Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Reconhecimento de Entidades Nomeadas Aplicado em Prontuários Médicos

Thais Cristina Cardozo de Souza

Trabalho de Conclusão de Curso

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Reconhecimento de Entidades Nomeadas Aplicado em Prontuários Médicos

Thais Cristina Cardozo de Souza
Orientador: Márcio Luis Lanfredi Viola

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do título de Bacharel em Estatística.

São Carlos
Setembro de 2022

Federal University of São Carlos
Exact and Technology Sciences Center
Department of Statistics

Recognition of Named Entities Applied in Medical Records

Thais Cristina Cardozo de Souza
Advisor: Márcio Luis Lanfredi Viola

Bachelors dissertation submitted to the Department of Statistics, Federal University of São Carlos - DEs-UFSCar, in partial fulfillment of the requirements for the degree of Bachelor in Statistics.

São Carlos
September 2022

Thais Cristina Cardozo de Souza

Reconhecimento de Entidades Nomeadas Aplicado em Prontuários Médicos

Este exemplar corresponde à redação final do trabalho de conclusão de curso devidamente corrigido e defendido por Thais Cristina Cardozo de Souza e aprovado pela banca examinadora.

Aprovado em 15 de setembro de 2022

Banca Examinadora:

- Márcio Luis Lanfredi Viola (Orientador)
- Rafael Bassi Stern
- Francisco Antonio Rojas

Agradecimentos

A Deus e ao meu anjo da guarda pela minha vida, sorte e fé que foram a minha base e luz durante as diversidades encontradas ao longo do curso.

Agradeço aos meus pais, por serem extremamente fortes durante a minha jornada e nunca permitirem que eu me afastasse ou esquecesse do meu objetivo e da minha definição de sucesso. O amor deles me nutriu e seus ensinamentos me formaram como ser humano. Os vejo em tudo que faço e tudo que sou.

Ao Prof. Márcio Viola, por ter aceitado me orientar, e embarcado nessa jornada de conhecimento. Sem ele eu não conseguiria traduzir em palavras o que gostaria de estudar. O agradeço por ter me orientado de forma paciente, didática e criativa, permitindo que várias peças do imenso quebra-cabeças que propus fossem se encaixando de maneira natural. Que muitos tenham o prazer de estar próximo do conhecimento que o senhor possui.

A minha amiga e médica Maria R. Lima Souza, pois, sem ela muitos dos termos necessários para o entendimento deste trabalho seriam apenas palavras jogadas sem alma. Seu dom pela medicina e paixão ao exercer sua profissão são exemplos de postura para mim.

"Apesar de tudo, jantei bem e fui ao teatro."

(Machado de Assis - Dom Casmurro)

Resumo

Uma entidade nomeada é um objeto do mundo real, como uma pessoa (por exemplo, Alan Turing), um local (por exemplo, Londres) ou uma organização (por exemplo, UFSCar). O Reconhecimento de Entidade Nomeada (em inglês, NER) é uma subárea do Processamento de Linguagem Natural (PLN), cujo objetivo é identificar várias entidades nomeadas que aparecem no texto. Este Trabalho de Conclusão de Curso propõe-se a detalhar as etapas para o uso de Reconhecimento de Entidades Nomeadas. Para tal, tendo como base o artigo de Kundeti *et al.* (2016) e Honnibal (2020), esse método será estudado e demonstrado sua aplicabilidade, com o auxílio da linguagem *Python*, biblioteca *spaCy* e do modelo *Med*7, em prontuários reais anonimizados que contêm informações textuais sobre pacientes.

Palavras-chave: *Entidades nomeadas, Med7, Processamento de linguagem natural, Redes neurais, spaCy.*

Abstract

A named entity is a real-world object, such as a person (e.g. Alan Turing), a location (e.g. London), or an organization (e.g. UFSCar). Named Entity Recognition (NER) is a subarea of Natural Language Processing (NLP) whose purpose is to identify various named entities that appear in text. This work proposes to detail the steps for the use of Named Entity Recognition. To this end, based on the article by Kundeti *et al.* (2016) and Honnibal (2020), this method will be studied and its applicability demonstrated, with the help of the *Python* language, *spaCy* library and the *Med7* model, in real anonymized medical records that contain textual information about patients.

Keywords: *Med7, Named entities, Neural networks, Natural language processing, spaCy.*

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

O ano era 1950 quando o matemático, cientista da computação e pesquisador Alan M. Turing nos apresentou a questão "As máquinas podem pensar?"(Turing, 1950). Desta forma, considerando que se uma máquina pudesse fazer parte de uma conversa com um humano, ela seria rotulada como uma máquina "pensante".

Posteriormente surgiram diferentes aplicativos, como o *ELIZA*, em 1966, que foi o primeiro *chatbot* na área da saúde a imitar um psicoterapeuta usando correspondência de padrões e seleção de respostas (Hajjar, 2021). Avançando cronologicamente, há o computador *HAL 9000* de "2001: Uma Odisseia no Espaço", que nos proporciona um novo questionamento: A tecnologia acabará por produzir uma máquina que poderá se comunicar conosco?

Os questionamentos levantados no parágrafo anterior notabilizam a importância do método denominado Processamento de Linguagem Natural (PLN), cujo principal objetivo é fazer com que os computadores executem tarefas úteis que envolverão a própria linguagem humana. Neste caso, podemos citar a habilitação da comunicação entre o humano e a máquina, atuação na melhoria do diálogo entre os próprios humanos ou até mesmo a execução de processos úteis que se utilizem de textos ou falas (Jurasfsky e Martin, 2008).

O PLN é uma vertente da Inteligência Artificial (IA, do inglês, *Artificial Intelligence*), que está relacionado a diversas áreas do conhecimento, como a Ciência da Computação, Linguística, Lógica e Estatística. Neste trabalho, focaremos na abordagem estatística, que, inicialmente, visava fazer inferências para o campo da linguagem natural, consistindo na captura de dados gerados de acordo com alguma distribuição de probabilidades desconhecida e, então, obter inferências sobre essa distribuição (Manning e Schutze, 2008).

Conforme a tecnologia avançou, o PLN, sob a perspectiva estatística, transformou-se em um processo misto, fortemente focado no uso de redes neurais de aprendizado profundo (algoritmos com maior complexidade no número de camadas e unidades em uma única camada de processamento) para realizar inferências em tarefas específicas e para desenvolver sistemas de ponta a ponta (Goldberg, 2017).

As redes neurais constituem um modelo de computação, cuja estrutura em camadas se inspirou na organização de uma rede de neurônios no cérebro, conectados intra e entre camadas (Haykin e Engel, 2007). Essa ferramenta de aprendizado de máquina criou um paradigma na IA que permite o aprendizado diretamente a partir dos dados, e não por meio de recursos de programação explícita.

De acordo com Treméa (2021), de modo geral, os neurônios de uma rede neural são organizados em três camadas: camada de entrada, camadas intermediárias e camada de saída. As camadas de entrada são responsáveis por receber os dados a serem analisados de fontes externas. Estes dados podem variar desde vetores numéricos até *pixels* de uma imagem ou até mesmo arquivos de áudio e textos. Depois da camada de entrada, existem as camada intermediárias, também conhecidas como camadas ocultas. Estes neurônios recebem os dados da camada de entrada e realizam conexões e análises de acordo com os parâmetros da rede. Por fim, os neurônios da camada de saída são responsáveis por receber as informações processadas nas camadas ocultas e exportá-las de acordo com a finalidade do algoritmo. A Figura 1.1 mostra a representação de uma rede neural.

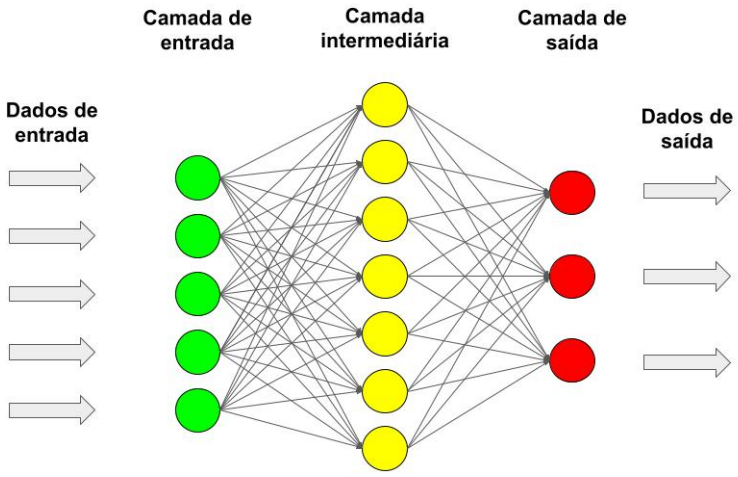

Figura 1.1: Representação de uma rede neural com uma camada intermediária.
Fonte: Retirado de Treméa (2021).

Dentre as arquiteturas mais utilizadas em redes neurais, há a Perceptron Multicamadas (MLP, do inglês, *Multilayer Perceptrons*) e Redes Neurais Convolucionais (CNN, do inglês, *Convolutional Neural Network*), que serão comentadas neste trabalho.

Tanto a MLP quanto a CNN podem ser usados para classificação de imagens ou textos. No entanto, a MLP utiliza um vetor como entrada enquanto a CNN recebe um tensor como entrada para entender a relação espacial entre *pixels* de imagens ou palavras. No Capítulo 2, os elementos presentes em ambas serão explicados e detalhados.

Alguns cenários, em que há a aplicação do Processamento de Linguagem Natural, correspondem ao uso de *chatbots* ou assistentes virtuais, presentes em ampla gama no mundo digital, inclusive no setor de saúde. Atualmente, nesse setor, esses assistentes podem capturar sintomas e fazer a triagem dos pacientes para o provedor mais adequado. Novas *startups* que formulam *chatbots* possuem sistemas que realizam o *Smart Exam*, o qual é um assistente médico virtual que utiliza a PNL conversacional para coletar dados pessoais de saúde e comparar as informações com diretrizes baseadas em evidências, juntamente com sugestões de diagnóstico para o provedor.

Além do exemplo supracitado, o Processamento de Linguagem Natural possui uma subárea de estudo no campo de extração de informação denominado Reconhecimento de Entidades Nomeadas (NER, do inglês *Named Entity Recognition*). Nela, buscamos identificar as entidades nomeadas de um texto, tais como nomes de pessoas, cidades e organizações, classificando-as em um conjunto pré-definido de categorias. Nesse contexto, os dados de entrada são um texto não-estruturado em um sistema de extração de entidades nomeadas, sendo este um texto em sua forma livre e sua saída é uma representação estruturada. Essa é considerada uma tarefa difícil, pois as entidades nomeadas constituem uma classe gramatical com muita variação lexical e de baixa frequência quando comparadas à massa total de dados textuais (Silva, 2020).

O NER, no domínio biomédico, pode envolver uma série de tarefas, como extração de proteínas, RNA, DNA de artigos científicos; identificação de sintomas, doenças, tratamentos, testes e problemas em textos clínicos; nome de fármacos ou nomes de medicamentos em textos biomédicos (Privatto, 2020).

Neste trabalho, o foco será o estudo de Reconhecimento de Entidades Nomeadas por meio de redes neurais, uma vez que este pode ser utilizado como uma proposta para

estruturação de Prontuários Médicos ou outros arquivos clínicos. Segundo Guilherme Hummel, Coordenador Científico da Hospitalar Hub, "Dados não-estruturados representam cerca de 80% de tudo o que é gerado pelos provedores de assistência médica. São "patinhos-feios" que não se encaixam em formatos específicos e nem observam regras semânticas determinísticas, o que torna mais difícil "encaixotá-los" nas práticas de *Health Analytics*".

1.1 Objetivos

Este trabalho tem como objetivo principal trazer ao leitor um estudo detalhado das etapas e modelos envolvendo o NER, utilizados na biblioteca de código aberto para processamento de linguagem natural em *Python*, $spaCy$, cujo desenvolvimento tem como foco ambientes de produção, possibilitando a criação de aplicativos que processem e entendam grandes volumes de texto. Sendo assim, este trabalho também apresenta como objetivo a introdução da abordagem estatística utilizada em um NER, detalhando a sua arquitetura e apresentando esse método como uma proposta de estruturação de dados no universo da saúde, proporcionando uma visão sobre a compreensão da qualidade e melhores resultados para pacientes e ensaios clínicos.

1.2 Organização

Essa monografia está organizada em quatro capítulos. No Capítulo 2 é exposto o referencial teórico relacionado ao Processamento de Linguagem Natural e os principais aspectos da estrutura de Reconhecimento de Entidades Nomeadas. O Capítulo 3 contém a descrição de cada etapa da implementação dos recursos propostos, a metodologia do trabalho, as métricas de avaliação utilizadas e resultados da etapa de pré-processamentos. O Capítulo 4, último capítulo deste Trabalho de Conclusão de Curso, apresenta as considerações finais.

Capítulo 2

Introdução ao Reconhecimento de Entidades Nomeadas

A estruturação de banco de dados de fontes textuais faz uso de Processamento de Linguagem Natural (PLN). Esta área utiliza conceitos de Inteligência Artificial e de Linguística para processar dados e automatizar tarefas que envolvam linguagem. Dentre as tarefas que são foco de estudo do PLN, pode-se citar Tradução de Máquina, Reconhecimento de Fala, Análise Automática de Discurso, Sumarização Automática de Textos, Extração de Informação, entre outras (Privatto, 2020).

Este trabalho tem como enfoque a tarefa de Extração de Informação (EI). No contexto da EI, há a tarefa chamada de Reconhecimento de Entidades Nomeadas (NER, do inglês *Named Entity Recognition*), cujo fundamento já fora tratado no Capítulo 1.

Este capítulo descreve os conceitos e ideias necessárias para a compreensão de todo o trabalho. Em relação ao PLN, o conceito de banco de dados e as principais técnicas de pré-processamento são abordadas. No contexto de NER, são apresentados a definição de sua tarefa; algumas formas de representação vetorial de palavras; as definições matemáticas dos modelos estatísticos de espaço contínuo que fundamentam o reconhecimento de uma entidade e o desenvolvimento da biblioteca que é utilizada para aplicar os conceitos aprendidos.

2.1 Intuição e conceitos: NER

Antes de inciar o referencial teórico necessário para entender o funcionamento de NER, será demonstrado, ludicamente, qual o seu propósito, algumas problemáticas e como este método pode auxiliar na extração de informações e estruturação de dados, fazendo com que estes sejam mais facilmente consultáveis e reutilizáveis.

Conforme citado no Capítulo 1, uma entidade nomeada pode ser uma palavra ou frase que identifica claramente um item de um conjunto de outros itens, os quais têm características semelhantes. A fim de exemplificar o reconhecimento de entidades, considere a seguinte frase: "Paciente X, 64 anos, funcionário público, chega ao Hospital Universitário Lauro Wanderley ($HULW$) com dispneia."[1] No API (do inglês, *Application Programming Interface*) de linguagem natural do *Google* [2], as seguintes classificações são obtidas: <Tipo=Pessoa **Paciente**> X, <Tipo=Número **64**> anos, <Tipo=Pessoa **funcionário público**>, chega ao <Tipo=Organização **Hospital Universitário Lauro Wanderley**> <Tipo=Organização **(HULW)**> com <Tipo=Outros **dispneia**>.

Conforme formalizado em Li *et al.* (2018) e exemplificado em Castro (2019) e Privatto (2020), a partir de uma sentença $s = \{p_1, p_2, p_3, ..., p_n\}$, em que n é o número de palavras em uma frase, o NER é responsável por encontrar e retornar uma lista (I_i, I_f, t). Cada elemento desta lista é uma entidade contida em s, em que I_i e I_f são, respectivamente, os índices iniciais e finais de uma entidade nomeada e t representa os tipos das entidades a partir de um conjunto de categorias predefinido.

O Exemplo 2.1 detalha os conceitos supracitados com $s =< p_1, p_2, p_3, ..., p_{21} >$ e a entidade "Pessoa", reconhecida nas palavras "funcionário público", por exemplo, será denotada por *<7, 8, Pessoa>*. Ou seja, a entidade nomeada "Pessoa" incia na palavra de índice 7 (p_7) e termina na palavra cujo índice é 8 (p_8) e esta palavra será classificada como entidade Pessoa.

[1] https://www.blogger.com/email-post.g?blogID=6676000049374062 16&postID=5730936420713653792

[2] https://cloud.google.com/natural-language/

Exemplo 2.1 (Exemplo da tarefa de reconhecimento de entidade nomeada.)

$$Paciente_{p1}\ X_{p2}\ ,_{p3}\ 64_{p4}\ anos_{p5}\ ,_{p6}\ funcionário_{p7}\ público_{p8}\ ,_{p9}\ chega_{p10}\ ao_{p11}\ Hospital_{p12}$$
$$Universitário_{p13}\ Lauro_{p14}\ Wanderley_{p15}\ (_{p16}\ HULW_{p17}\ )_{p18}\ com_{p19}\ dispneia_{p20}\ .{}_{p21}$$

$$\downarrow$$

Reconhecimento de entidade nomeada

$$\downarrow$$

$$<1,\ 1,\ Pessoa> \longrightarrow Paciente;$$

$$<4,\ 4,\ Número> \longrightarrow 64;$$

$$<7,\ 8,\ Pessoa> \longrightarrow funcionário\ público;$$

$$<12,\ 15,\ Organização> \longrightarrow Hospital\ Universitário\ Lauro\ Wanderley;$$

$$<17,\ 17,\ Organização> \longrightarrow HULW;$$

$$<20,\ 20,\ Outros> \longrightarrow dispneia.$$

A partir do Exemplo 2.1, percebe-se que o método procura realizar uma classificação sequencial de cada palavra p da sentença dada s. Primeiramente, pergunta-se se a palavra p é uma entidade e, caso seja, a que tipo de entidade t ela pertence. No Exemplo 2.1 foram reconhecidas seis entidades nomeadas.

Além dessa classificação de palavras, outra tarefa do modelo de NER que leva a alguns dos seus principais desafios é a delimitação correta das fronteiras das entidades. Considerando que no Exemplo 2.1 temos as delimitações de maneira correta para a entidade *Hospital Universitário Lauro Wanderley*. Um modelo de NER, que delimitasse de maneira diferente a que foi apresentada, poderia então reconhecer mais entidades do que o necessário ou ignorar partes da entidade, como demostra o Exemplo 2.2.

Exemplo 2.2 (Exemplos de delimitações incorretas.)

A) Com a entidade reconhecida de forma parcial:

$$<12,\ 13,\ Organização> \longrightarrow Hospital\ Universitário.$$

B) Com cada palavra sendo reconhecida como uma entidade diferente:

$$<12,\ 12,\ Organização> \longrightarrow Hospital;$$

$$<13,\ 13,\ Organização> \longrightarrow Universitário;$$

$$<14,\ 14,\ Organização> \longrightarrow Lauro;$$

$$<15,\ 15,\ Organização> \longrightarrow Wanderley.$$

Outro problema que um modelo de NER pode encontrar é em relação à ambiguidade das palavras, não apenas lendo o que está escrito e classificando, mas também compreendendo a declaração, sendo capaz de diferenciar, por exemplo, a palavra paciente, no sentido de ter paciência ("virtude"), e paciente, no sentido de pessoa que necessita de cuidados médicos.

Conforme Castro (2019), a tarefa de NER é aplicável não só em conteúdo textual de caráter geral, mas como também em acervos de domínios específicos, tais como Jornalismo, Biomedicina e Geologia. Quando se aplica extração de entidades em domínios específicos, tem-se a possibilidade de extrair diferentes tipos de informações, que são relevantes para os domínios em questão. No domínio Biomédico, há os exemplos comentados no Capítulo 1.

Após apresentar a intuição e os conceitos que norteiam um modelo de NER, a sua arquitetura é ilustrada pela Figura 2.1, cujas etapas serão tratadas nas seções e subseções seguintes.

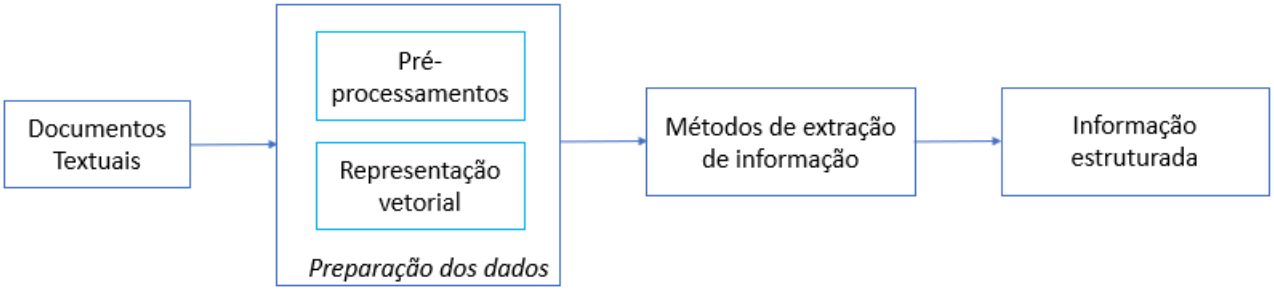

Figura 2.1: Arquitetura geral de modelo de NER.
Fonte: Adaptado de Privatto (2020).

2.2 Processamento de Linguagem Natural

Conforme visto na Figura 2.1, o primeiro passo para a modelagem do assunto proposto é a busca por um conjunto de documentos textuais do qual deseja-se extrair informações. A partir deste conjunto, realiza-se a preparação dos mesmos para que os métodos de extratores de informações possam agir (Privatto, 2020). Essas etapas, tradicionalmente, estão presentes em um processo de linguagem natural.

A seguir será tratado o conceito de banco de dados (*Córpus*) que, comumente, compõe processamentos de linguagem natural e, consequentemente, modelos de NER, assim como os principais pré-processamentos que são realizados quando realiza-se uma mineração de texto.

2.2.1 Córpus

Conforme visto no Exemplo 2.1, no API de linguagem natural utilizado na exemplificação, a palavra "dispneia" foi classificada na entidade "Outros", quando poderia ser melhor categorizada na entidade "Sintoma". Esse fato ocorre, pois, se o leitor tiver um problema ou objetivo específico que almeja abordar, faz-se necessário uma coleção de dados que suporte (ou pelo menos seja uma representação) o que deseja alcançar com aprendizado de máquina e PNL. Neste caso, o modelo deveria ser treinado em um banco de dados (*córpus*) que possuísse termos de domínio biomédico, por exemplo.

Segundo Sánchez *et al.* (1995), podemos caracterizar um *córpus* como sendo "um conjunto de dados linguísticos (pertencentes ao uso oral ou escrito da língua, ou a ambos), sistematizados segundo determinados critérios, suficientemente extensos em amplitude e profundidade, de maneira que sejam representativos da totalidade do uso linguístico ou de algum de seus âmbitos, dispostos de tal modo que possam ser processados por computador, com a finalidade de propiciar resultados vários e úteis para a descrição e análise".

Na escolha ou preparação do *córpus*, é necessário levar em consideração os tipos de informações que se quer obter para resolução de um determinado problema. Neste projeto, foi escolhido como objetivo aprofundar-se no modelo de Redes Neurais para NER por meio de aplicações no contexto médico. Portanto, o modelo será treinado com o *córpus MIMIC-III* (do inglês, *Medical Information Mart for Intensive Care III*), que é um dos maiores conjuntos de dados da língua inglesa disponíveis abertamente.

Desenvolvido pelo *MIT Lab* para *Computational Physiology*, o *MIMIC-III* inclui dados relacionados à saúde, desidentificados e associados a mais de quarenta mil pacientes que permaneceram em unidades de cuidados intensivos do *Beth Israel Deaconess Medical Center* entre 2001 e 2012. O banco de dados inclui informações como dados demográficos, medições de sinais vitais feitas à beira do leito ($\sim$ 1 ponto de dados por hora), resultados de exames laboratoriais, procedimentos, medicamentos, anotações do cuidador, relatórios de imagem e mortalidade (dentro e fora do hospital).

Com esse *córpus*, busca-se treinar o modelo de NER para que ele seja capaz de identificar alguns conceitos relacionados a medicamentos como: dosagem, nomes de medicamentos, duração, forma, frequência e via de administração.

2.2.2 Pré-processamentos

Esta subseção trata de uma das principais etapas para a construção de um modelo de aprendizado de máquina aplicado ao PLN. Costumeiramente, os dados coletados precisam ser preparados a fim de torná-los utilizáveis, melhorando a sua qualidade e organização. Esse processo deve ser realizado com atenção para que a limpeza dos dados não provoque perda de informação. Dados em forma de texto podem conter diversos ruídos, como emoções, pontuação, ambiguidades, palavras irrelevantes, entre outros. As técnicas que serão tratadas nessa seção são responsáveis por lidar com essa problemática.

Neste trabalho foi escolhido estudar e treinar um modelo para reconhecimento de entidades nomeadas por meio de Redes Neurais Convolucionais e Perceptron Multicamadas (tratadas a seguir). Segundo Camacho-Collados e Pilehvar (2017), realizar pré-processamento, quando lidamos com aprendizagem profunda em um domínio especifico (Biomédico, neste projeto), pode resultar em uma piora no funcionamento do modelo. Portanto, essa subseção objetiva apresentar algumas das principais técnicas utilizadas quando estamos no cenário geral de processamento de linguagem natural, trazendo ao leitor uma continuidade à fundamentação teórica proposta na Seção 2.2, porém, tais técnicas não serão incorporadas no modelo de estudo.

Tokenização

Conforme descrito na documentação da biblioteca *spaCy*, *Tokenização* é a tarefa de dividir um texto em segmentos significativos, chamados *tokens*. Há diversas definições

dadas pelos linguistas para um *token*. Neste trabalho usou-se a descrição de Maverick em "*Computacional Analysis of Present Day American English*" (Maverick, 1969) que define *token* como uma cadeia sequencial de caracteres alfanuméricos com espaços em ambos os lados. Em um *token*, podemos incluir hifens e apóstrofo, mas não sinais de pontuação.

Portanto, *tokenizar* um texto é segmentá-lo em palavras, pontuações e assim por diante. Esta técnica deve ser aplicada de acordo com regras específicas para cada idioma. Por exemplo, a pontuação no final de uma frase deve ser separada, enquanto "U.K." deve permanecer como um *token*.

Exemplo 2.3 (Exemplo de *tokenização* de uma sentença)

Ausência de tremores e movimentos involuntários

↓

Tokenização

↓

0	1	2	3	4	5
Ausência	de	tremores	e	movimentos	involuntários

Figura 2.2: Exemplo de *tokenização*.

Remoção de *stop words*

A técnica de remoção de *stop words* pode ser caraterizada pela exclusão de palavras que não possuem informações relevantes, portanto, não agregam valor ao texto. Como exemplo, podemos citar artigos, preposições, verbos auxiliares, pronomes e palavras que aparecem com alta frequência nos documentos. Estudos indicam que este processo tende a remover uma alta porcentagem de palavras do texto, porém, tal feito pode causar dificuldades ao processo de busca. A pesquisa por termos compostos exatos como "dor de cabeça", por exemplo, não traria resultados uma vez que a preposição "de" não consta mais no texto.

O mesmo tipo de problema pode ser encontrado ao realizar o técnica de de *stemming*: processo que analisa cada palavra individualmente e reduz à sua raiz (*stem*), por exemplo.

2.3 Representação de palavras

Uma das arquiteturas populares propostas para modelos de linguagem de redes neurais foi introduzida por Tomas Mikolov em *"Neural network based language models for highly inflective languages"*, no qual foram apresentados os vetores de palavras sendo primeiramente aprendidos usando uma rede neural com uma camada oculta. Esses vetores de palavras são, então, usados para treinar o modelo de linguagem. Dessa forma, os vetores de palavras são aprendidos mesmo sem a construção completa deste modelos de linguagem de redes neurais. Como objetivo, Mikolov procurou gerar vetores que contenham números, em que palavras similares, de acordo com seus contextos, estarão próximas no espaço (Silva, 2019).

Nesta seção aborda-se uma visão geral das principais características e métodos utilizadas na representação vetorial de palavras. Trata-se, também, como as mesmas estão conectadas à extração de informação por meio de modelos estatísticos. Para tal, devido à escrita altamente didática e completa, os artigos de Castro (2019), Privatto (2020) e Silva (2019), permearão todas as definições e conceitos que serão apresentados daqui em diante.

Ressalta-se, também, que, dados os objetivos presentes neste trabalho, os métodos e técnicas estudadas utilizam aprendizado profundo não supervisionado (quando o conhecimento é extraído automaticamente a partir do *córpus* de treinamento) baseado em *córpus*.

2.3.1 Word Embeddings: Glove

Na etapa de preparação dos dados, além da aplicação das técnicas de pré-processamento, também aplica-se métodos responsáveis por gerar a representação vetorial das palavras presentes nos textos. Dentre eles, pode-se citar os métodos *one-hot*, matriz de coocorrência e *word embeddings*.

Os métodos de *word embeddings* são descritos como uma técnica de aprendizado de máquina, que mapeia a busca de palavras em um espaço discreto de alta dimensão (a dimensão será a quantidade de palavras) para um vetor de número real em um espaço contínuo de baixa dimensão. A vantagem desta abordagem, em relação às demais supracitadas, está em conseguir representar as palavras por meio de vetores densos, cujas dimensões são mais representativas, capturando informações sintáticas e semânticas (Castro, 2019).

Esses métodos têm sido desenvolvidos utilizando diferentes técnicas baseadas em algoritmos de aprendizado de máquina, matrizes e grafos. Ao se fazer uso de *word embeddings*, a tarefa de NER pode ser tratada como uma tarefa de classificação, sendo, atualmente, bastante empregado o uso de classificadores que aprendem a categorizar as entidades presentes nos textos. Dentre as diversas técnicas existentes para *word embeddings*, será detalhado o método denominado *Glove*, dado que este método é o utilizado pela biblioteca $spaCy$.

Conforme introduzido por Pennington *et al.* (2014), *Glove* é um modelo de regressão log-bilinear global que obtém as representações vetoriais por meio de uma matriz de coocorrências entre palavras. Este modelo aproveita as informações estatísticas utilizando apenas os elementos diferentes de zero dessa matriz de palavras, em vez de usar toda a matriz esparsa ou janelas de contexto ao capturar dados complexos das relações linguísticas em um *córpus*. As estatísticas de ocorrências de palavras, calculadas considerando uma matriz de contagem de coocorrências em um *córpus*, são a principal fonte de informações disponíveis para os métodos não supervisionados aprenderem representações de palavras.

Usando as notações de Pennington *et al.* (2014), definimos a matriz de contagens de coocorrência palavra-palavra por X, cujo elemento X_{ij} representa o número de vezes que a palavra j ocorre no contexto da palavra i e $X_i = \sum_k X_{ik}$, o número de vezes que qualquer palavra aparece no contexto da palavra i. A partir destas quantidades, calcula-se $\hat{P}_{ij} = \hat{P}(j|i) = \frac{X_{ij}}{X_i}$, que é a probabilidade estimada da palavra j aparecer no contexto da palavra i.

Como forma de ilustrar este método, podemos tratar de um exemplo que mostra como certos aspectos do significado podem ser extraídos diretamente das probabilidades de coocorrência. Considere duas palavras i e j que exibem o aspecto particular de interesse, sendo i = *ice* (gelo) e j = *steam* (vapor). A relação dessas palavras pode ser examinada estudando a razão de suas probabilidades de coocorrência com várias palavras k, como mostra a tabela representada pela Figura 2.3.

Para as palavras (k) relacionadas a *ice*, mas não a *steam*, como k= *solid* (sólido), esperava-se um valor grande da razão $\frac{\hat{P}_{ik}}{\hat{P}_{jk}}$. De maneira similar, para palavras relacionadas a *steam*, mas não a *ice*, como k= *gas* (gás), deve-se ter um valor pequeno. Para palavras como *water* (água) ou $fashion$ (moda), que estão relacionadas tanto com *ice* e *steam*, ou

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figura 2.3: Probabilidades de coocorrência para palavras-alvo *ice* e *steam* com palavras de contexto.

Fonte: Retirado de Pennington *et al.* (2014).

com nenhuma delas, a proporção deve ser próxima a um. Todas essas expectativas foram confirmadas na Figura 2.3. Comparando as probabilidades, a relação é capaz de distinguir palavras relevantes (*solid* e *gas*) de palavras irrelevantes (*water* e $fashion$) e também é capaz de discriminar entre as duas palavras relevantes.

O argumento supracitado sugere que o ponto de partida apropriado para a aprendizagem de vetores de palavras deve ser com os índices de probabilidades de coocorrência em vez das próprias probabilidades. Observando que a relação $\frac{P_{ik}}{P_{jk}}$ depende de três palavras i, j e k, o modelo mais geral assume a forma

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}, \tag{2.4}$$

em que $w \in \mathbb{R}^d$ e $\tilde{w} \in \mathbb{R}^d$ representam, respectivamente. o vetor de palavras e o vetor de palavras de contexto separados.

Em uma avaliação apresentada por Pennington *et al.* (2014) e citada por Silva (2019), o $GloVe$ foi treinado em cinco *córpus* de tamanhos variados da língua inglesa: $Wikipedia$ 2010 com 1 bilhão de *tokens*; $Wikipedia$ 2014 com 1.6 bilhões de *tokens*; $Gigaword$ 5 com 4.3 bilhões de *tokens*; a combinação $Gigaword$ 5 + $Wikipedia$ 2014, com 6 bilhões de *tokens*; e em 42 bilhões de *tokens* de dados da Web. Após o treinamento, o $GloVe$ foi validado em uma tarefa de analogia de palavras e obteve uma acurácia de 75% , superando os demais métodos usados na comparação (*ivLBL, HPCA, SG, CBOW, vLBL, ivLBL, SVD, SVD-S, SVD-L* $CBOW^+$ *e* SG^+).

2.3.2 Redes Neurais Convolucionais

Segundo Jayan *et al.* (2013), existem diferentes abordagens de aprendizado supervisionado e não supervisionado para NER usando métodos estatísticos como HMM, *Decision*

Forest, Entropia máxima, SVM, etc. O termo Entidade Nomeada foi introduzido na sexta Conferência de Compreensão da Mensagem (do inglês, *Message Understanding Conference (MUC-6)*). Essas conferências do MUC contribuíram de forma decisiva para a pesquisa da área, fornecendo o *benchmark* para sistemas de entidades nomeadas.

Dentre os métodos supervisionados que lidam com NER, há as Redes Neurais Convolucionais (do inglês, *Convolutional Neural Network - CNN*). A motivação para o desenvolvimento dessas redes neurais teve início a partir dos estudos de $Hubel$ e $Wiesel$ acerca do córtex visual dos gatos, o qual é constituído por pequenas regiões de células que são sensíveis a áreas específicas no campo de visão. A excitação de determinadas células depende do formato e das orientações dos objetos vistos. A conexão entre as células ocorre por meio de uma arquitetura em camadas, que possibilita a construção da imagem com diferentes níveis de abstração (Almeida, 2020).

Devido à excelente escrita, representações visuais e didática do método que será descrito nesta subseção, os trabalhos de Castro (2019) e Privatto (2020) nortearam as definições e conceitos aqui apresentadas.

As $CNNs$, devido à sua característica de serem invariantes a distorções como rotação, translação e escala, foram, originalmente, utilizadas em problemas de reconhecimento de padrões e imagens. Porém, quando aplicadas no contexto de um PLN, podem ser usadas para determinar a representação vetorial de uma frase ou representação vetorial a partir dos caracteres que formam uma palavra.

A base do funcionamento dessas redes está em poder aprender diferentes níveis de características, observando os dados de entrada do modelo em mais de uma dimensão. Esse processo é chamado de Convolução e está ilustrado na Figura 2.4. Ele é realizado a partir de filtros convolucionais, que atuam da seguinte maneira:

1. Um filtro (na Figura 2.4 é a matriz 3×3, destacada na cor verde) de uma determinada dimensão percorre os dados de entrada (na Figura 2.4 são os dados originais, representados pela matriz 6×6), usando um passo [3] de um tamanho escolhido;

2. Para cada passo do filtro, é feita uma multiplicação dos dados de entrada, na mesma dimensão do filtro, pelos dados do próprio filtro;

[3] O tamanho do passo é o tamanho do deslocamento de um filtro, conforme ele se move no eixo horizontal da matriz de dados.

3. O valor resultante é armazenado em um mapa de características (na Figura 2.4 é a matriz de saída 4×4);

4. Opcionalmente, uma operação de amostragem (*Pooling* - Figura 2.5) é aplicada no mapa de características resultante. Esta operação pode ser de máximo, mínimo, média, ou alguma outra operação. O *pooling* também é aplicado em passos, observando os dados de uma dimensão determinada;

5. Os valores resultantes da operação de *pooling* são armazenados em outro mapa de características (na Figura 2.5, representada pela matriz 2×2).

Na Figura 2.4, tem-se o processo de convolução até a etapa em que é aplicado um filtro de dimensão 3×3 e passo de tamanho 1. Portanto, a figura representa o filtro (matriz 3×3, destacada na cor verde) sendo multiplicado por cada valor de uma matriz de tamanho igual (ou seja, 3×3) dos dados originais (representada pela cor laranja). Depois, esse filtro irá fazer essa mesma etapa na próxima matiz 3×3 (representada pela cor vermelho de borda tracejada). Após o filtro percorrer toda a matriz dos dados originais 6×6, fazendo essa multiplicação termo a termo, será gerada a matriz de valores representada pela matriz de saída 4×4.

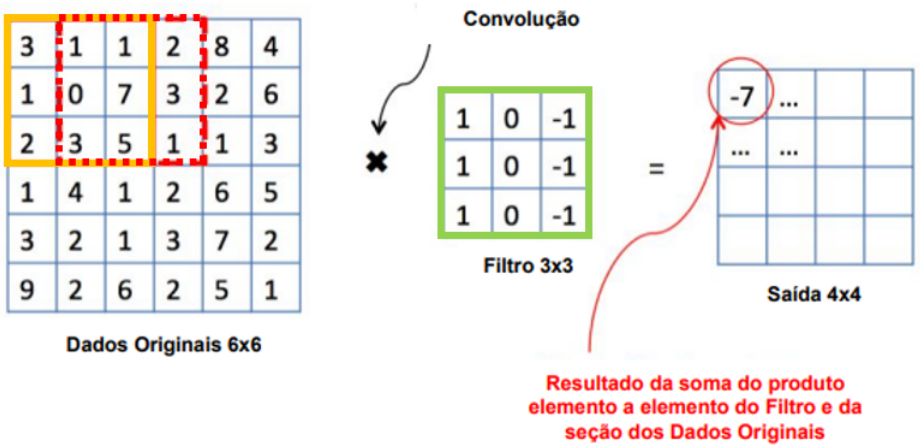

Figura 2.4: Exemplo de uma convolução.
Fonte: Adaptado de Castro (2019).

Em termos matemáticos, o que a Figura 2.4 representa é a seguinte equação:

$$(3 \cdot 1) + (1 \cdot 0) + (1 \cdot -1) + (1 \cdot 1) + (0 \cdot 0) + (7 \cdot -1) + (2 \cdot 1) + (3 \cdot 0) + (5 \cdot -1) = -7,$$

que será feita de maneira similar para toda a matriz de dados originais.

A Figura 2.5 mostra um exemplo de um processo de *pooling* usando a operação de máximo, observando uma janela de dados de dimensão 2×2 e passo de tamanho 2.

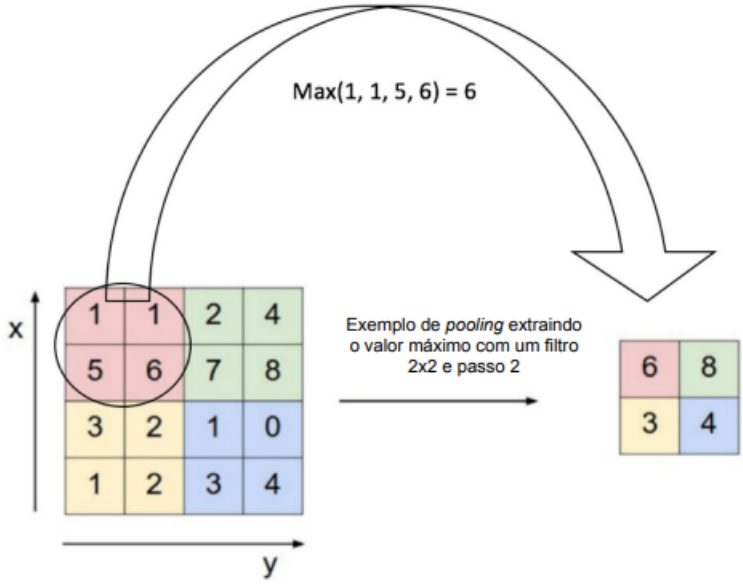

Figura 2.5: Exemplo de uma operação de *pooling*, usando a função de máximo.
Fonte: Retirado de Castro (2019).

Normalmente, são aplicados muitos filtros diferentes em uma mesma camada de dados, produzindo vários mapas de características diferentes, com o objetivo de aprender diferentes características. Quando a extração de informação é utilizada no Reconhecimento de Entidades Nomeadas, os extratores abordam o problema como um problema de classificação (Exemplo: identificar se uma palavra pertence ou não a alguma das entidade). Desse modo, os mapas de características são concatenados e redistribuídos em uma camada unidimensional ($flattening$) para que os neurônios dessa camada possam ser conectados a uma camada para classificação, que normalmente é uma função do tipo $softmax$ (função de ativação). Neste contexto, a camada $softmax$ aprende uma distribuição de probabilidade a partir dos resultados das convoluções, projetados na camada unidimensional.

Dado que o sentido de uma palavra é dado por seu contexto de ocorrência, o papel da CNN, em um PLN, é identificar as relações sintáticas e semânticas a nível de frase, alimentando uma rede neural capaz de captar dependências entre episódios e rótulos de uma sequência. Ao analisar um episódio de uma sequência, essas redes têm a capacidade de agregar características dos episódios anteriores à representação numérica do episódio atual.

A fim de entender o uso de uma CNN no contexto de PLN, a Figura 2.6 exemplifica o processamento da frase: "Técnicas aplicadas em diversos trabalhos na área de Processamento de Linguagem Natural".

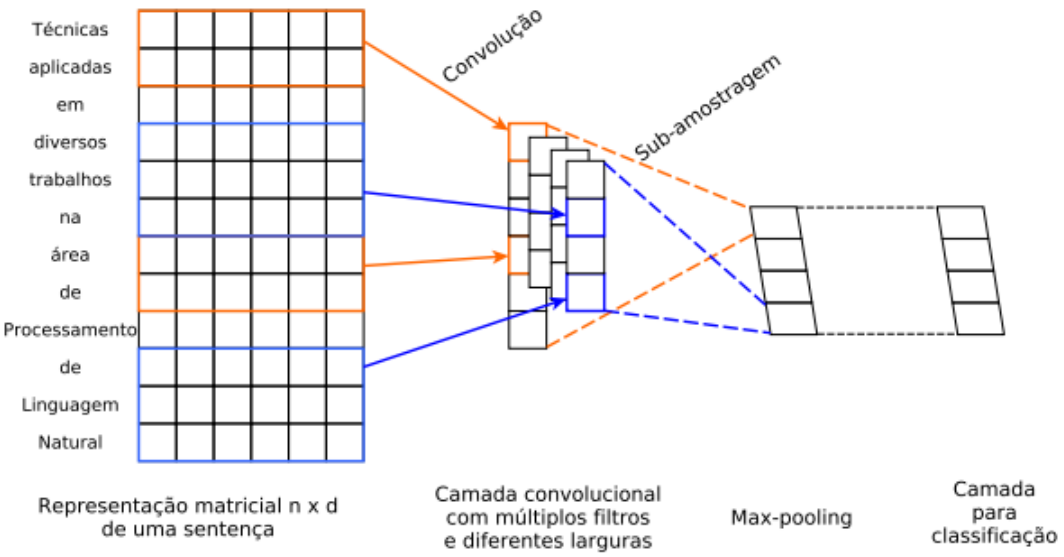

Figura 2.6: Exemplo de CNN aplicada à frases.
Fonte: Retirado de Privatto (2020).

Na Figura 2.6, por meio da junção dos *word embeddings* de cada uma das n palavras que compõem a frase exemplo, o primeiro passo é a criação da representação matricial $n \times d$, em que d é a dimensão dos *embeddings*.

Posteriormente, é realizado o processo de convolução, seguido do processo de subamostragem. A convolução tem como finalidade realizar a extração de características locais por meio de filtros de tamanho $g \times d$ (retângulos coloridos). Após a convolução, há uma subamostragem das características extraídas pela convolução, com o objetivo de escolher a característica mais relevante entre elas.

Conforme comentado, as etapas de convolução e subamostragem podem ser repetidas quantas vezes forem necessárias para o problema tratado. De maneira geral, são captadas características mais complexas ao se adicionar mais camadas à rede. Como última camada, é, frequentemente, utilizada uma função de ativação ($softmax$, por exemplo) para que a amostra possa, então, ser classificada. Ao se utilizar a CNN para somente criar uma representação vetorial, pode-se descartar a camada de classificação.

2.3.3 Perceptron Multicamadas

Para tratar o conceito de *Perceptron* Multicamadas no contexto de um PLN para NER, usou-se a descrição formalizada em Santos (2018).

Nela, Santos inicia descrevendo o método, que também pode ser chamado de redes neurais $feedforward$, como uma das arquiteturas principais do aprendizado profundo, cujo objetivo é aproximar uma função f^*. Considerando o contexto biomédico do trabalho, por hora, supomos que f^* será um classificador de tipos de sintomas. Dado um vetor x representando um sintoma, f^* irá mapear x na sua categoria de sintoma y, ou seja, $y = f^*(x)$.

Dessa forma, o *Perceptron* Multicamadas $f(x;\theta)$ aprenderá os parâmetros θ para aproximar o máximo possível da função f^*.

Costumeiramente, um *Perceptron* Multicamadas tem três tipos de camadas: camada de entrada, camada intermediária/oculta e camada de saída, sendo que ela pode apresentar mais de uma camada intermediária/oculta. A Figura 2.7 apresenta uma versão gráfica da arquitetura de uma Perceptron Multicamadas com 3 camadas.

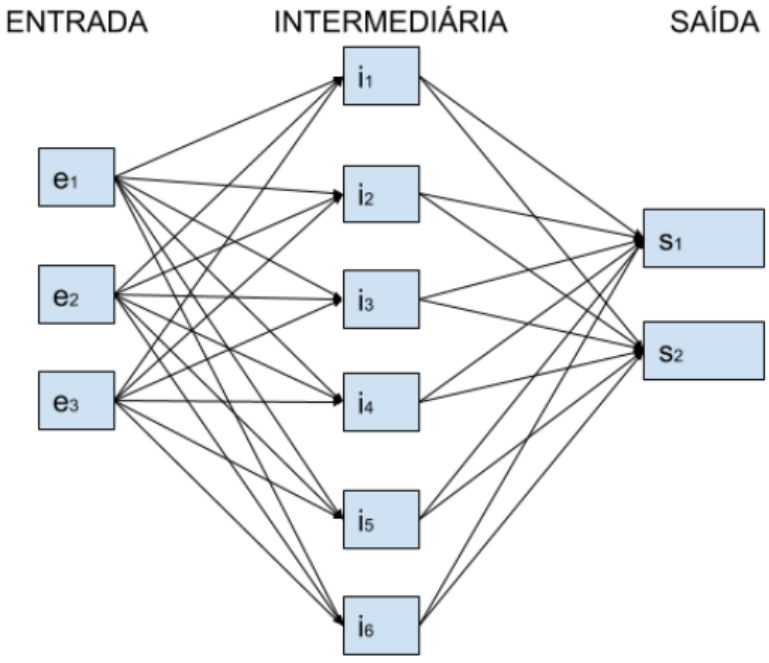

Figura 2.7: Arquitetura de uma Perceptron Multicamadas com 3 camadas.
Fonte: Extraído de Santos (2018).

Como citado, a Figura 2.7 é composta por 3 neurônios na camada de entrada, 6 neurônios na camada intermediária e 2 neurônios na camada de saída. A partir desta representação

gráfica, nota-se que cada neurônio i_x da camada intermediária recebe informação dos três neurônios da camada de entrada, assim como cada neurônio s_z da camada de saída recebe informação de todos os neurônios da camada intermediária.

Na Seção 2.4 será tratado da parte da estrutura da modelagem de NER em que a biblioteca *spaCy* utiliza o método de *Perceptron* Multicamadas.

2.4 Biblioteca spaCy

Devido à diversidade de abordagens que podem ser selecionadas para aplicar tanto um modelo PLN quanto um modelo NER, foi escolhido como foco estudar os alicerces e modelos estatísticos que compõem a biblioteca gratuita e de código aberto para PLN em Python, *spaCy*. Esta foi projetada para uso de produção a fim de auxiliar na construção de aplicações que lidam com grandes volumes de texto. Algumas de suas aplicações se dão na extração de informações, compreensão de linguagem natural e no pré-processamento de textos para aprendizagem profunda.

Até este ponto, tratou-se o fundamental teórico necessário para se compreender como aplicamos o reconhecimento de entidades nomeadas usando o aprendizado profundo de redes neurais. A partir de agora, será iniciado uma concatenação dos blocos de conhecimentos até aqui adquiridos.

Portanto, nesta seção, sabido que a biblioteca *spaCy* é a base prática de todo este projeto, propõe-se a entender como todos os métodos apresentados atuam como componentes dessa biblioteca, mostrando como eles são reunidos quando deseja-se treinar um modelo customizado, assim como, estudar algumas outras etapas exclusivas da *spaCy*. Para tal, será usada em exaustão o arquivo disponibilizado por um dos criadores da biblioteca, Matthew Honnibal, em "Embed, encode, attend, predict: The new deep learning formula for state-of-the-art NLP models"[4].

Nele, Honnibal sumariza que a abordagem pode ser resumida como uma fórmula simples de quatro passos: *embed* (incorporar), *encode* (codificar), *attend* (atender) e *predict* (prever). Além disso, enfatiza o conceito de generalidade, exemplificando que, ao se escrever um programa que preveja postagens abusivas de uma plataforma de mídia social, pode-

[4]Ressalta que este artigo foi escrito em 2016, portanto, por se tratar de uma ferramenta que sofre constantes atualizações algumas mudanças na arquitetura pode ter mudado desde então.

se generalizar o problema para que ele sempre pegue um texto e preveja um ID (sigla para *identity*), não importando se está sendo sinalizado postagens abusivas ou marcando *e-mails* que propõem reuniões. Se dois problemas receberem o mesmo tipo de entrada e produzirem o mesmo tipo de saída, é possível reutilizar o mesmo código do modelo e obter um comportamento diferente quando o associamos a dados diferentes. Para Honnibal, isto seria, metaforicamente, similar a jogar jogos diferentes que usem o mesmo mecanismo, reduzindo os problemas de PLN a problemas de aprendizado de máquina que recebem um ou mais textos como entrada.

Para atingir este pensamento, Honnibal propõe transformar esses textos em vetores para reutilizar soluções de aprendizado profundo de uso geral. O *spaCy* tem implementado sua própria biblioteca de aprendizado profundo, *thinc*, que é usada como base para diferentes modelos de PLN. Na maioria das tarefas gerais, o *spaCY* faz uso de redes neurais baseadas em CNN com alguns ajustes. Para o Reconhecimento de Entidade Nomeada, a estrutura dessas redes pode ser definida como uma abordagem baseada em transição emprestada de analisadores de redução de deslocamento, que pode ser melhor descrita nos seguintes passos:

- **Passo 1 - *Embed*:**

  Uma tabela de *embedding* mapeia vetores binários longos, esparsos em vetores mais curtos, densos e contínuos. Por exemplo, imagine um texto como uma sequência de caracteres $ASCII$ [5]. Existem 256 valores possíveis, então representa-se cada um dos valores como um vetor binário com 256 dimensões. O valor de *a* será um vetor de *0´s*, com o número 1 na coluna 97, enquanto o valor de *b* será um vetor de zeros com o número 1 na coluna 98. Esse procedimento é chamado de *word embeddings*, tratado na Subseção 2.3.1.

  Neste passo, a biblioteca possui algumas outras etapas, cuja principal é chamada de filtro *Bloom*. Uma estrutura de dados probabilísticos é usada para testar se um elemento é membro de um conjunto. Por exemplo, verificar a disponibilidade do nome de usuário, em que o conjunto é a lista de todos os nomes de usuário registrados.

  No cenário da biblioteca *spaCy*, essa técnica é responsável por compactar os vetores

[5] *inf sigla* de Código Padrão Americano para Intercâmbio de Informações, um esquema de codificação que atribui valores numéricos a caracteres visando padronizar a troca de dados entre computadores

de palavras, ou seja, reduzir o tamanho da tabela de *embedding*. Para fazer isso, este filtro utiliza um conceito chamado *hash*, que, por definição, é uma função que recebe uma entrada e emite um identificador exclusivo de comprimento fixo que é usado para identificação dessa entrada. Portanto, uma filtragem *Bloom* mantém apenas os *hashes* de palavras e usam esses como chaves no dicionário incorporado, em vez da própria palavra, criando um dicionário de *embedding* mais compacto, com palavras potencialmente colidindo e terminando com as mesmas representações vetoriais.

- **Passo 2 - *Encode*:**

  Dada uma sequência de vetores de palavras, a etapa de codificação (*encode*) calcula uma representação chamada de matriz de sentença, na qual cada linha representa o significado de cada *token* no contexto do restante da sentença. A tecnologia utilizada para este fim foi tratada na Subseção 2.3.2, cuja lista de palavras é codificada em uma matriz de frases, para levar o contexto em consideração por meio de uma CNN.

  Portanto, é nesse passo que o modelo pode aprender que a palavra paciente (que tem a virtude paciência) tem um significado diferente da palavras paciente (que precisa de cuidados médicos), mesmo que processadas em duas frases e em *tokens* separados. Uma grande fraqueza dos modelos de PNL que pode ter uma possível solução.

- **Passo 3 - *Attend*:**

  A etapa de *attend* reduz a representação da matriz produzida pela etapa *encode* para um único vetor, para que possa ser passada para uma rede $feedforward$ padrão para previsão. A vantagem característica de um mecanismo de *attend* sobre outras operações de redução é este mecanismo recebe como entrada um vetor de contexto auxiliar.

  Ao reduzir a matriz a um vetor, perde-se, necessariamente, informações. Por isso, o vetor de contexto é crucial: ele informa quais informações descartar para que o vetor sumário seja adaptado à rede que a consome. Pesquisas recentes mostraram que o mecanismo de atenção é uma técnica flexível e novas variações dele podem ser usadas para criar soluções elegantes e poderosas. Sendo assim, pode-se resumir este passo como sendo a decisão de quais partes são mais informativas em uma consulta

e obter as representações específicas do problema.

- **Passo 4 - *Predict*:**

  Uma vez que o texto ou conjunto de textos tenha sido reduzido em um único vetor, pode-se aprender a representação alvo (um rótulo de classe, um valor real, um vetor, etc). Também, pode-se fazer uma previsão estruturada usando a rede como controladora de uma máquina de estado, como um analisador baseado em transição.

  Este passo foi tratado na Subseção 2.3.3, pois, o *spaCy* usa a técnica de *Perceptron* Multicamadas para realizar esta inferência.

Por fim, ressalta-se que, em um modelo de NER, uma das métricas de avaliação mais usada é a proximidade da classificação da entidade com a escrita humana, ou seja, a avaliação intrínseca que normalmente consiste em comparar a saída do sistema para determinadas entradas com os resultados esperados (corretos), obtidos por meio da atribuição manual de sentidos em um corpus de referência (Silva, 2019). A biblioteca possui uma classe interna para avaliar modelos de NER, chamada *scorer*. O *scorer* usa a correspondência exata para avaliar o NER, ou seja, dado que a entidade deveria ser nomeada como "pessoa" se ela realmente foi classificada como tal, porém, será um *score* único para o modelo como um todo, não um score para cada classe que o modelo pode reconhecer.

Desta função, pode-se ter o valor de precisão, *recall* e F1, descritas a seguir. Para entender melhor cada métrica, primeiramente, é necessário definir alguns conceitos.

- **Verdadeiros Positivos (VP):** classificação correta da classe Positivo;

- **Falsos Negativos (FN - Erro Tipo II):** erro em que o modelo previu a classe Negativo quando o valor real era classe Positivo;

- **Falsos Positivos (FP - Erro Tipo I):** erro em que o modelo previu a classe Positivo quando o valor real era classe Negativo;

- **Verdadeiros Negativos (VN):** classificação correta da classe Negativo.

Sendo assim, define-se as métricas citadas anteriormente:

- **Precisão (*precision*):** dentre todas as classificações de classe Positivo que o mo-

delo fez, representa quantas estão corretas. É calculada pela expressão

$$\frac{VP}{VP + FP}.$$

- ***Recall* / Revocação / Sensibilidade:** dentre todas as situações de classe Positivo como valor esperado, representa quantas estão corretas. É calculada por

$$\frac{VP}{VP + FN}.$$

- **F1-Score**: É a média harmônica entre precisão e *recall*, dada por

$$2 \cdot \frac{precisão \cdot recall}{precisão + recall}.$$

Capítulo 3

NER aplicado a prontuários médicos

Neste capítulo, são postos em execução os objetivos traçados no Capítulo 1. Reitera-se que o objetivo deste seção é mostrar, de forma prática, o uso da biblioteca *spaCy* para processamento de linguagem natural na tarefa do reconhecimento de entidades nomeadas no cenário de prontuários médicos da língua inglesa. Busca-se portanto, verificar como realizar este procedimento na linguagem *Python* e visualizar as etapas deste método na mesma.

A tarefa do *NER* necessita, em resumo, de dois tipos de modelos que atuam em conjunto. O primeiro é utilizado como uma espécie de dicionário e também contém a função de destacar quais palavras, provavelmente, possuem o interesse de serem reconhecidas. Neste trabalho, esta função é realizada pela biblioteca *spaCy*. Considere, por exemplo, que seja destacada as entidades "de 8h em 8h, por cinco dias".

O segundo modelo é responsável por captar essas palavras destacadas e classificá-las nas classes já predefinidas pelo usuário. Neste estudo, esta etapa será realizada pelo pacote *Med7*, que será descrito nas próximas seções. No exemplo contido no parágrafo anterior, os *tokens* seriam classificados como "frequência" (8h em 8h) e "duração" (cinco dias), por exemplo. Ressalta-se que, apesar do *spaCy* ser capaz de realizar os dois processamentos, por se tratar de um contexto específico da saúde, esta biblioteca não possui um bom desempenho já que foi treinada para tarefas *NER* gerais, possuindo o reconhecimento de entidades mais comuns como pessoas, datas, lugares, entre outros, justificando, assim, a necessidade de um modelo pré-treinado para o universo em estudo.

Dito isso, o desenvolvimento deste capítulo será norteado por dois exemplos:

- O primeiro permitirá que o leitor compreenda como os diversos modelos já pré-treinados, contidos na biblioteca *spaCy*, funcionam. Para isso, um modelo será criado e treinado em uma unidade de prontuário médico deste o início, em que será informado a ele quais entidades reconhecer e quais serão suas nomeações. Assim, pretende-se expor e conectar cada etapa do treinamento ao embasamento teórico discorrido nos capítulos anteriores. Para demostrar que o modelo fora, de fato, treinado, os resultados (entidades nomeadas) serão aplicados em um outro prontuário diferente do treino;

- Posteriormente, no mesmo prontuário utilizado para teste, será aplicado o $Med7$ a um modelo treinado com o *córpus* descrito na Subseção 2.2.1 e com o dicionário contido na *spaCy*. Nesta segunda etapa, busca-se demonstrar como este modelo e técnica podem auxiliar na estruturação de prontuários nos dias de hoje. Salienta-se que por conta do sigilo dos dados, além do esforço necessário para que se treine e obtenha um modelo com desempenho satisfatório para o contexto, optou-se por demonstrar a aplicabilidade da biblioteca com um pacote que já a utiliza, o $Med7$.

Todas as análises foram realizadas via *Colaboratory* ou *Colab*, que é um produto do *Google Research* (área de pesquisas científicas do *Google*). O *Colab* permite que qualquer pessoa escreva e execute um código *Python* arbitrário pelo navegador e é, especialmente, adequado para aprendizado de máquina, análise de dados e educação. Tecnicamente, o *Colab* é um serviço de notebooks hospedados do *Jupyter* que não requer nenhuma configuração para uso e oferece acesso sem custo financeiro a recursos de computação como *GPUs*. Para o estudo serão utilizadas as versões 3.4 e 3.7 da biblioteca *spaCy* e do *Python*, respectivamente.

3.1 Exemplo ilustrativo

Nesta seção, serão ilustradas as etapas que o *spaCy* realiza para uma tarefa de *NER*. Ressalta-se que o exemplo será desenvolvido com prontuários médicos na língua inglesa, aleatoriamente retirados da internet, uma vez que o banco de dados *MIMIC-III* contém restrições ao acesso das anotações médicas.

Conforme explicado na documentação da biblioteca e abordado neste trabalho, muitos componentes do *spaCy* são alimentados por modelos estatísticos. Cada decisão que esses

componentes tomam (por exemplo, se uma palavra é uma entidade nomeada) é uma previsão baseada nos valores de peso atuais do modelo. Os valores de peso são estimados com base em exemplos do modelo que foi visto durante o treinamento. Para treinar um modelo, primeiramente são necessários dados de treinamento (exemplos de texto e os rótulos que o modelo deve prever serão descritos na Subseção 3.1.1).

O treinamento é um processo iterativo no qual as previsões do modelo são comparadas com os rótulos de referência para estimar o gradiente da perda, indicando a direção de máxima perda. O gradiente da perda é, então, usado para calcular o gradiente dos pesos por meio da "retropropagação" (em inglês, *backpropagation*). Os gradientes indicam o quanto os valores de peso devem ser alterados para que as previsões do modelo se tornem mais semelhantes aos rótulos de referência ao longo do tempo (Honnibal *et al.*, 2020). A Figura 3.1 esquematiza essa metodologia, na qual a entidade refere-se ao texto de entrada para o qual o modelo deve prever um rótulo (classe).

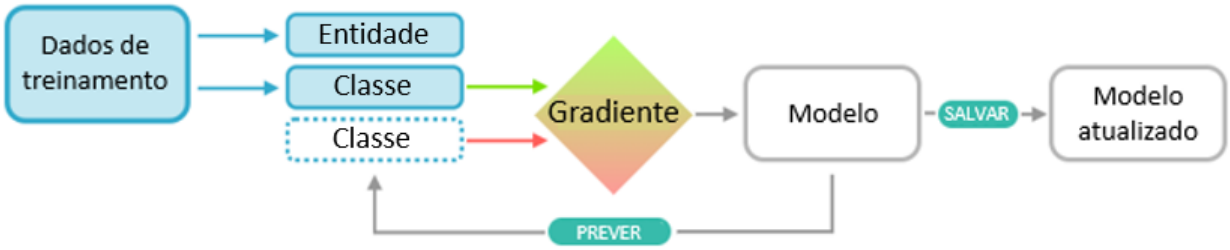

Figura 3.1: *Workflow* de treinamento de um novo modelo de *NER* - biblioteca *spaCy*.
Fonte: Traduzido de Honnibal *et al.* (2020).

De modo sucinto, para treinar um modelo que realize a tarefa NER no $spaCy$, são necessários cinco passos: preparação dos dados de treinamento, exemplos e seus rótulos; conversão dos dados para o formato *.spacy*; criação do arquivo que conterá as configurações necessárias para treinar o modelo; preenchimento do arquivo de configuração com os parâmetros necessários e execução do treinamento. Estas etapas serão descritas a seguir.

3.1.1 Preparação dos dados

Para treinar um modelo de reconhecimento de entidade nomeada personalizado, são necessários dados de texto relevantes com as anotações adequadas. Para o propósito desta subseção, será utilizado o primeiro prontuário médico do Apêndice A (Prontuário 1 - Treino). Neste caso, a primeira etapa da preparação dos dados se dá por definir em um texto quais entidades serão reconhecidas e o rótulo que as mesmas receberão. Para

este tipo de tarefa, foi utilizada uma ferramenta específica denominada *NER Annotator*. A Figura 3.2 mostra um trecho do prontuário utilizado para treinar o modelo com suas respectivas entidades para nomear, assim como as classes que elas devem ser reconhecidas.

De acordo com o que foi descrito na introdução deste capítulo, ele será dividido em dois exemplos. Para facilitar o entendimento do leitor optou-se por criar um exemplo ilustrativo que seja similar ao modelo *Med*7, que será melhor discutido na subseção posterior.

Dito isso, procurou-se, então, criar, treinar e testar um modelo que seja capaz de reconhecer sete tipos de entidades: dosagem (total), medicamento, duração, forma, frequência, via de administração e força. Portanto, os dados de entrada que o modelo deve receber e ser treinado é um texto anotado com essas classes específicas, como consta na Figura 3.2.

Por exemplo, as palavras "lorazepam"; "0.25 – 0.5 mg", "at bedtime" (que pode ser traduzida como "antes de deitar") e "as needed" (quando precisar) foram manualmente classificadas como "DRUG" (nome do medicamento); "STRENGHT" (força, quantidade do medicamento prescrito que há em uma dose); "FREQUENCY" (frequência, regime da dosagem que o medicamento deve ser administrado) e "DURATION" (duração, por quanto tempo o medicamento foi prescrito), respectivamente.

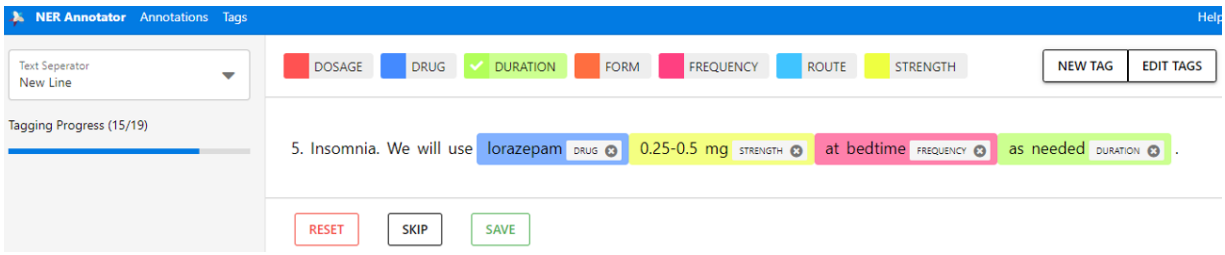

Figura 3.2: Amostra de prontuário anotado manualmente pela ferramenta *NER Annotator*.

A Figura 3.2 mostra como a ferramenta de anotação escolhida funciona e a Figura 3.3 ilustra como esses dados são salvos pelo *NER Annotator* e em qual formato eles estarão no modelo.

```
{"classes":["DOSAGE","DRUG","DURATION","FORM","FREQUENCY","ROUTE","STRENGTH"],
"annotations":[["5. Insomnia. We will use lorazepam 0.25-0.5 mg at bedtime as needed.",
{"entities":[[25,34,"DRUG"],[35,46,"STRENGTH"],[47,57,"FREQUENCY"],[58,67,"DURATION"]]}]]}
```

Figura 3.3: Amostra de prontuário anotado manualmente no formato *.json*.

Destaca-se que a Figura 3.3 é a visualização da teoria discutida no Exemplo 2.1 da

Seção 2.1, mostrando que os dados de entrada, para a biblioteca *spaCy* realizar a tarefa de *NER*, devem ser um arquivo *.json* que é formado basicamente por três *keys*:

- *Classes* (classes): representa as nomeações que serão feitas. Como citado anteriormente, neste contexto são sete: *DOSAGE, DRUG, DURATION, FORM, FREQUENCY, ROUTE* e *STRENGTH*;
- *Annotations* (anotações): caracteriza cada frase presente no prontuário. Para a ferramenta utilizada cada sentença é delimitada pelo ponto final;
- *Entities* (entidades): possui o começo, o fim da entidade e em qual classe esses *ID's* de *tokens* identificados serão classificados. Por exemplo, a palavra "lorazepam", anotada na Figura 3.2 como "DRUG", nesta lista é representada pela sequência "[25,34,"DRUG"]", a qual está indicando que a palavra possui como início e término, respectivamente, o vigésimo quinto ID e o trigésimo quarto ID da frase: "5. Insomnia. We will use lorazepam 0.25-0.5 mg at bedtime as needed." e esses nove IDs que denotam a palavra "lorazepam" serão nomeados como "DRUG".

Sendo assim, a primeira etapa de preparação dos dados consiste em utilizar uma ferramenta específica para, manualmente, classificar as entidades e salvar em um arquivo *.json* para, então, informar ao código que estes serão nossos dados de treinamento.

O *spaCy* usa a classe *DocBin*[1] para dados anotados. Portanto, a segunda e última etapa de preparação dos dados consistem em criar os objetos *DocBin* para os exemplos de treinamento. A classe *DocBin* serializa (ato de transformar um objeto em uma cadeia de bytes - dígitos binários - e desta forma poder ser manipulado de maneira mais fácil) com eficiência as informações de uma coleção de objetos *Doc*. É mais rápido e produz tamanhos de dados menores que o *pickle*, por exemplo, permitindo que o usuário "desserialize" (reverso da serialização, portanto, tomar dados estruturados a partir de algum formato e os reconstruir em um objeto) sem executar código *Python* arbitrário.

3.1.2 Configuração dos parâmetros do modelo

Com os dados que serão utilizados para treino, estando no formato adequado e prontos para uso, pode-se iniciar o segundo passo: preparação do modelo. Ela ocorre no arquivo *config.cfg*, que é um arquivo de configuração fornecido ao *spaCy* durante o processo de

[1] https://spacy.io/api/docbin

treinamento para que ele saiba o que treinar e como.

Para criar este arquivo, primeiramente, é preciso criar um arquivo *base_config.cfg*. Para fazer isso, pode-se usar o programa do *spaCy* (do inglês, *Graphical User Interface*, GUI), cuja interface é mostrada na Figura 3.4.

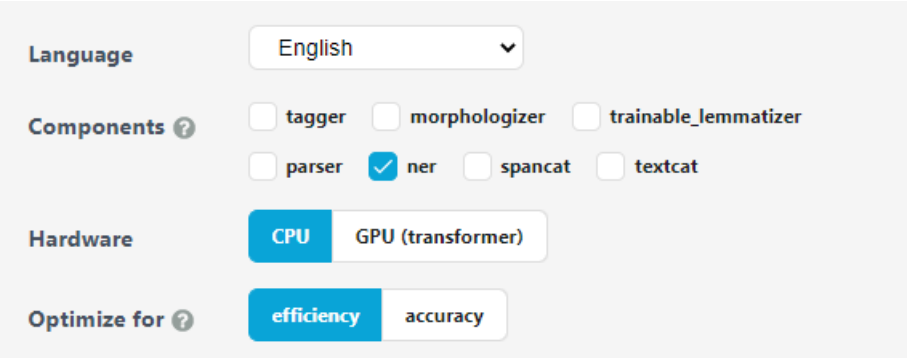

Figura 3.4: Recorte do *widget* de início rápido com a configuração escolhida para o modelo de treinamento.

Pela Figura 3.4 nota-se que, para seguir os propósitos deste trabalho, foram selecionados: *English* (inglês) como idioma que será treinado; *ner* como o único componente no *pipeline* para ser treinado; *CPU* (do inglês, *Central Processing Unit*) como o componente de hardware que será responsável por processar dados e transformar em informação e *efficiency* (eficiência), como parâmetro que otimizará o modelo. Essas seleções no *GUI* geram as linhas de códigos abaixo no arquivo *base_config.cfg* (veja Apêndice C).

```
[nlp]
lang = "en"
pipeline =
["tok2vec","ner"]
batch_size = 1000
```

Conforme já citado, com a configuração supracitada, definimos uma *pipeline* de treinamento usando um modelo de idioma inglês em branco, que contém um único módulo, o *NER* (a *tokenização*, representada por *tok2vec*, é gerada automaticamente para qualquer componente), a ser treinado, além de definir o número de amostras processadas antes que o modelo seja atualizado (*batch_size = 1000*). Salienta-se que todas essas configurações podem ser alteradas manualmente, por exemplo, poderíamos alterar o *batch_size* ou até mesmo retirá-lo.

Apesar do *spaCY* dar ao usuário muito controle sobre a arquitetura da rede neural e sobre todos os outros hiperparâmetros, essa facilidade de alteração cessa quando procura-se realizar alterações no código de níveis mais complexos, por exemplo, alterar o classificador de palavras de um *Perceptron* Multicamadas para uma Regressão Logística, tornando-se mais um problema de desenvolvedor/programador do que apenas uma simples mudança em uma etapa do arquivo.

Outro destaque no arquivo de configuração gerado, que contém o plano completo do modelo e processo de treinamento, refere-se à otimização, cujo código esta a seguir.

```
⋮
[training.optimizer]
@optimizers = Adam.v1
learn_rate = 0.001
beta1 = 0.9
beta2 = 0.999
eps = 1e-08
L2 = 1e-6
L2_is_weight_decay = true
grad_clip = 1.0
use_averages = true
⋮
```

Um otimizador executa, essencialmente, uma descida de gradiente estocástica, ou seja, ao invés de pegar todo o conjunto de dados, ele pega somente uma parte desses dados para fazer cada atualização dos pesos através do calculo do gradiente, encontrando, portanto, de forma iterativa, os valores dos parâmetros que minimizam determinada função de interesse. São necessários *arrays* unidimensionais para os pesos e seus gradientes e espera-se que o otimizador atualize os pesos e zere os gradientes no local. No caso foi configurado o otimizador *Adam*[2], que será responsável por retornar uma instância do *Optimizer*, ou seja, se um hiperparâmetro especificar um *schedule* ("agendamento") como uma lista ou gerador, seu valor será substituído pelo próximo item em cada chamada do *Optimi-*

[2]Adam foi apresentado por Diederik Kingma da OpenAI e Jimmy Ba da Universidade de Toronto em seu artigo "Adam: A Method for Stochastic Optimization" (Kingma e Ba, 2014). O nome Adam é derivado da estimativa adaptativa do momento

zer.step_schedules. Uma vez esgotado o *schedule*, será utilizado o seu último valor. Neste código, estão presentes os termos: taxa de aprendizado inicial de uma rede neural (learn_rate), que indica a que ritmo os pesos são atualizados; regularização $L2$, que remove uma pequena quantidade de pesos em cada iteração e, assim, os pesos nunca serão iguais a zero.

Agora que o arquivo *base_config* está configurado corretamente, é hora de convertê-lo em um arquivo *config.cfg*. Para fazer isso, é necessário executar um comando de terminal que pode ser feito no próprio Colab por meio do código:

```
!python -m spacy init fill-config base_config.cfg config.cfg
```

Ao executá-lo, o *spaCy* carrega as configurações do documento *base_config* no arquivo *config.cfg* este de formatação (`.cfg`) adequada para o ambiente. Com o arquivo *config.cfg* instalado, pode-se treinar, de fato, o modelo, assunto que será tratado na próxima subseção.

3.1.3 Treino, avaliação e teste do modelo

Ao treinar um modelo, não é útil que ele apenas memorize os exemplos, é necessário que ele também possa ser aplicado em dados não vistos. Afinal, não deseja-se que o modelo apenas aprenda que a instância "lorazepam" nesta amostra é nome de um medicamento, deseja-se que ele aprenda que "lorazepam", em contextos como esse, é provavelmente o nome de um medicamento. É por isso que os dados de treinamento devem sempre ser representativos dos dados que serão processados. Um modelo treinado em romances provavelmente terá um desempenho ruim no texto de saúde, por exemplo.

Isso também significa que, para saber como o modelo está se saindo e se ele está aprendendo as coisas certas, não é necessário apenas dados de treinamento, mas também de dados de avaliação. Ao testar o modelo apenas com os dados em que foi treinado, não haverá uma ideia de quão bem ele está generalizando. Quando deseja-se treinar um modelo do zero, geralmente é necessário pelo menos algumas centenas de exemplos para treinamento e avaliação, justificando, portanto, a escolha em ilustrar a aplicação da biblioteca em um modelo já treinado. Como esta seção busca trazer como o *spaCy* treina seus modelos, evidencia-se que não será realizado algumas boas práticas comuns a modelos de aprendizado de máquina como divisão da base em treino e teste, ou testar o modelo em

um conjunto de dados que não foi utilizado no treinamento. por exemplo.

Após configurado e carregado como serão os parâmetros do modelo (Subseção 3.1.2), realiza-se o treino por meio do comando:

```
!  python -m spacy train config.cfg --output ./ --paths.train
./annotations.spacy --paths.dev ./annotations.spacy
```

.

Primeiramente instala-se o documento que contém as configurações em que os parâmetros serão treinados (`!  python -m spacy train config.cfg`); direciona onde armazenar a saída do treinamento (`--output ./`); além de passar quais são os diretórios e documentos que devem ser usados para treino (`--paths.train ./annotations.spacy`) e validação do modelo (`--paths.dev ./annotations.spacy`). Observa-se que o mesmo prontuário de treino está sendo usado para criar as métricas de avaliação do modelo, o que provavelmente é responsável pela rápida conversão, taxa de aprendizado inicial de 0.001 e ótimas métricas conforme exibido na Figura 3.5.

Como resultado final, o comando gera dois documentos: um que contém o último modelo e o outro, com o melhor modelo do treinamento. Ambos são salvos no diretório *output* informado pelo usuário.

```
[ ] ℹ Saving to output directory: .
    ℹ Using CPU

    =========================== Initializing pipeline ===========================
    [2022-09-02 01:46:04,702] [INFO] Set up nlp object from config
    [2022-09-02 01:46:04,718] [INFO] Pipeline: ['tok2vec', 'ner']
    [2022-09-02 01:46:04,723] [INFO] Created vocabulary
    [2022-09-02 01:46:04,724] [INFO] Finished initializing nlp object
    [2022-09-02 01:46:04,937] [INFO] Initialized pipeline components: ['tok2vec', 'ner']
    ✓ Initialized pipeline

    ============================= Training pipeline =============================
    ℹ Pipeline: ['tok2vec', 'ner']
    ℹ Initial learn rate: 0.001
    E    #       LOSS TOK2VEC  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
    ---  ------  ------------  --------  ------  ------  ------  ------
      0       0          0.00     67.72    0.00    0.00    0.00    0.00
     39     200        161.24   1583.34  100.00  100.00  100.00    1.00
     79     400          0.00      0.00  100.00  100.00  100.00    1.00
    121     600          0.00      0.00  100.00  100.00  100.00    1.00
    167     800          0.00      0.00  100.00  100.00  100.00    1.00
    215    1000          0.00      0.00  100.00  100.00  100.00    1.00
    275    1200         12.87     13.49  100.00  100.00  100.00    1.00
    351    1400          0.00      0.00  100.00  100.00  100.00    1.00
    444    1600          0.00      0.00  100.00  100.00  100.00    1.00
    544    1800          0.23      0.04  100.00  100.00  100.00    1.00
    ✓ Saved pipeline to output directory
    model-last
```

Figura 3.5: Resultado do treinamento do modelo com a biblioteca *spaCy*.

A saída mostrada na Figura 3.5 nos informa os *epochs* (E); o número de interações (#); o valor da função de perda para cada componente do *pipeline* de execução configurado (*tok2vec* - *tokenização* e *ner* - reconhecimento de entidade nomeada), que, neste caso, indica quão bem o modelo generaliza os dados de treinamento e é usada para realizar "retropropagação", conforme citado em Seção 3.1; algumas métricas para todas as entidades nomeadas e a pontuação geral do *pipeline* (*score*), que é baseado no *training.score_weights* definido no *config.conf*. O treinamento continuará até que satisfaça uma das seguintes condições: atingir o *max_epochs*, atingir o *max_steps* ou atingir a *patience* (número de passos sem melhora no *score*), todos também definidos na configuração.

Salienta-se que, apesar de obter-se F1 (ents_F), precisão (ents_P) e recall (ents_R) para as entidades no geral (ou seja, métricas para todas as setes classes que há para se nomear cada entidade, não métricas de classe) de 100%, não é válido considerar que foi atingido um bom modelo, pois ele está superajustado (do inglês, *overfitting*), o que significa que ele essencialmente memorizou uma amostra, a que foi treinada. Essa conclusão será melhor discutida em conjunto da Figura 3.6.

Depois que o *pipeline* for treinado, ele armazenará o melhor e o último modelo no diretório de saída. Sendo assim, é possível carregar o melhor modelo e testá-lo no Prontuário 2 - Teste (veja Apêndice A). Para carregar o modelo, usa-se o comando *spacy.load(/content/model-best)*. A Figura 3.6 mostra o resultado obtido.

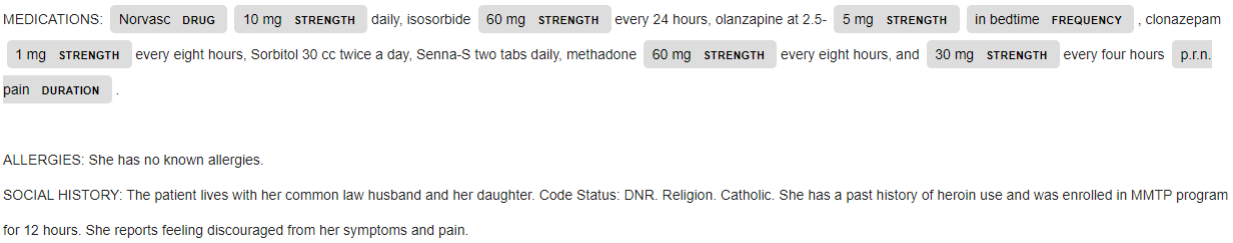

Figura 3.6: Parte do resultado do teste do modelo com a biblioteca *spaCy* em um novo prontuário médico.

Como era esperado, o modelo não possui um bom desempenho. No trecho do resultado obtido com o prontuário teste, mostrado na Figura 3.6, há diversas palavras que não foram reconhecidas, como, por exemplo: "*daily*" poderia ser rotulada como frequência; "*olanzapine*", poderia ser rotulada como medicamento ou "*tabs*" que é uma forma de comprimido de um remédio. Os modelos *NER* de aprendizado de máquina melhoram quando são alimentados com mais dados, especificamente, eles melhoram quando são

treinados com uma maior quantidade de dados de entrada variados. Estudos apontam que uma boa regra geral é começar com 200 amostras de treinamento e ir ajustando. No entanto, a proposta dessa seção é apenas exemplificar como construir um modelo específico de *NER* com a biblioteca *spaCy*, dando um noção de como funciona o processo de treinamento nesta biblioteca.

Na próxima seção, será discutido como é o desempenho e utilização do *Med7*, um modelo de processamento de linguagem natural clínica para reconhecimento de entidades nomeadas em registros eletrônicos de saúde, que já é empregado no dia a dia.

3.2 Med7

A identificação de conceitos médicos e a extração de informações é uma tarefa desafiadora assim como um ingrediente importante para analisar dados não estruturados em formato estruturado e tabulado para tarefas analíticas. Conforme mencionado nas seções anteriores, a construção de um modelo com bom desempenho no universo de processamento de linguagem natural, voltada para o reconhecimento de entidade nomeada direcionada ao contexto médico, requer centenas de dados de entrada para treinamento e avaliação. Dado o objetivo proposto para este trabalho, optou-se por demonstrar a aplicabilidade e praticidade da biblioteca *spaCy* pelo uso do "*Med7, um modelo de processamento de linguagem natural clínica transferível para registros eletrônicos de saúde*", sendo este o título do artigo de Kormilitzin *et al.* (2021) que norteará essa seção.

A Tabela 3.1 contém as sete categorias que o *Med7* foi treinado para reconhecer.

Tabela 3.1: Categorias e suas descrições para o modelo *Med7*.

Classe (em inglês)	**Classe (em português)**	**Descrição**
Dosage	Dosagem (total)	A quantidade total administrada de um medicamento.
Strenght	Força (cada medicamento)	A quantidade de um medicamento em uma determinada dosagem.
Drug	Nomes de medicamentos	Genérico ou nome comercial do medicamento.
Duration	Duração	O tempo que o medicamento foi prescrito (por 3 dias, crônico...).
Form	Forma	Configuração particular da droga que é comercializada para uso (tablete, comprimido, cápsula...).
Frequency	Frequência	O regime de dosagem em que o medicamento deve ser administrado (de 8h em 8h, uma vez ao dia...).
Route	Via de administração	O caminho pelo qual o medicamento é levado para o corpo .

Fonte: Traduzido de Kormilitzin *et al.* (2021).

Para exemplificar a aplicação em uma frase contendo todas as classes da Tabela 3.1, usou-se a oração: "*A patient was prescribed Magnesium hydroxide 400mg/5ml suspension PO*

of total 30ml bid for the next 5 days.". Ela pode ser traduzida como: A um paciente foi prescrito Hidróxido de Magnésio em suspensão líquida para ser tomado 400mg/5ml dos 30ml totais via oral quando precisar pelos os próximos 5 dias. Após aplicar o reconhecimento, utilizando os códigos que serão explicados nas subseções posteriores, foi obtido o seguinte resultado:

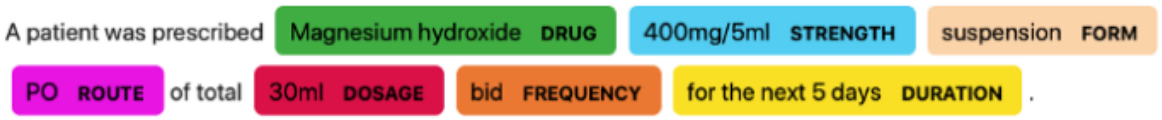

Figura 3.7: Resultado do reconhecimento de entidades na oração exemplo pelo modelo $Med7$.

Observa-se que o modelo classificou corretamente todas as possíveis entidades a ser reconhecidas. Destaca-se que essa frase demostra a necessidade de treinamento de um modelo para o domínio específico do problema uma vez que os termos *suspension* está relacionado a diluição do medicamento e não a suspensão como seria a tradução literal; *PO* equivale à sigla, em português, VO (via de administração oral) e *bid* significa "oferta", sendo considerado, neste caso, um "jargão" médico para tomar a medicação quando necessitar.

Posto isto, segundo os autores do artigo (Kormilitzin *et al.*, 2021), primeiramente o modelo auto-supervisionado foi pré-treinado prevendo a próxima palavra usando uma coleção de 2 milhões de registros de pacientes em texto livre da base de dados *MIMIC-III* e, em seguida, ajustado na tarefa de reconhecimento de entidade nomeada. Nele também foi implementado a reconstrução de palavras que o *spaCy* usa, semelhante aos objetivos do modelo de linguagem introduzidos em Devlin *et al.* (2018), mas, em vez de prever o identificador de palavra exato a partir do vocabulário, o vetor de palavras do *GloVe* foi previsto usando uma tabela de incorporação estática com uma função de perda de cosseno.

Conforme concluiu o artigo, o modelo alcançou uma pontuação $F1$ de 0,957 em todas as sete categorias. Além disso, foi avaliado como o modelo desenvolvido performou em outro conjunto de informações testando o modelo na base de dados da Unidade de Terapia Intensiva nos EUA para os registros de saúde mental de cuidados secundários (em inglês, $CRIS$) no Reino Unido. Uma aplicação direta do modelo NER, treinado aos dados do $CRIS$, resultou em desempenho reduzido de $F1 = 0,762$. No entanto, após o ajuste fino em uma pequena amostra do $CRIS$, o modelo alcançou um desempenho razoável de $F1 = 0,944$.

Nas seguintes subseções, será melhor descrito como foi usado o *córpus MIMIC-III* e será mostrado como aplicar este modelo no *Python* para o mesmo prontuário de teste utilizado como exemplo na Subseção 3.1.3.

3.2.1 Base de dados MIMIC-III

Nesta subseção será comentado a respeito do banco de dados descrito na Subseção 2.2.1, porém, agora, será descrito com enfoque na sua aplicação e usabilidade para construção e treinamento do modelo em estudo.

Os autores descrevem que o conjunto de dados anotado foi obtido dos registros de saúde eletrônicos do Mercado de Informações Médicas para Terapia Intensiva (em inglês, *Medical Information Mart for Intensive Care-III (MIMIC-III))* (Johnson *et al.*, 2016) como parte do evento de Desafios Clínicos Nacionais de *PNL*, realizado em 2018 (em inglês, *The 2018 National NLP Clinical Challenges - Track 2*, cuja sigla é *n2c2*). Neste evento houve uma tarefa compartilhada sobre conceitos relacionados à extração, eventos adversos a medicamentos (em inglês, *adverse drug events* (ADE)) e razões para prescrição. O conjunto de dados utilizado compreende uma coleção de altas médicas da Unidade de Terapia Intensiva (UTI) e contém informações muito ricas e detalhadas sobre os medicamentos utilizados para o tratamento.

Para a construção do modelo, é explicitado que o *córpus* foi dividido, aleatoriamente, em treinamento e teste, com 303 e 202 documentos, respectivamente. Os documentos foram anotados para nove categorias: ADE, dosagem (total), medicamento, duração, forma, frequência, razão, via de administração e força. No entanto, para seguir com os propósitos do artigo, foram consideradas apenas sete categorias relacionadas a medicamentos, desconsiderando duas categorias: ADE e razão. Também é descrito que objetivo dos autores foi desenvolver um modelo para medicamentos de forma que sua extração de informações fosse benéfico para a comunidade biomédica, sendo usado de forma robusta em uma variedade de tarefas de processamento de linguagem natural usando registros médicos de texto livre, similar ao objetivo secundário deste trabalho.

A Tabela 3.2 mostra algumas informações sobre os dados e como eles foram divididos. Destaca-se que o número de *tokens* exclusivos é calculado por palavras em minúsculas.

Tabela 3.2: Distribuição das entidades anotadas e demais informações nos conjuntos de dados de treinamento e teste.

Tipos de entidades anotadas	**Treino**	**Teste**	**Total**
Dosagem (total)	4227	2681	6908
Força (cada medicamento)	6694	4330	10924
Nomes de medicamentos	16257	10575	26832
Duração	592	378	970
Forma	6657	4359	11016
Frequência	6281	4012	10293
Via de administração	5460	3513	8973
Quantidade de documentos	303	202	505
Total de palavras	957972	627771	1585743
Total de palavras únicas	27602	21729	35763

Fonte: Traduzido de Kormilitzin *et al.* (2021).

Além dos conjuntos de dados *MIMIC-III* e daqueles presentes no *n2c2* de 2018, no artigo também foi avaliado o modelo desenvolvido em prontuários eletrônicos provenientes da plataforma de Pesquisa interativa de registro clínico (*UK-CRIS*) do Reino Unido, que é o maior banco de dados de saúde mental de cuidados secundários no Reino Unido. O UK-CRIS contém mais de 500 milhões de notas clínicas de 2,7 milhões de registros de pacientes não identificados de 12 Parceiros da Rede do Serviço Nacional de Saúde em todo o Reino Unido.

3.2.2 Passos para a aplicação do *Med7*

O primeiro passo para aplicação do *Med7*, no ambiente *Pyhton*, é a instalação e importação das bibliotecas ou pacotes necessários, que será realizada por meio dos comandos *! pip install* e *import*, respectivamente. No caso deste trabalho, será instalada e importada a biblioteca *spaCy*; importado o *pandas*, pacote construído em *Python* que é amplamente utilizado para análise e manipulação de dados e, também, será instalado e carregado o modelo *Med7*. Os códigos aqui utilizados foram baseados no *blog* público de um dos autores do artigo, Andrey Kormilitzin e encontram-se no Apêndice D.

Os modelos de vetor e transformador do *Med7* (no código: *en_core_med7_lg*) estão hospedados no repositório de modelos *Huggingface* [3], uma plataforma de ciência de dados que fornece ferramentas que permitem aos usuários criar, treinar e implantar modelos de aprendizado de máquina com base em tecnologias de código aberto. Ambos os modelos possuem como base sua implementação no *RoBERTa*, um método robusto e otimizado

[3] https://huggingface.co/kormilitzin

para pré-treinamento de sistemas de processamento de linguagem natural, que melhora as representações de codificador bidirecional de transformadores (*BERT*), o método auto-supervisionado lançado pelo *Google* em 2018. O *BERT* pode ser considerado uma técnica revolucionária que atinge resultados avançados no campo do *PLN* se baseando em textos não anotados extraídos da *internet*, em oposição a *córpus* que foram rotulados especificamente para uma tarefa.

Após instalado e importado o modelo, basicamente, há apenas mais um passo para que ele realize a tarefa *NER*: carregá-lo por meio do código *en_core_med7_lg.load()*. A seguir, é mostrado resultado obtido pelo modelo, ou seja, os conceitos que foram identificados, relembrando que foi utilizado o mesmo prontuário teste da Subseção 3.1.3.

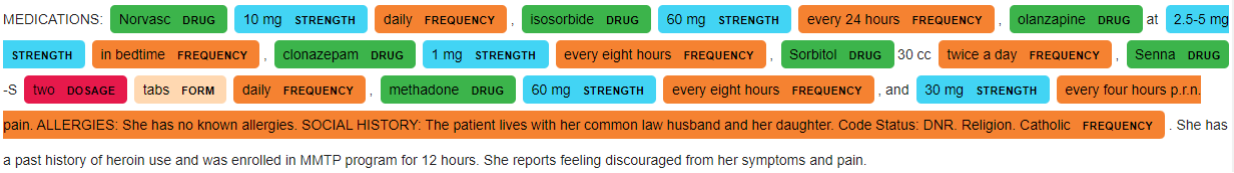

Figura 3.8: Parte do resultado do modelo *Med7* para *NER* aplicado em um prontuário médico.

Percebe-se que, apesar do modelo *NER* desenvolvido e estudado nesta seção ter sido treinado em um *córpus* razoavelmente grande e adequado para o contexto, ainda verifica-se um problema que foi pontuado nos capítulos iniciais deste trabalho: *NER* tornou-se mais um problema de dados do que um problema de algoritmo. A Figura 3.8 exemplifica isto, uma vez que há algumas entidades que não foram reconhecidas ou foram reconhecidas erroneamente, como, por exemplo, a sentença: "*every four hours p.r.n. pain. ALLERGIES: She has no known allergies.*" (tradução: "A cada quatro horas diante da presença de dor. Alergias: ela não possui nenhuma alergia conhecida."), que foi toda reconhecida como *FREQUENCY* quando, claramente, não é.

Capítulo 4

Considerações Finais

Este trabalho se propôs a melhorar o entendimento dos princípios e das etapas associadas ao NER, como também trazer uma maior profundidade teórica e estatística ao funcionamento da biblioteca *spaCy*. A fim de atingir esse objetivo no primeiro capítulo foi introduzido como essas técnicas e ferramentas são de grande importância para poder obter, por exemplo, detalhes da riqueza dos dados presentes em campos textuais como a anamnese, uma das principais fontes de informação para área da saúde, tanto para o tratamento do paciente como para a conduta da gestão e pesquisa clínica. Explicitou-se também que neste contexto trabalhar com um dado não estruturado, dificulta a sua manipulação e a extração de resultados mais corretos e replicáveis.

O Capítulo 2 pode ser considerado o que concentra a maior dificuldade deste trabalho e consequentemente uma das maiores motivações. Para sua construção foram estudados e referenciados inúmeros livros, artigos, entre outros, pois, cada um, por diversas vezes, possuía a descritiva do assunto em até certo ponto, sendo necessário e indicada a leitura dos artigos que constavam na referência do artigo em uso para um maior compreensão.

Outro obstáculo constantemente enfrentado é que, ao mesmo tempo em que havia uma escassez de materiais que tratassem em detalhes do tema, contexto, técnica e recursos escolhidos, também encontrou-se incontáveis metodologias e ferramentas para se realizar cada e qualquer passo do reconhecimento de uma entidade nomeada. Encerrando os entraves enfrentados para a escrita do Capítulo 2, ressalta-se a dificuldade de encontrar detalhes de como são utilizadas as metodologias estatísticas citadas em todos as documentações do *spaCy*, fundamentais para o desenvolvimento de cada passo e dos projetos

derivativos da biblioteca.

Todos os obstáculos supracitados encontrados para esse capítulo foram o que motivou a escrita de uma seção específica para a biblioteca *spaCy*, pois, expôs a necessidade da junção da teoria com a aplicabilidade em forma de um tutorial. Diante disso, criou-se o Capítulo 3 como uma forma desta monografia se tornar um material didático em português, específico para este tema de forma que um desinformado do assunto possa lê-lo e compreender o que está sendo realizado e os resultados que estão sendo atingidos.

Com o intuito de apresentar uma possível proposta de estruturação de documentos textuais médicos, o penúltimo capítulo deste trabalho mostrou como implementar a técnica descrita utilizando a biblioteca de interesse. Logo no início este capítulo resultou em algo não imaginado, uma vez que o exemplo ilustrativo de criação de um modelo para NER, customizado para o contexto da saúde, pôde ser expandido para diversos outros contextos, desde que o usuário tenha uma grande quantidade de dados apropriados e anotados.

No Capítulo 3 também foram obtidos algumas conclusões interessantes. Salienta-se que, ao estudar como treinar um modelo de *NER* com o *spaCy*, encontrou-se o obstáculo de ter restrições de acesso as anotações médicas do corpus descrito nos Capítulo 2 e Capítulo 3, assim como a impossibilidade de fazer grandes alterações nas configurações de como a biblioteca realiza as classificações das palavras, como trocar o *Perceptron* Multicamadas por uma Regressão Logística, por exemplo. Seguindo dentro das restrições impostas e dos objetivos traçados obteve-se como performance para o exemplo ilustrativo métricas que atingiram 100% de precisão, *recall* e $F1$, assim como funções de perdas com valores 0 e uma taxa de aprendizado inicial de 0,001, tudo isso devido ao modelo ter sido treinado e validado no mesmo conjunto de dados, ou seja, ele performava bem pois esta sendo superajustado, porém, como seu propósito era apenas de caráter ilustrativo tal fato foi considerado um problema esperado.

Em relação às conclusões referentes ao modelo pré-treinado *Med7*, será citada (em português) a conclusão dos próprios autores do artigo (Kormilitzin *et al.*, 2021), o qual norteou o entendimento do mesmo, uma vez que esta também representa a conclusão obtida no desenvolvimento desse trabalho:

> [...] Demonstramos que o aprendizado de transferência desempenha um papel essencial no desenvolvimento de um modelo robusto aplicável em diferentes

> domínios clínicos e o modelo *Med7* desenvolvido não requer uma infraestrutura cara e pode ser usado em máquinas padrão com *CPU*. Mais pesquisas são necessárias para melhorar o reconhecimento de conceitos naturalmente sub-representados e estamos planejando resolver esse problema, bem como normalização de conceitos extraídos [..]

Como um trabalho futuro sugere-se a comparação entre outros modelos pré-treinados existentes para este contexto, como, por exemplo, a comparação entre o *Med7* e o *scispaCy*, uma biblioteca do *Python* que contém pipelines, modelos e componentes *spaCy* (como um detector de abreviação e um vinculador de entidade que liga *tokens* a uma ontologia) para processamento de texto biomédico, científico ou clínico. Ou até usar no modelo *Med7* outros conceitos ligados ao *PLN* clínico que a própria biblioteca *spaCy* também dispõe, como a negação - contradição ou negação de algo. Em um contexto clínico, a ausência documentada de algo, por exemplo, uma doença, pode ser uma informação valiosa. Esta técnica pode ser implementada usando o *negspaCy*, um componente de *pipeline spaCy* para negar conceitos em um texto.

Referências Bibliográficas

Almeida, J. R. d. (2020). Transfer learning e convolutional neural networks para a classificação de imagens e reconhecimento de objetos no âmbito da perícia criminal.

Camacho-Collados, J. e Pilehvar, M. T. (2017). On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis.

Castro, P. V. Q. (2019). *Aprendizagem profunda para reconhecimento de entidades nomeadas em domínio jurídico*. Tese de doutorado, Universidade Federal de Goias, Goiânia.

Devlin, J., Chang, M.-W., Lee, K. e Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Goldberg, Y. (2017). *Neural network methods in Natural Language Processing*. Morgan amp; Claypool Publishers.

Hajjar, A. J. (2021). In-depth guide to nlp: What it is, how it works top use cases. *AI multiple research*.

Haykin, S. e Engel, P. M. (2007). *Redes neurais: Principios E pratica*. Artmed.

Honnibal, Matthew e Montani, I. (2020). spacy 101: Everything you need to know. https://spacy.io/usage/spacy-101. Acesso em: 01 fev. 2022.

Honnibal, M., Montani, I., Van Landeghem, S. e Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python.

Jayan, J. P., Rajeev, R. e Sherly, E. (2013). A hybrid statistical approach for named entity recognition for malayalam language. Em *Proceedings of the 11th Workshop on Asian Language Resources*, páginas 58–63.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L. e Mark, R. G. (2016). Mimic-iii, a freely accessible critical care database. *Scientific data*, **3**(1), 1–9.

Jurasfsky, D. e Martin, J. H. (2008). *Speech and language processing*. Pearson Prentice Hall, second edition.

Kingma, D. P. e Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kormilitzin, A., Vaci, N., Liu, Q. e Nevado-Holgado, A. (2021). Med7: a transferable clinical natural language processing model for electronic health records. *Artificial Intelligence in Medicine*, **118**, 102086.

Kundeti, S. R., Vijayananda, J., Mujjiga, S. e Kalyan, M. (2016). Clinical named entity recognition: Challenges and opportunities. Em *2016 IEEE International Conference on Big Data (Big Data)*, páginas 1937–1945.

Li, J., Sun, A., Han, J. e Li, C. (2018). A survey on deep learning for named entity recognition. *CoRR*, **abs/1812.09449**.

Manning, C. D. e Schutze, H. (2008). *Foundations of Statistical Natural Language Processing*. MIT.

Maverick, G. V. (1969). Computational analysis of present-day american english.

Pennington, J., Socher, R. e Manning, C. D. (2014). Glove: Global vectors for word representation. Em *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, páginas 1532–1543.

Privatto, P. I. M. (2020). Uma abordagem para reconhecimento de entidades nomeadas usando conhecimento externo.

Sánchez, A. *et al.* (1995). Definición e historia de los corpus. *CUMBRE–Corpus linguistico de espanol contemporaneo. Madrid: SGEL*, páginas 7–24.

Santos, F. A. O. (2018). Sobre o uso de conhecimento especialista para auxiliar no aprendizado de word embeddings.

Silva, A. V. (2020). *Um modelo de classificação para o Reconhecimento de Entidades Nomeadas*. Tese de doutorado, Universidade de São Paulo.

Silva, J. R. d. (2019). Geração de vetores de sentido para o português.

Treméa, S. (2021). *Redes Neurais Aplicadas a Grafos: Uma Abordagem Semi-Supervisionada*. Trabalho de Conclusão de Curso, Universidade Federal de São Carlos.

Turing, A. M. (1950). I. — Computing Machinery and Intelligence. *Mind*, **LIX**(236), 433–460.

Apêndice A

Prontuários - Exemplo Ilustrativo

- **Prontuário 1 - Treino:**

  "HISTORY OF PRESENT ILLNESS: The patient is a 41-year-old man with the AIDS complicated with recent cryptococcal infection, disseminated MAC and Kaposi's sarcoma. His viral load in July of 2007 was 254,000 and CD4 count was 7. He was recently admitted for debility and possible pneumonia. He was started on antiretroviral therapy, as well as Cipro and Flagyl and was also found to have pleural effusion on the right. His history is also significant for pancreatitis and transient renal failure during last hospitalization. He became frustrated since he was not getting better and discontinued all antibiotics. When taken home yesterday, he had symptoms consistent with a partial bowel obstruction. He was vomiting and had no bowel movements for a few days. Last night, was able to have a bowel movement and has not vomited since then. He was able to take small amounts of food. He now has persistent cough productive of clear sputum and some shortness of breath. He also complains of pain at his KS lesions on the right leg and left thigh, especially when touched, although that pain is incidental and not present when he is simply lying down. He has overall weakness.

  PAST MEDICAL HISTORY: Unremarkable.

  MEDICATIONS: Acetaminophen 650 mg q.6h. p.r.n. fever, which he has not been using, Motrin 400 mg q.6h. p.r.n. pain, which has not helped. His pain and dexamethasone with guaifenesin 5-10 mL q.4h. p.r.n. cough.

ALLERGIES: He has no known allergies.

SOCIAL HISTORY: The patient is now staying with his mother. He is the youngest of six children. Code Status: DNR. His brother is the health care proxy.

PHYSICAL EXAMINATION: Blood pressure 140/80, pulse 120, and respirations 28. Temperature 103.9. General Appearance: Ill-looking young man, diaphoretic. PERRLA, 3 mm. Oral mucosa moist without lesions. Lungs: Diminished breath sounds in the right middle lower lobe. Heart: RRR without murmurs. Abdomen: Distended with soft and nontender. Diminished bowel sounds. Extremities: Without cyanosis or edema. There is a large Kaposi's sarcoma on the right medial leg and left medial proximal thigh, which is somewhat tender. Neurological Exam: Cranial nerves II through XII are grossly intact. There is normal tone. Power is 4/5. DTRs nonreactive. Normal fine touch. Mental Status: The patient is somnolent, but arousable. Withdrawn affect. Normal speech and though process.

ASSESSMENT AND PLAN:

1. AIDS complicated with multiple opportunistic infections with poor performance status, which suggested a limited prognosis of less than six months. He will benefit from home hospice care and he declined any further antibiotic or antiretroviral treatments.

2. Pain, which is somatic nociceptive from KS lesions. The patient has not tolerated morphine in the past. We will start oxycodone 5 mg q.2h. as needed.

3. Cough. We will use oxycodone with the same indication as well.

4. Fever. We encouraged him to use Tylenol as needed.

5. Insomnia. We will use lorazepam 0.25-0.5 mg at bedtime as needed.

6. Psychosocial. We discussed his coping with the diagnosis. He is fully aware of his limited prognosis. Supportive counseling was provided to his mother.

Length of the encounter was one hour; more than half spent on exchange of information."

- **Prontuário 2 - Teste:**

"HISTORY OF PRESENT ILLNESS: The patient is a 55-year-old woman with

carcinoma of the cervix metastatic to retroperitoneum, lung, which was diagnosed approximately two years ago. There is a nodule in her lung, which was treated by excision in February of 2007 on the right side. She had spread to her kidney. She had right-sided nephrectomy and left-sided nephrostomy. She also had invasion of the bladder. Currently, all of her urine comes out through the renal nephrostomy. She complains of burning vaginal pain, as well as chronic discharge, which has improved slightly recently. She is not able to engage in intercourse because of the pain and bleeding. She also has pain with bowel movements, as well as painful urgency. The pain is at least 3-4/10 and is partially relieved with methadone rescues and interferes with her ability to sleep at night as she feels exhausted and tired. She has some nausea and diminished appetite. No hallucinations. She is anxious frequently and this is helped with clonazepam, which she has taken chronically for her anxiety disorder and recently started Zyprexa. She has occasional shortness of breath, which used to be helped with oxygen in the hospital.

PAST MEDICAL HISTORY: Peptic ulcer disease, hypertension.

REVIEW OF SYSTEMS: She has constipation with hard bowel movements.

MEDICATIONS: Norvasc 10 mg daily, isosorbide 60 mg every 24 hours, olanzapine at 2.5-5 mg in bedtime, clonazepam 1 mg every eight hours, Sorbitol 30 cc twice a day, Senna-S two tabs daily, methadone 60 mg every eight hours, and 30 mg every four hours p.r.n. pain.

ALLERGIES: She has no known allergies.

SOCIAL HISTORY: The patient lives with her common law husband and her daughter. Code Status: DNR. Religion. Catholic. She has a past history of heroin use and was enrolled in MMTP program for 12 hours. She reports feeling discouraged from her symptoms and pain.

PHYSICAL EXAMINATION: Blood pressure 120/80, pulse 80, and respirations 14. General Appearance: Mildly obese woman. PERRLA, 3 mm. Oral mucosa moist without lesions. Lungs: Clear. Heart: RRR without murmurs. Abdomen: Somewhat distended, but soft and nontender. There is firmness found in the low abdomen bilaterally. There is erythema in the intertriginous area and vulva, as well as some serous discharge from the vagina. Neurological Exam: Cranial nerves

II through XII are grossly intact. There is normal tone. Power is 5-/5. DTRs nonreactive. Sensation intact to fine touch. Mental Status: The patient is alert, fully oriented, normal speech, and thought process. Normal affect.

ASSESSMENT AND PLAN:

1. Carcinoma of the cervix metastatic to the retroperitoneum, bladder, and lung with irritable obstruction and gradual decline in the performance status. Given this, her prognosis is likely to be limited to six months and she will benefit from home hospice care.

2. Pain, which is a combination of somatic nociceptive pain due to the retroperitoneal invasion, as well as a neuropathic component from pelvic and nerve involvement by the surgery as well as radiation therapy and disease itself. We are going to increase methadone to 70 mg every eight hours and continue 30 mg for breakthrough. We will add pregabalin 50 mg three times a day and titrate the dose up as needed.

3. Nausea and poor appetite. We will start Megace 200 mg daily.

4. Shortness of breath. We will provide oxygen p.r.n.

5. Candidal infection. We will start clotrimazole 1% cream b.i.d.

6. Constipation. We will advance the bowel regimen to Sorbitol 30 cc three times a day and Senna-S three tabs twice a day.

7. Psychosocial. The patient is getting discouraged. We will provide supportive counseling.

Length of the encounter was 80 minutes; more than half spent on exchange of information.

Thank you for the opportunity to participate in the care for this patient."

Apêndice B

Anotação completa - Prontuário de treino

A seguir tem-se como ficou o arquivo final que treinou o modelo do exemplo ilustrativo, ou seja, após como cada entidade foi anotada e o resultado em formato *json* desse documento.

{"classes":["DOSAGE","DRUG","DURATION","FORM","FREQUENCY","ROUTE","STRENGTH"],
"annotations":[["HISTORY OF PRESENT ILLNESS: The patient is a 41-year-old man with the AIDS complicated with recent cryptococcal infection, disseminated MAC and Kaposi's sarcoma. His viral load in July of 2007 was 254,000 and CD4 count was 7. He was recently admitted for debility and possible pneumonia. He was started on antiretroviral therapy, as well as Cipro and Flagyl and was also found to have pleural effusion on the right. His history is also significant for pancreatitis and transient renal failure during last hospitalization. He became frustrated since he was not getting better and discontinued all antibiotics. When taken home yesterday, he had symptoms consistent with a partial bowel obstruction. He was vomiting and had no bowel movements for a few days. Last night, was able to have a bowel movement and has not vomited since then. He was able to take small amounts of food. He now has persistent cough productive of clear sputum and some shortness of breath. He also complains of pain at his KS lesions on the right leg and left thigh, especially when touched, although that pain is incidental and not present when he is simply lying down. He has overall weakness.\r",{"entities":[[306,328,"DRUG"],

[341,346,"DRUG"],[351,357,"DRUG"],[596,608,"DRUG"]]}],["PAST MEDICAL HISTORY: Unremarkable.\r",{"entities":[]}],["MEDICATIONS: Acetaminophen 650 mg q.6h. p.r.n. fever, which he has not been using, Motrin 400 mg q.6h. p.r.n. pain, which has not helped. His pain and dexamethasone with guaifenesin 5-10 mL q.4h. p.r.n. cough.\r",{"entities":[[13,26,"DRUG"],[27,33,"STRENGTH"],[34,39,"FREQUENCY"],[40,52,"DURATION"],[83,89,"DRUG"],[90,96,"STRENGTH"],[97,102,"FREQUENCY"],[103,114,"DURATION"],[151,164,"DRUG"],[170,181,"DRUG"],[182,189,"STRENGTH"],[190,195,"FREQUENCY"],[196,208,"DURATION"]]}],["ALLERGIES: He has no known allergies.\r",{"entities":[]}],["SOCIAL HISTORY: The patient is now staying with his mother. He is the youngest of six children. Code Status: DNR. His brother is the health care proxy.\r",{"entities":[]}],["PHYSICAL EXAMINATION: Blood pressure 140/80, pulse 120, and respirations 28. Temperature 103.9. General Appearance: Ill-looking young man, diaphoretic. PERRLA, 3 mm. Oral mucosa moist without lesions. Lungs: Diminished breath sounds in the right middle lower lobe. Heart: RRR without murmurs. Abdomen: Distended with soft and nontender. Diminished bowel sounds. Extremities: Without cyanosis or edema. There is a large Kaposi's sarcoma on the right medial leg and left medial proximal thigh, which is somewhat tender. Neurological Exam: Cranial nerves II through XII are grossly intact. There is normal tone. Power is 4/5. DTRs nonreactive. Normal fine touch. Mental Status: The patient is somnolent, but arousable. Withdrawn affect. Normal speech and though process.\r",{"entities":[]}],["ASSESSMENT AND PLAN:\r",{"entities":[]}],["1. AIDS complicated with multiple opportunistic infections with poor performance status, which suggested a limited prognosis of less than six months. He will benefit from home hospice care and he declined any further antibiotic or antiretroviral treatments.\r",{"entities":[[217,227,"DRUG"],[231,256,"DRUG"]]}],["2. Pain, which is somatic nociceptive from KS lesions. The patient has not tolerated morphine in the past. We will start oxycodone 5 mg q.2h. as needed.\r",{"entities":[[85,93,"DRUG"],[121,130,"DRUG"],[131,135,"STRENGTH"],[136,141,"FREQUENCY"],[142,151,"DURATION"]]}],["3. Cough. We will use oxycodone with the same indication as well.\r",{"entities":[[22,31,"DRUG"]]}],["4. Fever. We encouraged him to use Tylenol as needed.\r",{"entities":[[35,42,"DRUG"],[43,52,"FREQUENCY"]]}],["5. Insomnia. We will use lorazepam 0.25-0.5 mg at bedtime as needed.\r",{"entities":[[25,34,

"DRUG"],[35,46,"STRENGTH"],[47,57,"FREQUENCY"],[58,67,"DURATION"]]}],["6. Psychosocial. We discussed his coping with the diagnosis. He is fully aware of his limited prognosis. Supportive counseling was provided to his mother.\r", {"entities":[]}],["Length of the encounter was one hour; more than half spent on exchange of information.",{"entities":[]}]]}

Apêndice C

Configuração dos parâmetros

A seguir encontra-se o arquivo completo da configuração usada para treinar os parâmetros do exemplo ilustrativo (Subseção 3.1.2)

```
    # This is an auto-generated partial config. To use it with 'spacy train'
# you can run spacy init fill-config to auto-fill all default settings:
# python -m spacy init fill-config ./base_config.cfg ./config.cfg
[paths]
train = null
dev = null
vectors = null
[system]
gpu_allocator = null

[nlp]
lang = "en"
pipeline = ["tok2vec","ner"]
batch_size = 1000

[components]

[components.tok2vec]
factory = "tok2vec"
```

```
[components.tok2vec.model]
@architectures = "spacy.Tok2Vec.v2"

[components.tok2vec.model.embed]
@architectures = "spacy.MultiHashEmbed.v2"
width = ${components.tok2vec.model.encode.width}
attrs = ["ORTH", "SHAPE"]
rows = [5000, 2500]
include_static_vectors = false

[components.tok2vec.model.encode]
@architectures = "spacy.MaxoutWindowEncoder.v2"
width = 96
depth = 4
window_size = 1
maxout_pieces = 3

[components.ner]
factory = "ner"

[components.ner.model]
@architectures = "spacy.TransitionBasedParser.v2"
state_type = "ner"
extra_state_tokens = false
hidden_width = 64
maxout_pieces = 2
use_upper = true
nO = null

[components.ner.model.tok2vec]
@architectures = "spacy.Tok2VecListener.v1"
width = ${components.tok2vec.model.encode.width}
```

```
[corpora]

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
max_length = 0

[corpora.dev]
@readers = "spacy.Corpus.v1"
path = ${paths.dev}
max_length = 0

[training]
dev_corpus = "corpora.dev"
train_corpus = "corpora.train"

[training.optimizer]
@optimizers = "Adam.v1"
learn_rate = 0.001
beta1 = 0.9
beta2 = 0.999
eps = 1e-08
L2 = 1e-6
L2_is_weight_decay = true
grad_clip = 1.0
use_averages = true

[training.batcher]
@batchers = "spacy.batch_by_words.v1"
discard_oversize = false
tolerance = 0.2
```

```
[training.batcher.size]
@schedules = "compounding.v1"
start = 100
stop = 1000
compound = 1.001

[initialize]
vectors = ${paths.vectors}
```

Caso deseje obter mais informações sobre as funções que são usadas nessa configuração dos parâmetros acesse as páginas Thinc, que contém informações sobre as funções referentes a *schedules* e *optimizers* e a página *Model Architectures* que contém informações sobre as demais.

Apêndice D

Códigos em Python

A seguir encontram-se os códigos em Python da biblioteca *spaCy* e do modelo *Med7* utilizados nesta monografia, conforme citado em Capítulo 3.

```
# Instalando bibliotecas e modelos necessários
! pip install -U spacy -q

!python -m spacy info

import spacy
from spacy.tokens import DocBin
from tqdm import tqdm # para criar medidores de progresso/barras de progresso

import spacy
from spacy.tokens import DocBin
from tqdm import tqdm # para criar medidores de progresso/barras de progresso

nlp = spacy.blank("en") # Carregar um novo modelo spaCy em branco porém na
                        #lingua inglesa
db = DocBin() # Criar um objeto DocBin

import json
```

```
f = open('annotations.json') # Dados de treino feitos com auxílio da ferramenta
                             #NER Annotator
TRAIN_DATA = json.load(f)

# Lendo os dados de treinamento acima e armazenando no formato de
#objeto DocBin com seus rótulos definidos

for text, annot in tqdm(TRAIN_DATA['annotations']): # Dados no formato anterior
                                                    # o de treino
    doc = nlp.make_doc(text) # Criar objeto doc a partir de texto
    ents = []
    for start, end, label in annot["entities"]: # Adicionar índices dos caracteres
        span = doc.char_span(start, end, label=label, alignment_mode="contract")
        if span is None:
            print("Skipping entity")
        else:
            ents.append(span) # Método utilizado para lidar qundo os índices de
                              # algumas entidades se sobrepõem
    doc.ents = ents # Classificar o texto com as entidades
    db.add(doc)

db.to_disk("./annotations.spacy") # Salvar o objeto DocBin

# Preparando o arquivo de configuração para treinamento do modelo spaCy em dados
#personalizados
!python -m spacy init fill-config base_config.cfg config.cfg

# Treinar o modelo spaCy existente com novos parâmetros especificados
! python -m spacy train config.cfg --output ./ --paths.train ./annotations.spacy
--paths.dev ./annotations.spacy

nlp_ner = spacy.load("/content/model-best") #salvando o melhor modelo
```

```
# Testando em um novo prontuário
doc = nlp_ner("""HISTORY OF PRESENT ILLNESS: The patient is a 55-year-old woman
with carcinoma of the cervix metastatic to retroperitoneum, lung, which was
diagnosed approximately two years ago. There is a nodule in her lung, which was
treated by excision in February of 2007 on the right side. She had spread to her
kidney. She had right-sided nephrectomy and left-sided nephrostomy. She also had
invasion of the bladder. Currently, all of her urine comes out through the renal
nephrostomy. She complains of burning vaginal pain, as well as chronic discharge,
which has improved slightly recently. She is not able to engage in intercourse
because of the pain and bleeding. She also has pain with bowel movements, as well
as painful urgency. The pain is at least 3-4/10 and is partially relieved with
methadone rescues and interferes with her ability to sleep at night as she feels
exhausted and tired. She has some nausea and diminished appetite. No
hallucinations. She is anxious frequently and this is helped with clonazepam,
which she has taken chronically for her anxiety disorder and recently started
Zyprexa. She has occasional shortness of breath, which used to be helped with
oxygen in the hospital.

PAST MEDICAL HISTORY: Peptic ulcer disease, hypertension.
REVIEW OF SYSTEMS: She has constipation with hard bowel movements.
MEDICATIONS: Norvasc 10 mg daily, isosorbide 60 mg every 24 hours, olanzapine at
 2.5-5 mg in bedtime, clonazepam 1 mg every eight hours, Sorbitol 30 cc twice a
 day, Senna-S two tabs daily, methadone 60 mg every eight hours, and 30 mg every
  four hours p.r.n. pain.

ALLERGIES: She has no known allergies.
SOCIAL HISTORY: The patient lives with her common law husband and her daughter.
 Code Status: DNR. Religion. Catholic. She has a past history of heroin use and
 was enrolled in MMTP program for 12 hours. She reports feeling discouraged from
  her symptoms and pain.
PHYSICAL EXAMINATION: Blood pressure 120/80, pulse 80, and respirations 14.
General Appearance: Mildly obese woman. PERRLA, 3 mm. Oral mucosa moist without
lesions. Lungs: Clear. Heart: RRR without murmurs. Abdomen: Somewhat distended,
```

```
but soft and nontender. There is firmness found in the low abdomen bilaterally.
There is erythema in the intertriginous area and vulva, as well as some serous
discharge from the vagina. Neurological Exam: Cranial nerves II through XII are
grossly intact. There is normal tone. Power is 5-/5. DTRs nonreactive. Sensation
intact to fine touch. Mental Status: The patient is alert, fully oriented,
normal speech, and thought process. Normal affect.

ASSESSMENT AND PLAN:
1. Carcinoma of the cervix metastatic to the retroperitoneum, bladder, and lung
with irritable obstruction and gradual decline in the performance status. Given
this, her prognosis is likely to be limited to six months and she will benefit
from home hospice care.
2. Pain, which is a combination of somatic nociceptive pain due to the retroperitoneal
invasion, as well as a neuropathic component from pelvic and nerve involvement by
the surgery as well as radiation therapy and disease itself. We are going to
increase methadone to 70 mg every eight hours and continue 30 mg for breakthrough.
We will add pregabalin 50 mg three times a day and titrate the dose up as needed.
3. Nausea and poor appetite. We will start Megace 200 mg daily.
4. Shortness of breath. We will provide oxygen p.r.n.
5. Candidal infection. We will start clotrimazole 1% cream b.i.d.
6. Constipation. We will advance the bowel regimen to Sorbitol 30 cc three times a
day and Senna-S three tabs twice a day.
7. Psychosocial. The patient is getting discouraged. We will provide supportive
counseling.

Length of the encounter was 80 minutes; more than half spent on exchange of
information. Thank you for the opportunity to participate in the care for this patient
."""") # input sample text

from spacy import displacy
displacy.render(doc, style="ent", jupyter=True)

################## MED7
```

```
!pip install -U spacy  #instalando através do pip a biblioteca que será utilizada

! pip install https://huggingface.co/kormilitzin/en_core_med7_lg/
resolve/main/en_core_med7_lg-any-py3-none-any.whl

! pip install https://huggingface.co/kormilitzin/en_core_med7_trf/
resolve/main/en_core_med7_trf-any-py3-none-any.whl

import spacy # biblioteca importada
import pandas as pd # biblioteca importada
import en_core_med7_lg # modelo importado

med7 = en_core_med7_lg.load() #aqui estamos carregando o modelo pré pronto que
                              #será utilizado e o chamaremos de
                              # med7en_core_med7_lg.load()

text = """HISTORY OF PRESENT ILLNESS: The patient is a 55-year-old woman
with carcinoma of the cervix metastatic to retroperitoneum, lung, which was
diagnosed approximately two years ago. There is a nodule in her lung, which was
treated by excision in February of 2007 on the right side. She had spread to her
kidney. She had right-sided nephrectomy and left-sided nephrostomy. She also had
invasion of the bladder. Currently, all of her urine comes out through the renal
nephrostomy. She complains of burning vaginal pain, as well as chronic discharge,
which has improved slightly recently. She is not able to engage in intercourse
because of the pain and bleeding. She also has pain with bowel movements, as well
as painful urgency. The pain is at least 3-4/10 and is partially relieved with
methadone rescues and interferes with her ability to sleep at night as she feels
exhausted and tired. She has some nausea and diminished appetite. No
hallucinations. She is anxious frequently and this is helped with clonazepam,
which she has taken chronically for her anxiety disorder and recently started
Zyprexa. She has occasional shortness of breath, which used to be helped with
oxygen in the hospital.
```

PAST MEDICAL HISTORY: Peptic ulcer disease, hypertension.
REVIEW OF SYSTEMS: She has constipation with hard bowel movements.
MEDICATIONS: Norvasc 10 mg daily, isosorbide 60 mg every 24 hours, olanzapine at 2.5-5 mg in bedtime, clonazepam 1 mg every eight hours, Sorbitol 30 cc twice a day, Senna-S two tabs daily, methadone 60 mg every eight hours, and 30 mg every four hours p.r.n. pain.

ALLERGIES: She has no known allergies.
SOCIAL HISTORY: The patient lives with her common law husband and her daughter. Code Status: DNR. Religion. Catholic. She has a past history of heroin use and was enrolled in MMTP program for 12 hours. She reports feeling discouraged from her symptoms and pain.
PHYSICAL EXAMINATION: Blood pressure 120/80, pulse 80, and respirations 14. General Appearance: Mildly obese woman. PERRLA, 3 mm. Oral mucosa moist without lesions. Lungs: Clear. Heart: RRR without murmurs. Abdomen: Somewhat distended, but soft and nontender. There is firmness found in the low abdomen bilaterally. There is erythema in the intertriginous area and vulva, as well as some serous discharge from the vagina. Neurological Exam: Cranial nerves II through XII are grossly intact. There is normal tone. Power is 5-/5. DTRs nonreactive. Sensation intact to fine touch. Mental Status: The patient is alert, fully oriented, normal speech, and thought process. Normal affect.

ASSESSMENT AND PLAN:
1. Carcinoma of the cervix metastatic to the retroperitoneum, bladder, and lung with irritable obstruction and gradual decline in the performance status. Given this, her prognosis is likely to be limited to six months and she will benefit from home hospice care.
2. Pain, which is a combination of somatic nociceptive pain due to the retroperitoneal invasion, as well as a neuropathic component from pelvic and nerve involvement by the surgery as well as radiation therapy and disease itself. We are going to increase methadone to 70 mg every eight hours and continue 30 mg for breakthrough. We will add pregabalin 50 mg three times a day and titrate the dose up as needed.

```
3. Nausea and poor appetite. We will start Megace 200 mg daily.
4. Shortness of breath. We will provide oxygen p.r.n.
5. Candidal infection. We will start clotrimazole 1% cream b.i.d.
6. Constipation. We will advance the bowel regimen to Sorbitol 30 cc three times a
day and Senna-S three tabs twice a day.
7. Psychosocial. The patient is getting discouraged. We will provide supportive
counseling.

Length of the encounter was 80 minutes; more than half spent on exchange of
information. Thank you for the opportunity to participate in the care for this
patient."""

doc = med7(text) # onde o algortimo deve procurar para reconhecer cada entidade
                 # e salvar em uma variável chamada doc

# Criando diferentes cores para os 7 conceitos

col_dict = {}
seven_colours = ['#e6194B', '#3cb44b', '#ffe119', '#ffd8b1', '#f58231',
                 '#f032e6', '#42d4f4'] # cada valor após # e dentro de ''
                                      # representa o código de uma cor
for label, colour in zip(med7.pipe_labels['ner'], seven_colours):
    col_dict[label] = colour

options = {'ents': med7.pipe_labels['ner'], 'colors':col_dict}

# mostar o texto com as entidades reconhecidas e devidamente coloridas
spacy.displacy.render(doc, style='ent', jupyter=True, options=options)
```