Machine Learning Research, 2:551–558, 2002.

[26] COLLINS, M.. **A new statistical parser based on bigram lexical dependencies**. In: 34TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 24-27 JUNE 1996, UNIVERSITY OF CALIFORNIA, SANTA CRUZ, CALIFORNIA, USA, PROCEEDINGS., p. 184–191, 1996.

[27] BUCHHOLZ, S.; DAELEMANS, W.. **Complex answers: a case study using a WWW question answering system**. Natural Language Engineering, 7(4):301–323, 2001.

[28] FERNANDES, E. R.; DOS SANTOS, C. N. ; MILIDIÚ, R. L.. **A machine learning approach to portuguese clause identification**. In: COMPUTATIONAL PROCESSING OF THE PORTUGUESE LANGUAGE, 9TH INTERNATIONAL CONFERENCE, PROPOR 2010, PORTO ALEGRE, RS, BRAZIL, APRIL 27-30, 2010. PROCEEDINGS, p. 55–64, 2010.

[29] MILIDIU, R. L.; CRESTANA, C. E. M. ; D. SANTOS, C. N.. **A token classification approach to dependency parsing**. In: 2009 SEVENTH BRAZILIAN SYMPOSIUM IN INFORMATION AND HUMAN LANGUAGE TECHNOLOGY, p. 80–88, Sept 2009.

[30] LACROIX, O.. **Investigating np-chunking with universal dependencies for english**. In: PROCEEDINGS OF THE SECOND WORKSHOP ON UNIVERSAL DEPENDENCIES (UDW 2018), p. 85–90. Association for Computational Linguistics, 2018.

[31] ZEMAN, D.; POPEL, M.; STRAKA, M.; HAJIC, J.; NIVRE, J.; GINTER, F.; LUOTOLAHTI, J.; PYYSALO, S.; PETROV, S.; POTTHAST, M.; TYERS, F. M.; BADMAEVA, E.; GOKIRMAK, M.; NEDOLUZHKO, A.; CINKOVÁ, S.; JR., J. H.; HLAVÁCOVÁ, J.; KETTNEROVÁ, V.; URESOVÁ, Z.; KANERVA, J.; OJALA, S.; MISSILÄ, A.; MANNING, C. D.; SCHUSTER, S.; REDDY, S.; TAJI, D.; HABASH, N.; LEUNG, H.; DE MARNEFFE, M.; SANGUINETTI, M.; SIMI, M.; KANAYAMA, H.; DE PAIVA, V.; DROGANOVA, K.; ALONSO, H. M.; ÇÖLTEKIN, Ç.; SULUBACAK, U.; USZKOREIT, H.; MACKETANZ, V.; BURCHARDT, A.; HARRIS, K.; MARHEINECKE, K.; REHM, G.; KAYADELEN, T.; ATTIA, M.; EL-KAHKY, A.; YU, Z.; PITLER, E.; LERTPRADIT, S.; MANDL, M.; KIRCHNER, J.; ALCALDE, H. F.; STRNADOVÁ, J.; BANERJEE, E.; MANURUNG, R.; STELLA, A.; SHIMADA, A.; KWAK, S.; MENDONÇA, G.; LANDO, T.; NITISAROJ, R. ; LI, J.. **Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies**. In: PROCEEDINGS OF THE CONLL 2017 SHARED TASK: MULTILINGUAL PARSING FROM RAW TEXT TO UNIVERSAL DEPENDENCIES, VANCOUVER, CANADA, AUGUST 3-4, 2017, p. 1–19, 2017.

[32] RAMSHAW, L. A.; MARCUS, M.. **Text chunking using transformation-based learning**. In: THIRD WORKSHOP ON VERY LARGE CORPORA, VLC@ACL 1995, CAMBRIDGE, MASSACHUSETTS, USA, JUNE 30, 1995, 1995.

[33] BRILL, E.. **Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging**. Computational Linguistics, 21(4):543–565, 1995.

[34] MARCUS, M. P.; SANTORINI, B. ; MARCINKIEWICZ, M. A.. **Building a large annotated corpus of english: The penn treebank**. Computational Linguistics, 19(2):313–330, 1993.

[35] KUDOH, T.; MATSUMOTO, Y.. **Use of support vector learning for chunk identification**. In: FOURTH CONFERENCE ON COMPUTATIONAL NATURAL LANGUAGE LEARNING, CONLL 2000, AND THE SECOND LEARNING LANGUAGE IN LOGIC WORKSHOP, LLL 2000, HELD IN COOPERATION WITH ICGI-2000, LISBON, PORTUGAL, SEPTEMBER 13-14, 2000, p. 142–144, 2000.

[36] MCCALLUM, A.; FREITAG, D. ; PEREIRA, F. C. N.. **Maximum entropy markov models for information extraction and segmentation**. In: PROCEEDINGS OF THE SEVENTEENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML 2000), STANFORD UNIVERSITY, STANFORD, CA, USA, JUNE 29 - JULY 2, 2000, p. 591–598, 2000.

[37] LAFFERTY, J. D.; MCCALLUM, A. ; PEREIRA, F. C. N.. **Conditional random fields: Probabilistic models for segmenting and labeling sequence data**. In: PROCEEDINGS OF THE EIGHTEENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML 2001), WILLIAMS COLLEGE, WILLIAMSTOWN, MA, USA, JUNE 28 - JULY 1, 2001, p. 282–289, 2001.

[38] DOS SANTOS, C. N.. **Aprendizado de Máquina na identificação de sintagmas nominais: o caso do português brasileiro**. Instituto Militar de Engenharia, 2006.

[39] MILIDIÚ, R. L.; DOS SANTOS, C. N. ; DUARTE, J. C.. **Phrase chunking using entropy guided transformation learning**. In: ACL 2008, PROCEEDINGS OF THE 46TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, JUNE 15-20, 2008, COLUMBUS, OHIO, USA, p. 647–655, 2008.

[40] DE FREITAS, M. C.; UZEDA-GARRÃO, M.; OLIVEIRA, C.; DOS SANTOS, C. N. ; SILVEIRA, M. C.. **A anotaçao de um corpus para o aprendizado supervisionado de um modelo de sn**. In: PROCEEDINGS OF THE III TIL/XXV CONGRESSO DA SBC, 2005.

[41] DOS SANTOS, C. N.; MILIDIÚ, R. L.. **Entropy guided transformation learning**. In: FOUNDATIONS OF COMPUTATIONAL INTELLIGENCE, VOLUME 1: LEARNING AND APPROXIMATION, p. 159–184. Springer, 2009.

[42] FREITAS, C.; ROCHA, P. ; BICK, E.. **Floresta sintá(c)tica: Bigger, thicker and easier**. In: COMPUTATIONAL PROCESSING OF THE PORTUGUESE LANGUAGE, 8TH INTERNATIONAL CONFERENCE, PROPOR 2008, AVEIRO, PORTUGAL, SEPTEMBER 8-10, 2008, PROCEEDINGS, p. 216–219, 2008.

[43] AFONSO, S.; BICK, E.; HABER, R. ; SANTOS, D.. **Floresta sintá(c)tica: a treebank for portuguese**, 2002.

[44] BICK, E.. **The Parsing System"Palavras": Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework**. Aarhus Universitetsforlag, 2000.

[45] RATNAPARKHI, A.. **Maximum Entropy Models for Natural Language Ambiguity Resolution**. PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 1998. AAI9840230.

[46] SANG, E. F. T. K.; VEENSTRA, J.. **Representing text chunks**. In: EACL 1999, 9TH CONFERENCE OF THE EUROPEAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, JUNE 8-12, 1999, UNIVERSITY OF BERGEN, BERGEN, NORWAY, p. 173–179, 1999.

[47] MITCHELL, T. M.. **Machine Learning**. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[48] MCTEAR, M.; CALLEJAS, Z. ; GRIOL, D.. **The Conversational Interface: Talking to Smart Devices**. Springer Publishing Company, Incorporated, 1st edition, 2016.

[49] KUMAR, S.; GAO, X.; WELCH, I. ; MANSOORI, M.. **A machine learning based web spam filtering approach**. In: 30TH IEEE INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS, AINA 2016, CRANS-MONTANA, SWITZERLAND, 23-25 MARCH, 2016, p. 973–980, 2016.

[50] BRODÉN, B.; HAMMAR, M.; NILSSON, B. J. ; PARASCHAKIS, D.. **Bandit algorithms for e-commerce recommender systems: Extended abstract**. In: PROCEEDINGS OF THE ELEVENTH ACM CONFERENCE ON RECOMMENDER SYSTEMS, RecSys '17, p. 349–349, New York, NY, USA, 2017. ACM.

[51] BAHDANAU, D.; CHO, K. ; BENGIO, Y.. **Neural machine translation by jointly learning to align and translate**. CoRR, abs/1409.0473, 2014.

[52] JIANG, X.; PANG, Y.; LI, X.; PAN, J. ; XIE, Y.. **Deep neural networks with elastic rectified linear units for object recognition**. Neurocomputing, 275:1132–1139, 2018.

[53] MA, L.; LU, Z. ; LI, H.. **Learning to answer questions from image using convolutional neural network**. In: PROCEEDINGS OF THE THIRTIETH AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, FEBRUARY 12-17, 2016, PHOENIX, ARIZONA, USA., p. 3567–3573, 2016.

[54] SAPANKEVYCH, N. I.; SANKAR, R.. **Time series prediction using support vector machines: A survey**. IEEE Comp. Int. Mag., 4(2):24–38, 2009.

[55] BENGIO, Y.; COURVILLE, A. C. ; VINCENT, P.. **Representation learning: A review and new perspectives**. IEEE Trans. Pattern Anal. Mach. Intell., 35(8):1798–1828, 2013.

[56] GOODFELLOW, I. J.; BENGIO, Y. ; COURVILLE, A. C.. **Deep Learning**. Adaptive computation and machine learning. MIT Press, 2016.

[57] MCCULLOCH, W. S.; PITTS, W.. **A logical calculus of the ideas immanent in nervous activity**. The bulletin of mathematical biophysics, 5(4):115–133, Dec 1943.

[58] ELMAN, J. L.. **Finding structure in time**. Cognitive Science, 14(2):179–211, 1990.

[59] JORDAN, M. I.. **Artificial neural networks**. chapter Attractor Dynamics and Parallelism in a Connectionist Sequential Machine, p. 112–127. IEEE Press, Piscataway, NJ, USA, 1990.

[60] HOCHREITER, S.; SCHMIDHUBER, J.. **Long short-term memory**. Neural Computation, 9(8):1735–1780, 1997.

[61] CHUNG, J.; GÜLÇEHRE, Ç.; CHO, K. ; BENGIO, Y.. **Empirical evaluation of gated recurrent neural networks on sequence modeling**. CoRR, abs/1412.3555, 2014.

[62] BENGIO, Y.; SIMARD, P. Y. ; FRASCONI, P.. **Learning long-term dependencies with gradient descent is difficult**. IEEE Trans. Neural Networks, 5(2):157–166, 1994.

[63] GERS, F. A.; SCHMIDHUBER, J. ; CUMMINS, F. A.. **Learning to forget: Continual prediction with LSTM**. Neural Computation, 12(10):2451–2471, 2000.

[64] SHA, F.; PEREIRA, F. C. N.. **Shallow parsing with conditional random fields**. In: HUMAN LANGUAGE TECHNOLOGY CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, HLT-NAACL 2003, EDMONTON, CANADA, MAY 27 - JUNE 1, 2003, 2003.

[65] BENGIO, Y.; DUCHARME, R.; VINCENT, P. ; JANVIN, C.. **A neural probabilistic language model**. J. Mach. Learn. Res., 3:1137–1155, Mar. 2003.

[66] MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. S. ; DEAN, J.. **Distributed representations of words and phrases and their compositionality**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 26: 27TH ANNUAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS 2013. PROCEEDINGS OF A MEETING HELD DECEMBER 5-8, 2013, LAKE TAHOE, NEVADA, UNITED STATES., p. 3111–3119, 2013.

[67] MORIN, F.; BENGIO, Y.. **Hierarchical probabilistic neural network language model**. In: AISTATS, 2005.

[68] DOS SANTOS, C. N.; MILIDIÚ, R. L. ; RENTERÍA, R. P.. **Portuguese part-of-speech tagging using entropy guided transformation learning**. In: COMPUTATIONAL PROCESSING OF THE PORTUGUESE LANGUAGE, 8TH INTERNATIONAL CONFERENCE, PROPOR 2008, AVEIRO, PORTUGAL, SEPTEMBER 8-10, 2008, PROCEEDINGS, p. 143–152, 2008.

[69] HARTMANN, N.; FONSECA, E. R.; SHULBY, C.; TREVISO, M. V.; SILVA, J. ; ALUÍSIO, S. M.. **Portuguese word embeddings: Evaluating on word analogies and natural language tasks**. In: PROCEEDINGS OF THE 11TH BRAZILIAN SYMPOSIUM IN INFORMATION AND HUMAN LANGUAGE TECHNOLOGY, STIL 2017, UBERLÂNDIA, BRAZIL, OCTOBER 2-5, 2017, p. 122–131, 2017.

[70] MOTTA, E. N.. **Indução e Seleção Incrementais de Atributos no Aprendizado Supervisionado**. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro, PUC-Rio, 9 2014.

[71] CHOLLET, F.; OTHERS. **Keras**. `https://keras.io`, 2015.

[72] APACHE. **openNLP Natural Language Processing Library**, 2014. http://opennlp.apache.org/.

[73] LEVY, O.; GOLDBERG, Y.. **Dependency-based word embeddings**. In: PROCEEDINGS OF THE 52ND ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, ACL 2014, JUNE 22-27, 2014, BALTIMORE, MD, USA, VOLUME 2: SHORT PAPERS, p. 302–308, 2014.

[74] KUDO, T.; MATSUMOTO, Y.. **Chunking with support vector machines**. In: LANGUAGE TECHNOLOGIES 2001: THE SECOND MEETING OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, NAACL 2001, PITTSBURGH, PA, USA, JUNE 2-7, 2001, 2001.

[75] SHEN, H.; SARKAR, A.. **Voting between multiple data representations for text chunking**. In: ADVANCES IN ARTIFICIAL INTELLIGENCE, 18TH CONFERENCE OF THE CANADIAN SOCIETY FOR COMPUTATIONAL STUDIES OF INTELLIGENCE, CANADIAN AI 2005, VICTORIA, CANADA, MAY 9-11, 2005, PROCEEDINGS, p. 389–400, 2005.

[76] ZHAI, F.; POTDAR, S.; XIANG, B. ; ZHOU, B.. **Neural models for sequence chunking**. In: PROCEEDINGS OF THE THIRTY-FIRST AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, FEBRUARY 4-9, 2017, SAN FRANCISCO, CALIFORNIA, USA., p. 3365–3371, 2017.

[77] PETERS, M. E.; AMMAR, W.; BHAGAVATULA, C. ; POWER, R.. **Semi-supervised sequence tagging with bidirectional language models**. In: PROCEEDINGS OF THE 55TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, ACL 2017, VANCOUVER, CANADA, JULY 30 - AUGUST 4, VOLUME 1: LONG PAPERS, p. 1756–1765, 2017.

[78] AKBIK, A.; BLYTHE, D. ; VOLLGRAF, R.. **Contextual string embeddings for sequence labeling**. In: PROCEEDINGS OF THE 27TH INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS, p. 1638–1649. Association for Computational Linguistics, 2018.

[79] VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L. ; POLOSUKHIN, I.. **Attention is all you need**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 30: ANNUAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS 2017, 4-9 DECEMBER 2017, LONG BEACH, CA, USA, p. 6000–6010, 2017.

[80] DEVLIN, J.; CHANG, M.; LEE, K. ; TOUTANOVA, K.. **BERT: pre-training of deep bidirectional transformers for language understanding**. CoRR, abs/1810.04805, 2018.

[81] PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K. ; ZETTLEMOYER, L.. **Deep contextualized word representations**. In: PROCEEDINGS OF THE 2018 CONFERENCE OF THE NORTH

AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES, NAACL-HLT 2018, NEW ORLEANS, LOUISIANA, USA, JUNE 1-6, 2018, VOLUME 1 (LONG PAPERS), p. 2227–2237, 2018.