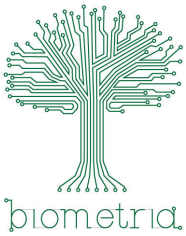
Universidade Estadual Paulista “Júlio de Mesquita Filho”
Instituto de Biociências – Câmpus de Botucatu
Programa de Pós-graduação em Biometria



Uso de Redes Neurais Artificiais para Extração de Dados de Prontuários Médicos

Naila Camila da Rocha

Botucatu
2021

Naila Camila da Rocha

Uso de Redes Neurais Artificiais para Extração de Dados de Prontuários Médicos

Dissertação de Mestrado apresentada ao Curso de Programa de Pós-graduação em Biometria da Universidade Estadual Paulista "Júlio de Mesquita Filho" como parte dos requisitos necessários para a obtenção do título de Mestre em Biometria.

Orientador: Prof(a). Dr(a). Liciana Vaz de Arruda Silveira

Coorientador: Prof. Dr. José Eduardo Corrente

Botucatu
2021

FICHA CATALOGRÁFICA ELABORADA PELA SEÇÃO TÉC. AQUIS. TRATAMENTO DA INFORM.
DIVISÃO TÉCNICA DE BIBLIOTECA E DOCUMENTAÇÃO - CÂMPUS DE BOTUCATU - UNESP
BIBLIOTECÁRIA RESPONSÁVEL: ROSEMEIRE APARECIDA VICENTE-CRB 8/5651

Rocha, Naila Camila da.
Uso de redes neurais artificiais para extração de dados de prontuários médicos / Naila Camila da Rocha. - Botucatu, 2021

Dissertação (mestrado) - Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências de Botucatu
Orientador: Liciana Vaz de Arruda Silveira
Coorientador: José Eduardo Corrente
Capes: 90194000

1. Registros médicos. 2. Análise por agrupamento. 3. Redes neurais (Computação. 4. Distância de Gower.

Palavras-chave: Analise de agrupamentos; Distancia de Gower; Prontuários médicos; Reconhecimento de entidades nomeadas; Redes neurais.

Resumo

Diversos estudos recentes têm utilizado inteligência artificial na extração e tratamento de dados secundários na área da saúde, obtidos em prontuários eletrônicos hospitalares. No entanto, alguns estudos são inviáveis devido a informações incompletas ou inseridas apenas em campos narrativos. O objetivo deste trabalho é desenvolver uma rede neural que utilize os dados desses campos para obter informações estruturadas referentes aos sintomas, diagnósticos, medicamentos, condições, exames e tratamentos. A rede neural proposta facilitará a descoberta de relações entre doenças e sintomas, prevalências e incidências, a identificação de condições clínicas, a evolução de enfermidades e os efeitos das medicações prescritas. O algoritmo utiliza métodos de processamento de linguagem natural para extração de textos e redes neurais convolucionais para reconhecimento de padrões. Foram simulados diferentes valores e funções para a determinação dos hiperparâmetros e otimizadores mais adequados para o modelo de Reconhecimento de Entidades Nomeadas (NER) desenvolvido através da biblioteca *spaCy* em *Python*. Para uma análise exploratória dos dados extraídos e demonstração da aplicabilidade do modelo foram executadas técnicas da estatística multivariada de análise de agrupamento, obtendo quatro grupos que melhor representam os perfis dos pacientes e os medicamentos por eles utilizados. Os resultados obtidos foram significativos considerando a complexidade do modelo, com um *F-Score* de 63,9% e *Precision* de 72,7%. A classe Condição do Paciente chegou a atingir 90,3% de *Precision*, seguido por Medicação com 87,5%. No desenvolvimento do presente trabalho, foram utilizados dados de 30.000 prontuários de pacientes do Hospital das Clínicas da Faculdade de Medicina de Botucatu/SP - Brasil (HCFMB), gerando um *corpus* com 1.200 textos clínicos. A utilização de NER em dados clínicos se mostrou uma ferramenta capaz de extrair informações que não existem em campos estruturados de prontuários médicos. Além disso, análises de agrupamento utilizando esses dados revelam comportamentos e características até então desconhecidas, relacionadas com as Entidades extraídas.

Palavras-chave: Redes Neurais Convolucionais, Análise de Agrupamento, Reconhecimento de Entidades Nomeadas, Prontuários Médicos.

Sumário

1 Introdução

Quando um paciente está em um ambiente hospitalar, o documento que concentra todas as informações acerca da sua saúde, dados sociodemográficos e história pregressa, ou seja, dados clínicos e não clínicos, são os prontuários. Pinto (2006) descreve esses arquivos como a memória escrita da história das condições de saúde de uma pessoa.

Os dados são coletados desde a triagem do indivíduo ao entrar no estabelecimento médico até o relatório padrão de alta hospitalar em sua saída. Entre estes, existem ainda: anamnese, exame clínico, exames complementares, evolução da enfermidade e medicamentos prescritos.

Vários estudos na área da saúde concluem que a análise desses dados podem fornecer informações importantes para a tomada de decisão em diagnósticos médicos (SONG, 2013; GOTH, 2012; KOHANE, 2011; FALCÃO et al., 2009; ZWEIGENBAUM et al., 2007; ANANIADOU; KELL; TSUJII, 2006; SPASIC et al., 2005).

No entanto, estes registros frequentemente incluem dados narrativos não estruturados, o que envolve certa complexidade em sua análise devido ao uso de acrônimos, negação, erros gramaticais, diferenças culturais daqueles que os inseriram no sistema, estilos descritivos distintos, etc. Existem várias situações em que os dados não estão disponíveis ou o preenchimento está incompleto. Eventuais erros humanos de preenchimento ou digitação, bem como a falta de informações por incompatibilidade de sistema em que foram originalmente digitados, também são aspectos relevantes.

Outro ponto importante é a existência de campos de texto livre como “evolução do paciente” em que muitos profissionais preferem preencher neste local informações que deveriam ser preenchidas em campos estruturados específicos, dificultando a obtenção de buscas de dados referenciados. Assim, o desenvolvimento de estudos com a utilização de informações da área da saúde normalmente demandam muito pré-processamento por parte dos pesquisadores e muitas vezes se tornam inviáveis. O sucesso na execução de pesquisas neste tipo de campo facilitaria a identificação de doenças e a busca por características de pacientes para associações, entre outras análises específicas e de rotina.

Algumas pesquisas mostram que o uso combinado de dados estruturados e dados de texto livre têm ajudado pesquisadores a descobrir novas relações entre doenças e sintomas, identificação de condições clínicas e na prescrição de ações preventivas de surtos epidemiológicos (PEISSIG

et al., 2012; ROQUE et al., 2011; PAKHOMOV et al., 2007).

A mineração de texto é uma tecnologia capaz de tornar possível a utilização de dados presentes em texto livre, pois ela permite recuperar informações, extrair dados, resumir documentos, associar regras e realizar análises qualitativas ou quantitativas em documentos de texto. De acordo com Aranha e Passos (2006), a mineração de texto é a aplicação da tecnologia com o objetivo de descobrir conhecimento em texto e nasceu da necessidade de encontrar, de forma automática, informações, padrões e anomalias.

Já as redes neurais profundas, segundo Montavon, Samek e Müller (2018), fazem parte de uma família mais ampla de métodos de aprendizado de máquina, baseados em redes neurais artificiais. É uma ferramenta com diversas aplicações, incluindo o processamento de linguagem natural. Uma rede neural profunda é uma coleção de neurônios organizados em uma sequência de múltiplas camadas, onde os neurônios recebem como entrada as ativações de neurônios da camada anterior, e realizam um cálculo simples seguido por uma ativação não linear. Os neurônios da rede implementam em conjunto um mapeamento não linear complexo da entrada para a saída. Este mapeamento é aprendido a partir dos dados, adaptando os pesos de cada neurônio com a técnica *backpropagation* do erro.

1.1 Objetivos

O objetivo deste estudo foi desenvolver uma rede neural em conjunto com métodos de extração de linguagem natural capaz de obter informações históricas estruturadas dos prontuários médicos de pacientes atendidos no Hospital das Clínicas da Faculdade de Medicina de Botucatu (HCFMB) de modo a facilitar o diagnóstico de surtos epidemiológicos, busca de características específicas que não constam nos campos estruturados da base e melhoria no desempenho de pesquisas de informações no cotidiano do hospital.

Com o intuito de atingir o objetivo proposto, foi necessário:

- Comparar métodos existentes que utilizam linguagem natural para extração de textos e definir o mais adequado para este modelo;
- Escolher os métodos e modelos que serão utilizados para reconhecimento de padrões e agrupamento de dados;
- Desenvolver uma rede neural capaz de extrair e transformar os dados dos prontuários em informações estruturadas;
- Avaliar o desempenho das extrações e classificações obtidas na amostra, ou seja, a qualidade de informação extraída;
- Utilizar métodos de análise multivariada para obter padrões e análisar os dados extraídos;

- Utilizar métodos de validação para a rede neural e para o modelo proposto.

Este trabalho está organizado em cinco capítulos: (i) fundamentação teórica com o estudo dos métodos existentes na literatura para a coleta e classificação dos dados; (ii) metodologia com a apresentação dos dados e determinação do modelo e seus parâmetros; (iii) avaliação e validação do modelo em uma amostra de prontuários médicos; (iv) utilização dos resultados em uma aplicação prática; (v) conclusão com os principais aprendizados e recomendações para pesquisas futuras.

2 Fundamentação Teórica

Neste capítulo foram definidos os conceitos de mineração de texto, linguagem natural, redes neurais e aprendizagem profunda, além de uma visão geral sobre reconhecimento de entidades nomeadas, foco central do trabalho.

Em seguida, foram abordadas as fundamentações teóricas das métricas utilizadas na análise da qualidade dos dados e validação do modelo. Também foi apresentado o método estatístico multivariado de análise de agrupamentos, utilizado para demonstrar a aplicabilidade do modelo.

2.0.1 Mineração de Texto e Linguagem Natural

Ao cadastrar dados não estruturados nos sistemas, utilizando a linguagem natural, surgem fatores que atrapalham o desempenho de buscas automáticas, como é o caso de utilização de siglas, abreviaturas, erros de digitação e vícios de linguagem, entre outros. A mineração de textos é uma tecnologia que permite a descoberta de informações nestes tipos de dados.

O processamento de linguagem natural, segundo os autores Jusoh e Alfawareh (2012), visa à geração e a compreensão da língua natural, considerando análises morfológicas, sintáticas e semânticas.

O uso combinado de mineração de texto aplicado à linguagem natural e redes neurais torna possível a criação de algoritmos capazes de ensinar o computador a fazer esse tipo de busca, recuperação da informação, indexação, extração da informação, associação de documentos, sumarização, clusterização e classificação/categorização.

Assim, Descoberta de Conhecimento em Textos (*Knowledge Discovery in Text - KDT* ou *Text Mining*) pode ser definida, conforme os autores Thuraisingham (2014), Sullivan (2000), Hearst (1999), Tan (1999), Chen et al. (1994), como o processo de extrair padrões ou conhecimento, interessantes e não triviais, a partir de documentos textuais ou ainda uma forma de examinar uma coleção de documentos e descobrir informação não contida em nenhum dos documentos, ou seja, derivar novas e relevantes informações de uma grande coleção de textos. Esta tecnologia também realiza várias funções de busca, análise de dados exploratória, análise linguística e categorização.

É relevante ressaltar que mineração de textos é diferente de um mecanismo de busca, pois

neste último o usuário já sabe o que quer encontrar. A mineração de textos, como mencionado anteriormente, ajuda o usuário a descobrir informações desconhecidas.

Conforme Costa (2010), o processo de descoberta de conhecimento textual é composto das etapas de pré-processamento, processamento (mineração) e pós-processamento (avaliação dos resultados). Todo esse processo foi totalmente originado do modelo *Knowledge Discovery in Databases - KDD*, também conhecido como mineração de dados, onde o processo é baseado em dados estruturados, diferente da mineração de texto, que tem a finalidade de extrair padrões relevantes de dados não estruturados e gerar conhecimento a partir do excesso de informação em diversos âmbitos (MALI; ATIQUE, 2014; THOMAS; MCNAUGHT; ANANIADOU, 2011).

A Mineração de Dados pode ser dividida em tarefas preditivas e descritivas. As tarefas preditivas, conforme Silva, Peres e Boscarioli (2017), Castro e Ferrari (2016), Braga (2005), utilizam de valores de atributos descritivos para tentar predizer valores futuros ou desconhecidos, já às tarefas descritivas, têm o objetivo de encontrar padrões que podem descrever os dados.

Existem vários *softwares* de Mineração de Dados no mercado, tanto comerciais quanto de código aberto. Os *softwares* comerciais são tecnologias fornecidas por empresas como: *Microsoft*, SAS, IBM e *Oracle*. Existem também ferramentas de código aberto como R, *Python*, *Weka* e *Orange*.

2.0.2 Redes Neurais e Aprendizagem Profunda

De acordo com Goodfellow et al. (2016), Loh e Garin (2001), os algoritmos de *Machine Learning* possuem a capacidade de adquirir seu próprio conhecimento, extraindo padrões de dados brutos. Peres (2017) caracteriza como um conjunto de técnicas utilizadas para a aquisição de conhecimento pelo sistema.

As redes neurais artificiais são um modelo computacional que tem o seu funcionamento inspirado no cérebro humano, projetado para definir o melhor caminho para a realização de cada tarefa, com base em uma associação de informações conhecidas. Esta técnica trabalha como classificadora de padrões de natureza estatística, nas quais as classes definidas são representadas por pontos em um espaço de decisão multidimensional, segundo Haykin (2009).

Técnicas de Aprendizagem Profunda *(Deep Learning)*, também conhecidas como Redes Neurais Profundas *(Deep Neural Network)*, consistem em um tipo particular de *Machine Learning* e devido à utilização de muitas camadas de processamento de informações não lineares ocultas, permitem aprender uma sequência de funções que realizam a transformação de vetores, mapeando-os de um espaço a outro, até que se atinja o resultado pretendido (PONTI; COSTA, 2018).

O aprendizado acontece com a alteração da ponderação inicial (sinapses) por meio de algoritmos que, de acordo com Peres (2017), podem ser de forma supervisionada (agente externo indica a resposta desejada para os padrões de entrada) ou não supervisionada (não existe um agente externo indicando a resposta desejada para os padrões de entrada). Também existe a

aprendizagem por reforço (agente externo avalia a resposta final da rede), considerado por Haykin (2009) como uma subcategoria do aprendizado não supervisionado.

Schmidhuber (2015), Deng e Yu (2014), Baldi (2012) argumentam que redes neurais profundas podem ser aplicadas a problemas de classificação, predição, reconhecimento facial, detecção de objetos em imagens e extração de características, entre outros.

Por muito tempo pesquisadores tiveram dificuldade em desenvolver redes neurais com mais de três camadas, devido à falta de algoritmos eficientes para o treinamento das mesmas. Hinton e Osindero (2006) possibilitaram o desenvolvimento de redes com muitas camadas ocultas após criarem um método de treinamento prévio não supervisionado.

Sabe-se que quanto maior a quantidade de camadas, maior é a dificuldade de treinamento, mas melhor a capacidade de aprendizado. (ZHOU; GREENSPAN; SHEN, 2017) mencionam que a quantidade de camadas ocultas influencia em quão ajustada pode ficar a rede, especialmente em problemas não linearmente separáveis, pois quanto maior a quantidade de camadas, melhor será a diferenciação para a classificação de um dado entre as classes.

Redes Neurais Profundas são compostas pela camada de entrada, camadas intermediárias ou ocultas e a camada de saída. A camada de entrada é onde as unidades recebem os padrões. Nesta camada deve existir uma unidade especial conhecida como *bias*, usada para aumentar os graus de liberdade, permitindo uma melhor adaptação, por parte da rede neural, ao conhecimento a ela fornecido. Nas camadas intermediárias são feitos os processamentos e as extrações de características. Já na camada de saída, o resultado final é concluído e apresentado. Assim, o número de camadas define a capacidade de representação das relações entre o espaço de entrada e o de saída. A inexistência da camada intermediária condiciona o modelo a representar bem somente relações linearmente independentes, característica do *Perceptron*.

Perceptron foi o primeiro neurônio artificial, desenvolvido por Frank Rosenblatt. Esse neurônio possuía a capacidade de receber informações, processar e enviar um resultado *booleano* para a próxima camada de neurônios conectada a ele. Atualmente, o modelo de neurônio mais utilizado é o *sigmoidal*, que possui um funcionamento similar, com a diferença de produzir uma saída *sigmoidal* ao invés de uma saída *booleana* (NIELSEN, 2015).

Bishop e Looney (1998) definem que as redes sem realimentação *(feedforward)* têm neurônios agrupados em camadas e o sinal percorre a rede em uma única direção, da entrada para a saída. Os neurônios da mesma camada não são conectados e cada conexão entre os neurônios tem um peso numérico.

Nas redes com realimentação ou recorrentes *(recurrent)*, a saída de alguns neurônios alimentam neurônios da mesma camada (inclusive o próprio) ou de camadas anteriores. O sinal percorre a rede em duas direções, tem memória dinâmica e capacidade de representar estados em sistemas dinâmicos. Pode-se citar a rede de Hopfield como um exemplo de rede com realimentação.

Entre as técnicas de Aprendizado Profundo presentes na literatura, destacam-se as técnicas baseadas em *Autoencoders*, as Redes Neurais Convolucionais e as Máquinas Restritas de Boltzmman *(Restricted Boltzmann Machines)*, entre outras. Segundo Goodfellow et al. (2016), Deng e Yu (2014), estas técnicas abrangem uma ampla possibilidade de atividades, incluindo visão computacional, processamento de áudio e fala, processamento de linguagem natural, robótica, bioinformática e finanças.

O sucesso desses algoritmos em várias áreas vem fazendo com que grupos de pesquisadores publiquem códigos, ferramentas e até mesmo modelos completos já treinados para algumas aplicações. Existem diversos *frameworks* disponíveis para a implementação de redes neurais e utilização de métodos de aprendizado de máquina. Dentre os principais estão o *Theano, TensorFlow, Torch, Caffe, MXNet e Neon* (ZHOU; GREENSPAN; SHEN, 2017).

Jones (2017) cita algumas arquiteturas de Aprendizado Profundo que se tornaram bastante populares: Redes Neurais Recorrentes *(recurrent neural networks)*, Memória de Longo e Curto Prazo *(long short-term memory)*, Unidade Recorrente Bloqueada *(gated recurrentunit)*, Redes Neurais Convolucionais *(convolutional neural networks)*, Redes de Crenças Profundas *(deepbelief networks)* e Redes de Empilhamento Profundo *(deepstacking networks)*.

As redes neurais artificiais profundas estão ganhando espaço na área médica, tanto na análise de textos em bancos de dados, como para ajudar na classificação de imagens (tomografias, exames de fundo de olho, eletrocardiogramas). Essas análises feitas por inteligências artificiais vêm ajudado na realização de diagnósticos de doenças e nas decisões quanto o tratamento mais adequado para cada situação.

2.0.3 Reconhecimento de Entidades Nomeadas (REN)

O *Named Entity Recognition (NER)*, também conhecido em português como Reconhecimento de Entidades Nomeadas (REN) ou Identificação de Entidade é uma aplicação de Processamento de Linguagem Natural (PNL). Sua função é processar grandes quantidades de textos livres para identificar e classificar automaticamente entidades nomeadas, como locais, horários e pessoas em documentos de texto. O grande desafio é conseguir diferenciar as entidades conforme o contexto e domínio considerado.

Como publicado no AIHub (2020), o NER pode ser implementado com métodos estatísticos ou baseados em regras, os quais requerem uma grande quantidade de dados de treinamento rotulados e são normalmente treinados de maneira supervisionada ou semi-supervisionada. As abordagens estatísticas para NER incluem Modelos Ocultos de Markov (HMM), Máxima Entropia (ME) e Campos Aleatórios Condicionais (CRF), bem como abordagens de Aprendizado Profundo com Redes Neurais Recorrentes que aprendem a identificar e classificar Entidades Nomeadas (NEs) a partir de um *Corpus* de treinamento. *Corpus* é uma coleção de resumos de características anotadas manualmente.

Li et al. (2020) definem formalmente NER como: dada uma sequência de *tokens* $s = (w_1, w_2, \ldots, w_N)$, NER é produzir uma lista de tuplas (I_s, I_e, t), cada uma das quais é uma entidade nomeada mencionada em s. Desta forma, $I_s \in [1, N]$ e $I_e \in [1, N]$ são os índices inicial e final de uma menção de entidade nomeada; t é o tipo de entidade de um conjunto de categorias predefinidas.

Existem vários modelos prontos e treinados de NER escritos majoritariamente em Java, mas sua grande maioria está disponível apenas na língua inglesa. Nos últimos anos surgiram trabalhos que exploram este assunto na lingua portuguesa, como os trabalhos desenvolvidos por Schneider et al. (2020), Peters et al. (2020), Lopes, Teixeira e Oliveira (2019).

O *spaCy* é uma biblioteca de código aberto para PNL escrita em *Python* e *Cython*. É um dos ambientes mais utilizados para construção de modelos NER, possui como arquitetura Redes Neurais Convolucionais Residuais (CNN) e análise incremental com *embeddings* de *Bloom* para NER. O NER estatístico do OpenNLP depende do Máxima Entropia (ME) e o GATE conta com Modelos Ocultos de Markov (HMM). O DBPedia *Spotlight* executa NER com correspondência de *substring* usando o algoritmo Aho-Corasick (AIHUB, 2020).

Nome	Linguagem	Licença	Método
DBPedia	Java	Apache	Baseado em Regras
GATE	Java	LGPL	Modelos Ocultos de Markov
NLTK	Python	Apache	Máxima Entropia
OpenNLP	Java	Apache	Máxima Entropia
SpaCy	Python	MIT	Redes Neurais Convolucionais
Stanford CoreNLP	Java	GPL	Campos Aleatórios Condicionais

Tabela 1. Métodos e linguagens utilizadas na construção de modelos de NER. Elaborada pela própria autora com base nas informações disponibilizadas por (AIHUB, 2020; VYCHEGZHANIN; KOTELNIKOV, 2019).

Conforme AIHub (2020), a arquitetura do *spaCy* utiliza filtros convolucionais 1D que são aplicados sobre o texto de entrada para prever como as próximas palavras podem alterar as entidade atuais. As próximas palavras podem alterar, reduzir ou gerar a entidade atual. A sequência de entrada é incorporada a *embeddings bloom*, que modelam os caracteres, prefixo, sufixo e classe gramatical de cada palavra. Os blocos residuais são usados para os CNNs e os tamanhos dos filtros são escolhidos com a pesquisa de feixe.

Foram publicados alguns estudos recentes de comparação de ferramentas para o Reconhecimento de Entidades Nomeadas, o *spaCy* obteve melhores resultados nos casos de NER's personalizadas, ou seja, com entidades criadas e treinadas pelo próprio usuário.

No estudo de Shelar et al. (2020) *"Named Entity Recognition Approaches and Their*

Comparison for Custom NER Model" o *spaCy* obteve um melhor resultado para o modelo NER personalizado em comparação com *Apache OpenNLP* e *TensorFlow* quando se trata de identificar entidades no texto de entrada.

Dawar, Samuel e Alvarado (2019) também publicaram um artigo que comparava o sistema de treinamento, *design* e arquitetura de vários modelos NER, incluindo *spaCy*, *Stanford's Named Entity Recognizer* e *IBM Bluemix's Natural Language Understanding*. A conclusão é que o *spaCy* tem um reconhecimento significativamente melhor e preciso, agregando novas informações que podem enriquecer a base de dados.

2.0.3.1 Modelos Ocultos de Markov

Os Modelos Ocultos de Markov ou *Hidden Markov Model (HMM)* são usados para relacionar uma sequência de observações a uma sequência de estados ocultos, ou seja, relacionar uma sequência de palavras de um rótulo de atividade a uma sequência de componentes semânticos correspondentes, segundo os autores Leopold et al. (2019), Wang, Deng e Acero (2005), Rabiner e Juang (1986). Para isso, é necessário determinar a probabilidade de todas as possíveis sequências de estados ocultos e selecionar o mais provável.

Conforme definido por Leopold et al. (2019), dada uma sequência de observação O, a probabilidade de uma sequência de estado oculto particular Q, ou seja, probabilidade $P(O, Q)$, é calculado da seguinte forma: $P(O, Q) = P(Q) \times P(O|Q)$, ou seja a multiplicação da probabilidade de encontrar a sequência particular de estados ocultos Q, dado como $P(Q)$ e a probabilidade de encontrar as observações de O que estão associados aos estados ocultos específicos de Q, dado como $P(O|Q)$.

Leopold et al. (2019) definem que $P(Q)$ é calculada como a probabilidade Markoviana de que uma sequência de estados ocultos que começa com um determinado estado no índice 0, ou seja, o estado q_0, multiplicado pela probabilidade de que cada estado subsequente q_i segue seu estado precedente q_{i-1}, ou seja, $P(Q) = P(q_0) \times \prod_{i=1}^{n} P(q_i|q_{i-1})$, onde $n = |O|$. $P(O|Q)$ é calculada como $\prod_{i=1}^{n} P(o_i|q_i)$, onde $P(o_i|q_i)$ que denota a probabilidade de que a observação o_i corresponde ao estado q_i.

As probabilidades usadas para calcular $P(Q)$ derivam da matriz de transição (captura as probabilidades de se mover de um estado para outro, ou seja, as probabilidades de transição entre entidades) e as probabilidades associadas com $P(O|Q)$ da matriz de probabilidade de emissão/observação (captura a probabilidade de um observação ocorrer em um determinado estado), segundo Wang, Deng e Acero (2005).

2.0.3.2 Máxima Entropia

A estrutura de Máxima Entropia ou *Maximum Entropy (ME)*, conforme os autores Jurafsky e Martin (2018), Chieu e Ng (2003), Ratnaparkhi (1998), estima as probabilidades com base

no princípio de fazer tão poucas suposições quanto possível, além das restrições impostas. Essas restrições são derivadas dos dados de treinamento, expressando algumas relação entre características e resultados. A distribuição de probabilidade que satisfaz a propriedade acima é a aquela com a entropia mais alta.

A entropia pode ser definida como única, ou seja, mede a quantidade de informação em uma variável aleatória, concorda com a distribuição de máxima verossimilhança e tem a forma exponencial:

$P(o|h) = \frac{1}{Z(h)} \prod_{j=1}^{k} \alpha_j^{f_j(h,o)}$, onde o se refere ao resultado, h a história (ou contexto), e $Z(h)$ é uma função de normalização (CHIEU; NG, 2003).

As funções características usadas na estrutura de Máxima Entropia são binárias. Um exemplo de função característica é:

$$f_j(h,o) = \begin{cases} 1, & \text{se } o = \text{resultado (entidade), h = contexto} \\ 0, & \text{caso contrário} \end{cases}$$

De acordo com Jurafsky e Martin (2018), Ratnaparkhi (1998), o intuito é construir um modelo de distribuição de probabilidade que se aproxime da distribuição de probabilidade empírica obtida através do processo de treinamento do modelo.

O procedimento de estimação de Máxima Entropia combina evidências obtidas no treinamento utilizando um modelo log-linear e produz um modelo em que toda função de características f_j está relacionada com o parâmetro α_j. O parâmetro α_j pode ser interpretado como pesos para a função f_j correspondente. Os parâmetros α_j são estimados por um procedimento chamado *Generalized Iterative Scaling (GIS)*. Chieu e Ng (2003) afirmam que este é um procedimento iterativo que melhora a estimativa dos parâmetros em cada iteração.

Durante o teste, é possível que o classificador atribua uma sequência inadimissível de palavras a uma entidade. Para eliminar tais sequências, definimos um probabilidade de transição entre classes de palavras $P(c_i|c_j)$ ser igual a 1, se a sequência for admissível e 0, caso contrário. A probabilidade das entidades $c_1, ..., c_n$ atribuídas às palavras em uma frase s em um documento D é definido como segue:

$P(c_1, ..., c_n|s, D) = \prod_{i=1}^{n} P(c_i|s, D) \times P(c_i|P(c_{i-1})$, onde $P(c_i|s, D)$ é determinado pelo classificador de Máxima Entropia. Chieu e Ng (2003) citam que o algoritmo de Viterbi é usado para selecionar a sequência de entidades com a maior probabilidade.

2.0.3.3 Campos Aleatórios Condicionais

Os Campos Aleatórios Condicionais ou *Conditional Random Fields (CRF)* é definido por Settles (2004), Lafferty, McCallum e Pereira (2001) como modelos gráficos estatísticos

não direcionados usados para calcular a probabilidade condicional de valores em nós de saída determinados, baseado em valores atribuídos a outros nós de entrada determinados.

A estrutura mais comum de dependências entre as variáveis é apresentada em um CRF de cadeia linear que representa essas dependências em uma sequência temporal, desta forma prediz as variáveis de saída como uma sequência. Neste tipo de modelo, os CRFs fazem uma suposição de independência de Markov de primeira ordem e, portanto, podem ser entendidos como máquinas de estados finita ou autômato finito condicionalmente treinados, conforme os trabalhos de Settles (2004), McCallum e Li (2003).

Seja $O = (o_1, o_2, ..., o_n)$ uma sequência de palavras observadas de comprimento n e S um conjunto de estados em uma máquina de estados finitos, cada um correspondendo a um rótulo $l \in L$. Seja $S = (s_1, s_2, ..., s_n)$ a sequência de estados em S que correspondem aos rótulos atribuídos às palavras na sequência de entrada O.

Os Campos Aleatórios Condicionais (CRF), de acordo com Settles (2004), definem a probabilidade condicional de uma sequência de estado dada uma sequência de entrada, ou seja, $P(s|o) = \frac{1}{Z_o} \exp(\sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_j f_j(s_{i-1}, s_i, o, i))$, onde Z_o é um fator de normalização de todos as sequências de estados, $f_j(s_{i-1}, s_i, o, i)$ é uma de m funções que descreve uma característica, e λ_j um vetor de pesos que deve ser estimado a partir de um conjunto de treino.

2.0.3.4 Redes Neurais Convolucionais

As Redes Neurais Convolucionais ou *Convolutional Neural Network - CNN* recebem este nome porque possuem camadas de convolução que, juntamente com outros tipos de camadas, determinam uma arquitetura específica, segundo Kulkarni et al. (2015). Elas possuem campos receptivos locais, pesos compartilhados e *pooling*.

A convolução foi concebida para que as redes neurais pudessem aprender diferentes níveis de características, observando os dados de entrada do modelo em mais de uma dimensão. As camadas de convolução podem ser entendidas como conjunto de filtros/*kernel* a serem aprendidos, representados em forma de matrizes de ativação. Maia et al. (2018) explicam que cada camada de convolução tem como objetivo extrair um conjunto de características da camada imediatamente anterior a ela.

Uma CNN realiza o aprendizado por meio de métodos *feedward e backpropagation*. Normalmente existem as camadas de convolução, *pooling*, ativação e de saída *fully connected*.

2.0.3.4.1 Redes Neurais Convolucionais em Processamento de Linguagem Natural (PLN)

As Redes Neurais Convolucionais vem alcançando resultados notáveis quando aplicadas em problemas de Processamento de Linguagem Natural (PLN). Alguns exemplos de estudos são citados por Zhang e Wallace (2015).

Para utilizar CNN em análises textuais é necessário expressar os dados em forma de vetores. Existem diversas formas de fazer essa transformação, como utilizar codificadores (*encoders*) junto com operações 1D e camadas do tipo *embedding*. Bonnin (2016) define *encoders* como a indexação de cada palavra de acordo com sua frequência na base de dados.

2.0.3.4.2 Camadas do tipo *Embedding*

Camadas do tipo *embedding* relacionam palavras de acordo com sua proximidade semântica, como definido por Bonnin (2016). Essas camadas realizam a transformação linear $W : p \rightarrow \mathbb{R}^n$, sendo que p é a palavra sofrendo transformação W para um espaço de dimensão n. A função W possui parâmetros θ que é uma matriz mapeando cada palavra p, tal que $W_\theta p_{(n)} = \theta_n$. A medida que ocorre o aprendizado, W se ajusta.

2.0.3.4.3 Arquitetura e Funcionamento de uma Rede Neural Convolucional para a Classificação de Texto

Zhang e Wallace (2015) apresentam toda a arquitetura e funcionamento de uma Rede Neural Convolucional para classificação de texto. A figura 1 exibe as várias camadas de processamento que é feito pela rede, começando pelo recebimento com a frase *tokenizada* que será convertida para uma matriz da frase, as linhas serão representações de vetores de palavras de cada *token*. Todo este processo é feito por *Word Embeddings*.

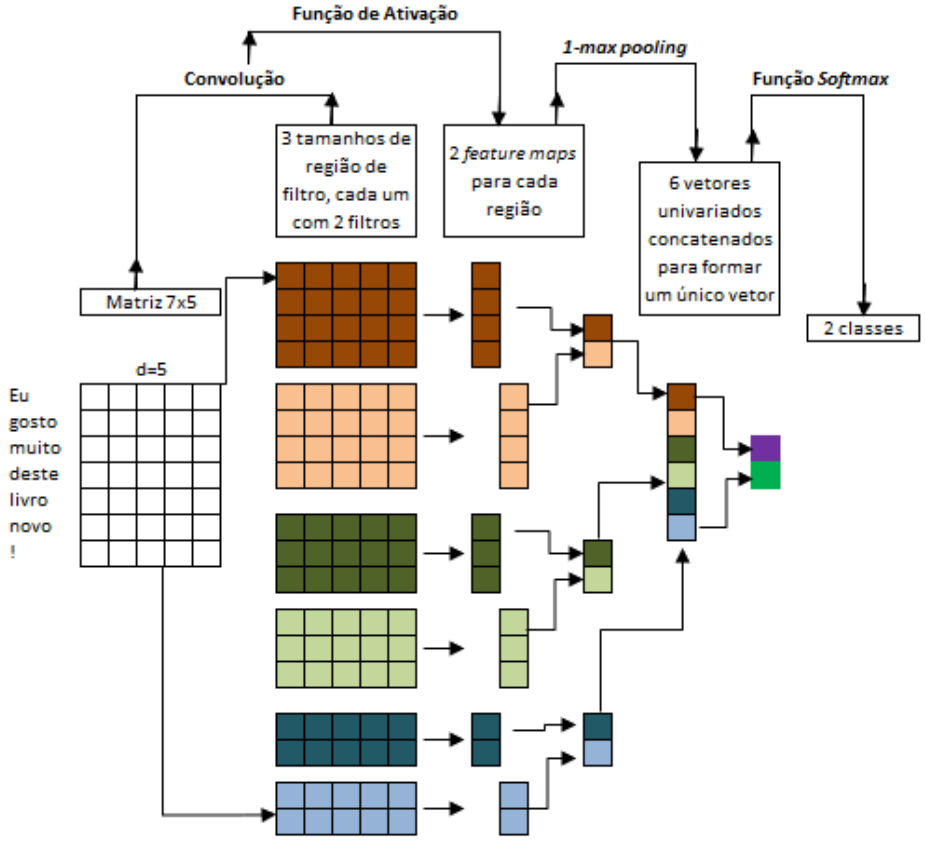


Figura 1. Arquitetura CNN para classificação de texto. Fonte: (ZHANG; WALLACE, 2015), modificada pela autora.

Na figura acima Zhang e Wallace (2015) ilustram três tamanhos de região de filtro 2, 3 e 4, cada um com 2 filtros. Os filtros realizam convoluções na matriz da frase e geram *feature map*. O agrupamento 1-max é realizado em cada mapa, ou seja, o maior número de cada *feature map* é registrado. Assim, um vetor de característica univariada é gerado a partir de todos os seis mapas, e estas 6 características são concatenadas para formar um vetor de características para a penúltima camada. A camada final de *softmax* em seguida, recebe esse vetor de características como entrada e o usa para classificar a frase.

A dimensionalidade dos vetores de palavras é denotado por d. Se o comprimento de uma determinada frase é s, então a dimensionalidade da matriz da frase é $s \times d^2$. Posteriormente, a matriz da frase pode ser tratada semelhante a uma imagem, podendo ser realizada a convolução por meio de filtros lineares. Em texto é razoável usar filtros com larguras iguais à dimensionalidade do vetor de palavras d, conforme explanam Zhang e Wallace (2015).

A dimensionalidade do *feature map* gerado por cada filtro irá variar em função do comprimento da frase e o tamanho da região do filtro. Uma *pooling function* é, portanto, aplicada a cada *feature map* para induzir um vetor de comprimento fixo.

Desta forma, uma CNN exige a especificação da arquitetura de modelo, a definição dos hiperparâmetros de acompanhamento, incluindo o tamanho da região do filtro e os parâmetros de regularização. Na realidade requer, no mínimo, a especificação dos vetores de palavras de entrada, tamanho da região do filtro, o número de *features maps*, a *função de ativação*, a estratégia de *pooling*, e a definição do *dropout*.

2.0.3.4.4 Camada do tipo *Pooling*

As camadas de convolução são seguidas de uma camada de *pooling*, também conhecidas como seleção ou agrupamento. Zhang e Wallace (2015), Boureau, Ponce e LeCun (2010) esclarecem que camadas deste tipo tem como objetivo simplificar e representar estatisticamente a camada anterior. A quantidade de dados é significamente reduzida sem grandes perdas. Os valores resultantes da operação de *pooling* são armazenados em outro *feature map*. Um procedimento comum para o *pooling* é o *1-max pooling*, no qual, uma unidade de *pooling* simplesmente gera a ativação máxima na região de entrada, ou seja, apenas o maior número da unidade é passado para a saída.

2.0.3.4.5 Funções de Ativação

A função de ativação tem como objetivo mapear a entrada em uma distribuição conhecida, como exposto por Krizhevsky, Sutskever e Hinton (2017), Zhang e Wallace (2015), Pei, Ge e

Chang (2015), Chen e Manning (2014), Maas, Hannun e Ng (2013). Algumas utilizadas são: linear, *Rectified Linear Unit (ReLu)*, *Leaky ReLU*, ELU (Unidade Linear Exponencial), Tangente Hiperbólica (TanH), *SoftPlus*, Sigmóide e a *Softmax*. Também é possível não aplicar nenhuma função de ativação, isso indica que para alguns conjuntos de dados uma transformação linear é suficiente para capturar a correlação entre a palavra *embedding* e o rótulo de saída.

As funções de ativação precisam ter derivada e ser uma função monotônica, segundo Goodfellow et al. (2016). As funções não lineares se sobressaem em relação às funções lineares, pois todas as funções não lineares são diferenciáveis e o empilhamento da rede também é possível, o que ajuda na criação das redes neurais profundas. As funções de ativação não lineares são divididas principalmente com base em seu intervalo ou curvas. Abaixo a definição de algumas funções de ativação existentes:

1. Sigmóide/Logística - definida por Goodfellow et al. (2016), Gupta (2017), como gradiente suave que permite previsões claras e os valores de saída estão entre 0 e 1, normalizando a saída de cada neurônio. No entanto as saídas são não centradas em zero e para valores muito altos ou muito baixos de X, quase não há alteração na previsão, causando um problema de gradiente de fuga ou desaparecimento (*vanishing gradient*). Isso pode resultar na recusa da rede em aprender mais ou em ser muito lenta para chegar a uma previsão precisa. Este método tem um alto custo computacional.

   Equação: $f(x) = \frac{1}{1+\exp(-x)}$
   Intervalo: $[0, 1]$
   Derivada: $f'(x) = f(x)(1 - f(x))$

2. Tangente Hiperbólica/TanH - segundo conceitos apresentados por Goodfellow et al. (2016), Gupta (2017), é centrado em zero, o que torna mais fácil modelar entradas com valores fortemente negativos, neutros e fortemente positivos. Tem as mesmas desvantagens da função Sigmóide em relação ao gradiente de fuga e o alto custo computacional.

   Equação: $f(x) = \frac{2}{1+\exp(-2x)} - 1$
   Intervalo: $[-1, 1]$
   Derivada: $f'(x) = 1 - f(x)^2$

3. ReLU (*Unidade Linear Retificada*) - é computacionalmente eficiente, segundo Goodfellow et al. (2016), Ramachandran, Zoph e Le (2017), Gupta (2017), pois permite uma convergência rápida do modelo e permite o *backpropagation*. O problema é o *ReLU dying* - quando as entradas se aproximam de zero ou são negativas, o gradiente da função torna-se

zero, a rede não pode realizar *backpropagation* e não pode aprender.

Equação: $f(x) = \begin{cases} 0, & \text{se } x < 0 \\ x, & \text{se } x \geq 0 \end{cases}$

Intervalo: $[0, +\infty]$

Derivada: $f'(x) = \begin{cases} 0, & \text{se } x < 0 \\ 1, & \text{se } x \geq 0 \end{cases}$

4. *Leaky ReLU* - de acordo com Ramachandran, Zoph e Le (2017), Gupta (2017), Goodfellow et al. (2016), é uma variação da ReLU que tem uma pequena inclinação positiva na área negativa, então ela permite o *backpropagation*, mesmo para valores de entrada negativos e previne o problema *dying ReLU*. No entanto, apresenta resultados não consistentes para valores de entrada negativos.

   Equação: $f(x) = \begin{cases} \alpha x, & \text{se } x < 0 \\ x, & \text{se } x \geq 0 \end{cases}$

   Intervalo: $[-\infty, +\infty]$

   Derivada: $f'(x) = \begin{cases} \alpha, & \text{se } x < 0 \\ 1, & \text{se } x \geq 0 \end{cases}$

5. ReLU paramétrico - Ramachandran, Zoph e Le (2017), Gupta (2017), Goodfellow et al. (2016) descrevem que ela permite que a inclinação negativa seja aprendida. Esta função fornece a inclinação da parte negativa da função como um argumento. Portanto, é possível realizar o *backpropagation* e aprender o valor mais adequado de α. No entanto, não tem consistência na resolução de problemas diferentes.

   Equação: $f(\alpha, x) = \begin{cases} \alpha x, & \text{se } x < 0 \\ x, & \text{se } x \geq 0 \end{cases}$

   Intervalo: $[-\infty, +\infty]$

   Derivada: $f'(\alpha, x) = \begin{cases} \alpha, & \text{se } x < 0 \\ 1, & \text{se } x \geq 0 \end{cases}$

6. *Softmax* - é capaz de lidar com várias classes, segundo Ramachandran, Zoph e Le (2017), Gupta (2017), Goodfellow et al. (2016). Normaliza as saídas para cada classe entre 0 e

1 e divide por sua soma, dando a probabilidade do valor de entrada estar em uma classe específica. Normalmente *Softmax* é usado apenas para a camada de saída, para redes neurais que precisam classificar entradas em várias categorias.

Equação: $f_i(\vec{x}) = \frac{\exp(x_i)}{\sum_{j=1}^{J} \exp(x_j)}$, para $i = 1, ..., J$

Intervalo: $[0, 1]$

Derivada: $\frac{\partial f_i(\vec{x})}{\partial x_j} = f_i(\vec{x})(\delta_{ij} - f_j(\vec{x}))$

7. Função de ativação *Swish* - é uma nova função de ativação autoguiada. Ramachandran, Zoph e Le (2017) citam que ela tem um desempenho melhor do que o ReLU, com um nível semelhante de eficiência computacional. Ao contrário das funções Sigmóide e Tanh, *Swish* é ilimitado acima, o que o torna útil próximo aos gradientes com valores próximos a 0, evitando a saturação conforme o treinamento se torna lento perto de valores de gradiente 0. A suavidade desempenha um papel importante na generalização e otimização. Ao contrário de ReLU, *Swish* é uma função suave que a torna menos sensível à inicialização de pesos e taxa de aprendizagem. Também é limitado abaixo, o que ajuda em fortes efeitos de regularização. Como ReLU e *Softplus*, *Swish* produz saídas negativas para pequenas entradas negativas devido à sua não monotonicidade. A não monotonicidade de *Swish* aumenta a expressividade e melhora o fluxo de gradiente, o que é importante considerando que muitas pré-ativações se enquadram nessa faixa.

   Equação: $\sigma(x) = \frac{x}{1+\exp(-x)}$

8. ELU (*Unidade Linear Exponencial*) - também é proposto para resolver o problema de *dying neurons*. Computacionalmente intensivo e de convergência lenta, devido à função exponencial. Ramachandran, Zoph e Le (2017), Goodfellow et al. (2016) a define como semelhante ao *Leaky ReLU*, embora teoricamente melhor do que ReLU, mas não há boas evidências que na prática que ELU é sempre melhor do que ReLU. $f(x)$ é monotônico apenas se α for maior ou igual a 0. A derivada $f'(x)$ de ELU é monotônica apenas se α estiver entre 0 e 1.

   Equação: $f(\alpha, x) = \begin{cases} \alpha(\exp(x) - 1), & \text{se } x \leq 0 \\ x, & \text{se } x > 0 \end{cases}$

   Intervalo: $[-\alpha, +\infty]$

   Derivada: $f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha, & \text{se } x \leq 0 \\ 1, & \text{se } x > 0 \end{cases}$

9. *Softplus* - é muito semelhante ao ReLU, exceto perto de 0, onde o *Softplus* é mais suave e diferenciável. É muito mais fácil e eficiente calcular ReLU e sua derivada do que para a função *Softplus* que tem ln (.) e exp (.) Em sua formulação, a derivada da função *Softplus* é a função logística, como apresentado por Ramachandran, Zoph e Le (2017), Gupta (2017), Goodfellow et al. (2016).

   Equação: $f(x) = \ln(1 + \exp(x))$

   Derivada: $f'(x) = \frac{1}{(1+\exp(x))}$

10. *Maxout* - é uma generalização das funções ReLU e *Leaky ReLU*, segundo Gupta (2017), Goodfellow et al. (2016). É uma função de ativação que pode ser aprendida. Uma unidade *Maxout* pode aprender uma função convexa linear por partes com até k peças.

    Equação: $max(w_1^T x + b_1, w_2^T x + b_2)$

2.0.3.4.6 Funções de Custo, Funções de Perdas ou Funções de Erro

As Funções de Custo, Funções de Perdas ou Funções de Erro são medidas de erro entre o valor previsto e o valor real. Geralmente as funções de custo e perda são tratadas como sinônimos, mas a função de custo pode conter termos de regularização além da função de perda (RASCHKA; MIRJALILI, 2017; RUDER, 2016).

A função de perda (ou erro) é para um único exemplo de treinamento, enquanto a função de custo (J) considera todo o conjunto de treinamento, ou seja, mede o desempenho de um modelo de aprendizado de máquina para todos os dados fornecidos. Bonnin (2016), Raschka e Mirjalili (2017) especificam que alguns tipos de função de perda são: função de perda de regressão (erro quadrático médio, erro logarítmico médio quadrático, erro médio absoluto), funções de perda de classificação binária (entropia cruzada binária, perda de articulação (*Hinge Loss*), perda de articulação quadrada (*Squared Hinge Loss*)) e funções de perda de classificação multiclasse (perda de entropia cruzada multiclasse, perda de entropia cruzada multiclasse esparsa, perda de divergência de *Kullback Leibler*).

Abaixo as definições de algumas funções de perdas existentes:

1. Erro Médio Absoluto (MAE): é uma média de diferenças absolutas entre as previsões e resultados esperados, onde todos os desvios individuais têm importância uniforme.

   $MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$

Em que i é o índice da amostra, $\hat{y}$ é o valor previsto, y é o valor esperado e n é o número de amostras no conjunto de dados.

2. Erro Quadrático Médio (MSE): é a diferença quadrática média entre as previsões e os resultados esperados.

   $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

3. Raiz quadrada do erro médio (RMSE): é a média da raiz quadrada da soma das diferenças quadradas entre as previsões e as observações reais.

   $RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$

4. Função de perda de entropia cruzada binária: mede a distância do valor verdadeiro (que é 0 ou 1) da previsão para cada uma das classes e, em seguida, calcula a média desses erros de classe para obter a perda final. É a diferença entre duas distribuições de probabilidade p e q, onde p é o valor verdadeiro e q é a estimativa. A entropia cruzada funcionará melhor quando os dados forem normalizados (forçados entre 0 e 1).

   $H(x) = \sum_{i=1}^{N} p(x) \ln q(x)$

5. Entropia cruzada categórica: É uma função de perda usada para categorização de rótulo único. Isso ocorre quando apenas uma categoria é aplicável para cada ponto de dados. Em outras palavras, um exemplo pode pertencer a apenas uma classe.

   $L(y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{ij} \times \ln(\hat{y}_{ij}))$

6. Perda de entropia cruzada multiclasse: Na maioria das vezes, a perda de entropia cruzada multiclasse é usada como uma função de perda. A forma generalizada de perda de entropia cruzada é a perda de entropia cruzada multiclasse.

   $J(\theta) = -\sum_{c=1}^{M} y_{o,c} \ln (p_{o,c})$

   M é o nº de classes e y é o indicador binário (0 ou 1), se o rótulo de classe c for a classificação correta para a entrada o. Já p é a pontuação de probabilidade prevista da entrada o pertencente à classe c.

7. Perda de Divergência *Kullback-Liebler* (KL-Divergência): *KL Divergence* é uma medida de como uma probabilidade de uma distribuição difere de outra distribuição.

   $Dkl(P||Q)$ é interpretado como o ganho de informação quando a distribuição Q é usada em vez da distribuição P.

$Dkl(Q||P)$ é interpretado como o ganho de informação quando a distribuição P é usada em vez da distribuição Q.

O ganho de informação (IG) mede quanta informação um recurso nos dá sobre a classe.

$D_{kl}(P||Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)}$

$D_{kl}(P||Q) = \sum_{x \in X} P(x) \ln (\frac{P(x)}{Q(x)})$

$D_{kl}(P||Q) = \int P(x) \ln \frac{P(x)}{Q(x)}$

A Divergência KL também é conhecida como "entropia relativa". $Dkl(P||Q)$ é lido como Divergência de Q para P.

2.0.3.4.7 Taxa de Aprendizado - *Learning Rate*

Com a taxa de aprendizado é possível determinar os melhores pesos e vieses para melhorar o desempenho da rede neural, conforme Smith (2017), Ruder (2016), Zeiler (2012). Eles iniciam com valores aleatórios e é preciso alterá-los para minimizar a função de perda, ou seja, de acordo com a derivada estes valores são alterados na direção ao mínimo local. Quanto menor a taxa de aprendizado, mais precisa a rede será para atingir o mínimo local. No entanto, as taxas de aprendizado pequenas exigem muitas etapas para atingir o mínimo, o que leva muito tempo. Por outro lado, taxas de aprendizagem maiores são mais rápidas, mas menos precisas.

2.0.3.4.8 Otimizadores de Parâmetros

Também é preciso definir um algoritmo que proporcione os ajustes dos pesos w, como explicado por Ruder (2016), Zeiler (2012). Os melhores pesos são aqueles que diminuem a diferença entre y e $\hat{y}$, ou seja, encontrar o mínimo local ou global da função custo $J(y, \hat{y})$ em função dos parâmetros w. Otimizamos os parâmetros tentando encontrar os mínimos locais e globais, onde a função de perda, ou seja, a diferença quadrática do valor previsto e o valor real é o mínimo.

Gradiente Descendente é utilizado para otimizar parâmetros em redes neurais e muitos outros algoritmos de aprendizado de máquina, de acordo com Ruder (2016). É uma forma de minimizar a função perda $J(\theta)$ que é parametrizada pelos parâmetros do modelo $\theta \in \mathbb{R}^n$, onde n é o número de parâmetros na função de hipótese.

Os três tipos básicos de gradiente descendente são: *Batch Gradient Descent, Stochastic Gradient Descent (SGD) e Mini-Batch gradient descent*. A convergência acontece mais rápido quando aplicamos o otimizador *Momentum* a superfícies com curvas. Se utilizarmos o Gradiente

Acelerado de Nesterov (NAG), calculamos o gradiente não em relação à etapa atual, mas em relação à etapa futura. Avaliamos o gradiente do previsto e com base na importância, atualizamos os pesos, segundo apresentado por Ruder (2016). Abaixo a definição de alguns otimizadores utilizados.

1. *Stochastic Gradient Descent (SGD)* - Na descida do gradiente estocástico, atualizamos todos os parâmetros para cada exemplo de treinamento $x(i)$ e o rótulo $y(i)$ individualmente, em vez de calcular o gradiente da função de custo em relação aos parâmetros de todo o conjunto de treinamento (RUDER, 2016).

   $\theta = \theta - \alpha \nabla J(\theta; x(i), y(i))$

   Mesmo que exija um número maior de iterações para atingir os mínimos do que o gradiente descendente típico, devido à sua aleatoriedade em sua descida, ainda é computacionalmente muito menos caro do que o gradiente descendente típico. Não tem a redundância, fazendo uma atualização de cada vez.

2. O otimizador *Adagrad (Adaptive Gradient Algorithm)* conectada ao conjunto de dados esparsos, com redes neurais de grande escala, adaptando a taxa de aprendizado aos parâmetros, tendo atualizações maiores para parâmetros não frequentes e atualizações menores para parâmetros frequentes. Enfim, temos uma taxa de aprendizagem diferente para cada parâmetro θ em cada registro. Portanto, o procedimento é ter uma atualização por parâmetro e vetorizá-lo, conforme Ruder (2016). É preciso ajustar a taxa de aprendizado em *Momentum* e NAG.

   $g_i = \nabla J(\theta_i)$, onde g é o gradiente em relação a cada parâmetro θ_i.

   Assim, a atualização do SGD acaba sendo: $\theta_i = \theta_i - \alpha.g_i$, onde todos os parâmetros são atualizados. A diferença do *Adagrad* é que ele modifica a taxa de aprendizado α para cada parâmetro θ_i com base em todos os gradientes anteriores que foram calculados até agora para cada θ_i, como demonstrado por Ruder (2016).

   Para SGD, *Momentum* e NAG, atualizamos todos os parâmetros θ de uma vez. Também usamos a mesma taxa de aprendizagem η. No *Adagrad*, usamos diferentes taxas de aprendizagem para cada parâmetro θ e para cada passo de tempo t.

3. O *RMSProp* é propagação de raiz quadrada média e tenta resolver as taxas de aprendizagem radicalmente decrescentes de *Adagrad* usando uma média móvel do gradiente quadrado. Ele utiliza a magnitude das recentes descidas do gradiente para normalizar o gradiente. No *RMSProp*, a taxa de aprendizagem é ajustada automaticamente e ele escolhe uma taxa de aprendizagem diferente para cada parâmetro. *RMSProp* divide a taxa de aprendizagem pela média da decadência exponencial de gradientes quadrados.

4. *RMSprop* com *Adagrad*: A fim de diminuir a taxa de aprendizagem monotonicamente no algoritmo *Adagrad*, também conforme Ruder (2016), tenta-se obter uma média decrescente

de todos os gradientes quadrados passados de uma forma recursiva. Assim, a média de execução dos gradientes quadrados depende apenas da média anterior e do gradiente atual, o gradiente de execução sendo denotado por $E[g^2]T$ no tempo t.

5. ADAM (*Adaptive Moment Estimation*): é um algoritmo de otimização de taxa de aprendizagem adaptativa, conforme Ruder (2016), Kingma e Ba (2014), que foi projetado especificamente para treinar redes neurais profundas. Usa os algoritmos de otimização de descida de gradiente *RMSprop* e *Momentum*, onde tenta armazenar ambos (média exponencialmente decrescente dos últimos gradientes quadrados e também a média exponencialmente decrescente dos gradientes anteriores). Assim, calcula a média decrescente de gradientes quadrados anteriores e gradientes anteriores.

   É um método de taxa de aprendizagem adaptativa, o que significa que calcula taxas de aprendizagem individuais para diferentes parâmetros. Observe que o gradiente da função de custo da rede neural pode ser considerado uma variável aleatória, uma vez que geralmente é avaliado em algum pequeno lote aleatório de dados. O primeiro momento é a média e o segundo momento é a variância não centrada (o que significa que não subtraímos a média durante o cálculo da variância), segundo definições de Ruder (2016), Kingma e Ba (2014).

   $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$

   $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g^2 t$, onde m é a média decrescente dos gradientes anteriores e v é a média decrescente dos gradientes quadrados anteriores.

   Inicialmente m e v são iniciados com 0 vetores e como as taxas de decaimento são pequenas o suficiente no início onde (ambos os betas estão próximos de 1), vemos que ambos são tendenciosos para 0. Assim, calculando os termos corrigidos de viés:

   $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$

   $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$

   Posteriormente, atualizando os parâmetros semelhantes à técnica de otimização anterior:

   $\theta_t = \theta_{t-1} \breve{} \frac{\eta}{\sqrt{(\hat{v}_t)+\epsilon}} \hat{m}_t$

   α é a taxa de aprendizado.

   β_1 é a taxa de decaimento exponencial para as estimativas do primeiro momento.

   β_2 é a taxa de decaimento exponencial para as estimativas do segundo momento.

   ϵ é um número muito pequeno para evitar qualquer divisão por zero na implementação.

   Kingma e Ba (2014) explicam que os hiperparâmetros $\beta_1, \beta_2 \in [0, 1)$ controlam as taxas de decaimento exponencial dessas médias móveis. As médias móveis são inicializadas como 0, levando a estimativas de momento que são enviesadas em torno de 0, especialmente durante os passos de tempo iniciais. Este viés de inicialização pode ser facilmente neutralizado,

resultando em estimativas corrigidas de viés. O valor de $\beta_1 = 0,9$, $\beta_2 = 0,999$ e $\epsilon = 10^{(-8)}$ são bons valores para a taxa de aprendizagem de acordo com os autores do ADAM.

(RUDER, 2016) recomenda utilizar métodos adaptativos de taxa de aprendizagem, se os dados de entrada forem esparsos.

Por fim, é feita a etapa de treinamento. Nesta etapa, é definida o número de épocas, que é uma iteração passando por todos os dados do conjunto de treino. O *batch size* que é o número de dados utilizados para atualização dos pesos e os dados treino e validação.

São necessárias múltiplas iterações, de forma que o modelo aprenda eficientemente e consiga fazer generalizações futuras, sendo capaz de prever exemplos que não foram vistos anteriormente.

2.0.3.4.9 *Dropout* de unidades (ocultas e visíveis) em uma Rede Neural

Uma estratégia de regularização comuns para CNNs, segundo autores como Ruder (2016), Srivastava et al. (2014), é o *dropout* de unidades (ocultas e visíveis) aplicado na entrada para a penúltima camada. Seria descartar uma unidade, removê-la temporariamente da rede, junto com todas as suas conexões de entrada e saída. Esta técnica estimula as unidades a aprender, a extrair e representar características dos dados de entrada sem depender das unidades vizinhas. A escolha de quais unidades descartar é aleatória. No caso mais simples, cada unidade é mantida com uma probabilidade fixa p independente de outras unidades. A camada *dropout* afeta somente o treino e tem como objetivo prevenir o *overfitting*. Durante a predição todos os neurônios são mantidos ativos.

2.0.3.4.10 Camada Final *Fully Connected*

A camada final de conexões na rede é uma camada totalmente conectada, *fully connected*, ou seja, essa camada conecta todos os neurônios da camada de *pooling* a cada um dos N neurônios de saída, com N sendo a quantidade de classes do modelo para finalizar a classificação (ZHANG; WALLACE, 2015; BOUREAU; PONCE; LECUN, 2010).

2.0.3.4.11 Redes Neurais Convolucionais Residuais (ResNet)

As Redes Neurais Convolucionais Residuais (ResNet), corforme He et al. (2016), são implementadas com saltos de camada que contêm não linearidades (ReLU) e normalização de lote entre eles. Uma matriz de peso adicional pode ser usada para aprender os pesos de salto. Uma motivação para pular camadas é evitar o problema de *vanishing gradients* (quando o gradiente é extremamente pequeno, fazendo com que o peso não mude de valor), reutilizando ativações de uma camada anterior até que a camada adjacente aprenda seus pesos.

Durante o treinamento, os pesos se adaptam para silenciar a camada a montante e amplificar a camada previamente ignorada. No caso mais simples, apenas os pesos para a conexão da camada adjacente são adaptados, sem pesos explícitos para a camada a montante. O salto simplifica efetivamente a rede, usando menos camadas nos estágios iniciais de treinamento. Isso acelera o aprendizado, reduzindo o impacto de *vanishing gradients*, pois há menos camadas para se propagar. A rede então restaura gradualmente as camadas ignoradas à medida que aprende o *feature space*. Perto do final do treinamento, quando todas as camadas são expandidas, o *feature space* fica mais próximo das camadas principais fazendo que o aprendizado aconteça com maior velocidade, como explanado por He et al. (2016).

2.0.4 Análise da Qualidade dos Dados e Validação do Modelo

Em relação às métricas de avaliação de uma entidade, pode-se avaliar a correspondência exata, também conhecida como restrita, considerando a identificação das fronteiras das entidades, que é definido do primeiro ao último *token* da entidade. Os limites inicial e final da fronteira precisam estar idênticos ao real. Já na correspondência flexível, também conhecida como relaxada, uma entidade pode ser classificada como correta mesmo que as suas fronteiras não sejam exatamente iguais ao valor real, segundo definições encontradas no artigo de Li et al. (2020).

Olson e Delen (2008) caracterizam a Matriz de Confusão, também conhecida como Matriz de Coincidência, Matriz de Classificação, Matriz de Contingência ou Matriz de Erro, como uma tabela que permite a visualização do desempenho de um algoritmo de classificação. Os elementos diagonais representam o número de pontos para os quais o rótulo previsto é igual ao rótulo verdadeiro, enquanto os elementos fora da diagonal são aqueles que foram rotulados incorretamente pelo classificador.

De acordo com Olson e Delen (2008), Burke et al. (1988) as métricas de desempenho que podem ser calculadas a partir da Matriz de Confusão, são:

TP (Verdadeiro Positivo) é a quantidade de entidades identificadas e classificadas corretamente - predição para X é positiva e X é de fato positivo.

FP (Falso Positivo) é a quantidade de entidades incorretamente identificadas e classificadas - predição para X é positiva, mas X é negativo.

TN (Verdadeiro Negativo) é a quantidade de entidades que não foram identificadas corretamente pelo modelo - predição para X é negativa e X é de fato negativo.

FN (Falso Negativo) é a quantidade de entidades que não foram identificadas incorretamente pelo modelo - predição para X é negativa e X é de fato positivo.

True Positive Rate*, também conhecida como Sensibilidade ou *Recall, é a probabilidade de que um resultado positivo real seja positivo. Mede a capacidade de um sistema NER de reconhecer todas as entidades.

$$TruePositiveRate = \frac{TP}{TP+FN}$$

***True Negative Rate*, também conhecida como Especificidade**, é a probabilidade de que um negativo real tenha um teste negativo. Indica a capacidade de um sistema NER excluir corretamente palavras que não são entidades em um corpus.

$$TrueNegativeRate = \frac{TN}{TN+FP}$$

***Accuracy* (ou Acuracidade)** avalia a proporção de todos os entidades corretas (verdadeiros positivos e verdadeiros negativos), sobre todos os resultados obtidos.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

***Precision* (ou Precisão), também conhecida como valor Preditivo Positivo** é a probabilidade de que um positivo seja real. É definida como a razão de TP pelo número total de entidades identificadas e reconhecidas pelo modelo.

$$Precision = \frac{TP}{TP+FP}$$

F-Score é uma medida relevante para modelos de classificação como NER, pois faz a média harmônica entre a precisão e a abrangência do modelo, medindo tanto o seu desempenho para reconhecer as entidades corretas, por meio da abrangência, quanto para classificar corretamente, por meio da precisão, conforme definido por Olson e Delen (2008), Burke et al. (1988).

F1-Score* ou *F-measure fornece uma maneira de combinar o *recall* e a *precision* para obter uma única medida. O *recall* e a *precision* podem ter pesos relativos no cálculo da *F1-Score*.

$$FScore = \frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}$$

Resultados deste tipo de pesquisa não são diretamente comparáveis, pois cada ferramenta tem seu próprio critério de avaliação. O mais adequado é uma apuração com métricas e critérios padronizados, variando apenas as técnicas, ferramentas ou conjunto de dados utilizados.

No entanto, iremos citar dois estudos que utilizaram diferentes ferramentas de NER, como o conduzido por Pires, Devezas e Nunes (2017) que mostra o desempenho de ferramentas como *Stanford CoreNLP* com uma medida *F-Score* de 56,10%, *OpenNLP* com 53,63%, *spaCy* com 46,81% e NLTK com 30,97%.

O segundo estudo é o de Neto (2019) que utilizou tipos de representação de linguagem *Flair Embeddings* para o Português. O treino foi realizado com o *corpus* das Coleções Douradas do Concurso Avaliativo de Reconhecimento de Entidades Mencionadas (HAREM) que contém as 10 categorias padrões dessa tarefa (PESSOA, LOCAL, ORGANIZAÇÃO, etc). Até então o melhor resultado conseguido era de *F-Score* de 70,33% e Neto (2019) conseguiu 74,64%.

Para conseguirmos generalizar o modelo construído, ou seja, determinar sua performance em dados não vistos no treinamento, precisamos determinar a base que será utilizada para teste. Abaixo algumas técnicas que podem ser utilizadas com este propóstito:

Divisão Simples ou *Holdout* é definido por (OLSON; DELEN, 2008) como o particionamento dos dados em dois subconjuntos mutuamente exclusivos chamados de conjunto de treinamento e conjunto de teste. É comum designar 2/3 dos dados como o conjunto de treinamento e o 1/3 restante como o conjunto de teste. O conjunto de treinamento é usado na construção do modelo que é então testado no conjunto de teste. Os dados também podem ser divididos em três subconjuntos mutuamente exclusivos: treinamento, validação e teste. O conjunto de validação é usado durante a construção do modelo para evitar o sobreajuste.

Validação Cruzada *k-fold*, também chamada de Estimativa de Rotação, é aplicada para minimizar o viés associado à amostragem aleatória das amostras de dados de treinamento e validação na comparação da precisão preditiva de dois ou mais métodos. O conjunto de dados completo é dividido aleatoriamente em k subconjuntos mutuamente exclusivos de tamanho aproximadamente igual. O modelo de classificação é treinado e testado k vezes, segundo Olson e Delen (2008).

Bootstrapping* e *Jackknifing são técnicas de reamostragem/validação cruzada, que são usadas para gerar novas amostras a partir dos dados originais da população representativa. De acordo com Olson e Delen (2008), Efron (1982), são utilizados especialmente para conjuntos de dados menores.

Bootstrap é um método estatístico para estimar a distribuição de um estimador por amostragem com substituição da amostra original, na maioria das vezes com o objetivo de derivar estimativas robustas de erros padrão e intervalos de confiança de um parâmetro populacional como média, razão de chance, coeficiente de correlação ou coeficiente de regressão. Na estimativa do desempenho dos métodos de classificação, o *bootstrapping* é usado para um número fixo de instâncias dos dados originais para usar no treinamento e usar o resto do conjunto de dados para teste. Efron (1982) salienta que este processo pode ser repetido muitas vezes.

Já o *Jackknife* é um processo iterativo que funciona excluindo sequencialmente uma observação no conjunto de dados e, em seguida, recalculando a estatística desejada. Um parâmetro

é calculado em todo o conjunto de dados e é recalculado repetidamente removendo um elemento após o outro. A estimativa de um parâmetro derivada dessa amostra menor é chamada de estimativa parcial. Um pseudo-valor é então calculado como a diferença entre a estimativa da amostra inteira e a estimativa parcial. A principal aplicação do *Jackknife*, como indica Olson e Delen (2008), Efron (1982), é reduzir o viés e avaliar a variância de um estimador.

2.0.5 Métodos Estatísticos Multivariados - Análise de Agrupamentos

A análise de agrupamento, segundo Kassambara (2017), é uma técnica para agrupar observações em categorias com base em suas semelhanças ou diferenças com relação aos valores observados em várias variáveis para cada indivíduo.

Baker (2010), Xu e Wunsch (2005) definem como uma forma de análise exploratória de dados em que as observações são divididas em diferentes grupos que compartilham características comuns. O objetivo é encontrar pontos de dados que naturalmente se agrupam, dividindo todo o conjunto de dados em um conjunto de clusters. O agrupamento é particularmente útil nos casos em que as categorias mais comuns dentro do conjunto de dados não são conhecidos com antecedência.

A classificação das observações em grupos requer métodos de mensuração da distância ou de similaridade/dissimilaridade entre cada par de observações, que tem como resultado uma matriz conhecida como matriz de dissimilaridade ou matriz de distâncias. A escolha da medida de distância utilizada, de acordo com Kassambara (2017), Berkhin e Becher (2002), é crucial neste tipo de análise de agrupamento e tem grande influência no resultado final.

Os métodos clássicos de mensuração de distância é a Euclidiana e de *Manhattan*, que estão definidas abaixo de acordo com Kassambara (2017). Sendo que x e y são dois vetores de tamanho n.

1. Distância Euclidiana: $D_{euc}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
2. Distância de *Manhattan*: $D_{man}(x, y) = \sum_{i=1}^{n} |(x_i - y_i)|$

Outras medidas de dissimilaridades utilizadas são distâncias baseadas em correlações, conforme Kassambara (2017). Diferentes métodos de correlações são utilizados, como: distância da correlação de *Pearson, Eisen Cosine, Spearman* e *Kendall*.

No caso de dados mistos, quando contém dados lógicos, numéricos, categóricos ou de texto, é recomendável a utilização da distância de *Gower* (1971). Para cada tipo de variável, uma determinada métrica de distância será utilizada e dimensionada para ficar entre 0 e 1. Em seguida, uma combinação linear usando pesos é calculada para criar a matriz de distância final (PODANI; SCHMERA, 2006; KAUFMAN; ROUSSEEUW, 2009).

A distância de *Gower* é calculada como a média das dissimilaridades parciais entre os indivíduos. Conforme Kaufman e Rousseeuw (2009), a forma geral do coeficiente é a seguinte:

$$D_{Gower}(i,j) = \frac{\sum_k \delta_{ijk} d_{ijk}}{\sum_k \delta_{ijk}}$$

d_{ijk} representa a distância entre a i-ésima e a j-ésima unidade calculada considerando a k-ésima variável, levando em consideração sua natureza:

- Lógica - são consideradas como variáveis binárias assimétricas. Neste caso se $x_{ik} = x_{jk}$, então $\delta_{ijk} = 0$, caso contrário, $\delta_{ijk} = 1$;
- Categórica ou de Texto - são consideradas como variáveis categóricas nominais. Neste caso se $x_{ik} = x_{jk}$, então $\delta_{ijk} = 0$, caso contrário, $\delta_{ijk} = 1$;
- Numérica - são consideradas como variáveis com escala de intervalo e $d_{ijk} = \frac{|x_{ik} - x_{jk}|}{R_k}$, sendo R_k o intervalo da k-ésima variável.

Existem vários métodos de clusterização, como apresentado por Kassambara (2017), Xu e Wunsch (2005), dentre eles: Clusterização Particionada, que faz parte as técnicas *K-Means, K-Medoids e CLARA*; Clusterização Hierárquica, que faz parte a Clusterização Aglomerativa e Divisiva.

- Agrupamento *K-Means* é um dos algoritmos mais conhecidos de clusterização, conforme Kassambara (2017), Kaufman e Rousseeuw (2009), Reynolds, Richards e Rayward-Smith (2004). Ele identifica o número k de centróides e, em seguida, aloca todos os pontos de dados para o cluster mais próximo, enquanto mantém os centróides o menor possível. k representa a quantidade de classes e centróide é o local imaginário ou real onde as coordenadas do centróide são a média das coordenadas dos objetos no grupo. A ideia básica é minimizar a variância intragrupo.
- Algoritmo PAM *(Partitioning Around Medoids)* ou *k medoids* é baseado na busca por k objetos representativos ou medóides entre as observações do conjunto de dados, de acordo com os autores Kassambara (2017), Kaufman e Rousseeuw (2009), Reynolds, Richards e Rayward-Smith (2004). Depois de encontrar um conjunto de k medóides, k clusters são construídos atribuindo cada observação ao medóide mais próximo. Um medóide pode ser definido como o objeto de um cluster cuja dissimilaridade média para todos os objetos no cluster é mínima, ou seja, é um ponto localizado mais centralmente no cluster.
- Algoritmo CLARA *(Clustering Large Applications)* é uma extensão do método de clustering do PAM para grandes conjuntos de dados. Em vez de encontrar medóides para todo o conjunto de dados, CLARA considera uma pequena amostra dos dados com tamanho fixo e aplica o algoritmo PAM para gerar um conjunto ótimo de medóides para a amostra

(KASSAMBARA, 2017; KAUFMAN; ROUSSEEUW, 2009; REYNOLDS; RICHARDS; RAYWARD-SMITH, 2004).

A fim de encontrar o número ideal de clusters, como indicado porKaufman e Rousseeuw (2009), é recomendado escolher com base no contexto em questão ou utilizar um dos seguintes métodos: método do Cotovelo (que usa a soma dos quadrados dentro do cluster), método *Silhouette* (medida de quão semelhante um objeto é do seu próprio cluster em comparação com outros clusters), método de estatística de lacuna (compara a variação intracluster para diferentes valores de k com seus valores esperados. A estimativa dos clusters ideais será o valor que maximiza a estatística de lacuna) ou função *NbClust* (Pacote do software R que fornece 30 índices para determinar o número de clusters, além de propor o melhor metódo conforme as combinações de número de clusters, medidas de distância e métodos de agrupamento).

No método de clusterização PAM, Kaufman e Rousseeuw (2009) mencionam que é recomendado selecionar o número de clusters utilizando o método *Silhouette*.

3 Metodologia

Este capítulo apresenta a base de dados, os passos utilizados para a aplicação do modelo de Reconhecimento de Entidades Nomeadas, a metodologia utilizada para avaliação e validação, além do modelo estatístico aplicado sobre os resultados.

3.0.1 Apresentação dos dados

O Hospital das Clínicas da Faculdade de Medicina de Botucatu (HCFMB) vincula-se à Secretaria de Estado da Saúde de São Paulo no Brasil para fins administrativos e associa-se à Faculdade de Medicina de Botucatu da Universidade Estadual "Júlio de Mesquita Filho" – UNESP para fins de ensino, pesquisa e extensão. O HCFMB é a maior instituição pública vinculada ao Sistema Único de Saúde na região (HCFMB, 2019).

O hospital atualmente utiliza o sistema de prontuários eletrônicos SOUL MV Hospitalar e os dados não estruturados foram extraídos do campo "evolução do paciente" presentes nestes prontuários. Após sua extração, foi possível realizar análises considerando também informações estruturadas dos campos: data de entrada e saída, sexo, raça, escolaridade, profissão, tipo sanguíneo, situação (óbito), data de nascimento, estado civil, especialidade e o código de Classificação Estatística Internacional de Doenças e Problemas Relacionados com a Saúde (CID). Os dados foram fornecidos pelo Centro de Informática Médica do Hospital das Clínicas da Faculdade de Medicina de Botucatu (CIMED) por meio de arquivo eletrônico.

Como critério de exclusão, não fazem parte da amostra prontuários sem o preenchimento do campo evolução. As informações referentes a identificação do paciente ou equipe médica também não fazem parte dos dados por questões de confidencialidade.

Este estudo teve aprovação por parte do Comite de Ética em Pesquisa em 20 de maio de 2020 (parecer 4.038.578; CAAE 30365920.5.0000.5411) e cadastrado no SIPE (450/2019).

3.0.2 Determinação dos Métodos

Para a transformação dos dados não estruturados do campo "evolução do paciente" da amostra de prontuários eletrônicos para uma base de dados estruturados, foi construído um modelo que utiliza Redes Neurais Convolucionais para Reconhecimento de Entidades Nomeadas.

Também foram aplicadas algumas técnicas de Processamento de Linguagem Natural no pré-processamento do modelo, para preparação da base e no pós-processamento, para converter o resultado em uma matriz binária (prontuários/entidades), indicando a presença da entidade em cada um dos prontuários.

Após a extração dos dados foi utilizada a Análise de Agrupamentos com distância de *Gower* e o método de clusterização PAM *(Partitioning Around Medoids)* para demonstrar como os dados "descobertos" podem trazer informações relevantes para o conhecimento científico/acadêmico.

A seguir, cada uma das etapas citadas acima serão melhores detalhadas.

3.0.3 Tamanho do conjunto de dados

Uma série de fatores influenciam na determinação do tamanho amostral, como o propósito do estudo, tamanho da população, o risco de selecionar uma amostra enviesada e o erro de amostragem permitido.

Conforme informações do Centro de Informática Médica do Hospital das Clínicas da Faculdade de Medicina de Botucatu (CIMED), de 2012 a 2019 o Hospital cadastrou cerca de 978.000 (novecentos e setenta oito mil) prontuários médicos no seu sistema. Assim, foram selecionados aleatóriamente 30.000 destes prontuários para compor nossa amostra, aproximadamente 3% da base atual existente.

3.0.4 Ferramentas para o Reconhecimento de Entidades Nomeadas

Pelo bom desempenho em modelos de Reconhecimento de Entidades Nomeadas Personalizados, como citado na seção 2.3, por ser uma biblioteca de código aberto escrita em *Python*, por possuir como arquitetura Redes Neurais Convolucionais Residuais (CNN) e análise incremental com *embeddings* de *Bloom*, o *spaCy* foi escolhido para a construção do modelo.

3.0.5 Pré-processamento dos dados

Inicialmente começamos importando as bibliotecas necessárias para o pré-processamento e construção do modelo no *Google Colaboratory* (https://colab.research.google.com/) que permite a execução de códigos utilizando a linguagem de programação *Python* na nuvem por meio do navegador, ou seja, sem a necessidade de utilizar os recursos de uma máquina local.

Todas as implementações estão disponíveis no Github (https://github.com/nailarocha/ModeloNER), podendo ser utilizadas para consulta e implementações futuras.

Os principais pacotes utilizados foram: *pandas* para manipulação de *dataframes* e análises de dados; *numpy* para processamento de *array* de uso geral, aplicado para computação científica; *seaborn* para gráficos estatísticos; *spaCy* para processamento avançado de linguagem natural; *re*

para operações com expressões regulares compiladas; *scikit-learn* para aprendizado de máquinas. As descrições de cada pacote foram baseadas nas informações do *Python Package Index* (PyPI), que é um repositório de *software* para a linguagem de programação *Python* (PYPI, 2021).

Posteriormente importamos a base de prontuários e criamos uma função (com o pacote "re") para tratar os dados, removendo símbolos, acentos, quebras de linhas e caracteres especiais. Os dados também foram normalizados para letras minúsculas.

3.0.6 *Corpus* Clínico

Dos 30.000 prontuários disponíveis para o estudo, 1.200 foram utilizados na construção do *corpus*. Ao considerarmos apenas textos únicos, sem repetição de textos padrões e com entidades presentes, temos uma base de 1.036 prontuários. Ao aplicar a estratégia *holdout*, conforme definido na seção 2.4, nos 1.036 prontuários, 725 foram utilizados para treinamento e 311 reservados para a realização dos testes. Estes prontuários geraram um *corpus* clínico com as seguintes entidades: MEDICAÇÃO, CONDIÇÃO, TRATAMENTO, SINTOMA, EXAME e DIAGNÓSTICO, que serão definidas abaixo.

- MEDICAÇÃO: todo e qualquer medicamento citado no prontuário, seja prescrito na consulta atual ou do histórico do paciente;
- CONDIÇÃO: condições prévias do paciente, como características fisiológicas consideradas normais (não sintomática de alguma doença);
- TRATAMENTO: conduta prescrita já realizada ou a realizar relacionada ao diagnóstico;
- SINTOMA: fenômeno subjetivo ou característica fisiológica referida por um paciente geralmente relacionado com uma doença;
- EXAME: procedimento médico que visa auxiliar o diagnóstico;
- DIAGNÓSTICO: determinação de uma doença a partir da descrição dos sintomas e realização de exames.

Os dados foram classificados manualmente, criando o "padrão-ouro", em que uma sequência de anotações de entidades nomeadas, como *strings* do tipo tuplas no formato "start_char, end_char, label", representando as posições da entidade. Essas anotações foram feitas na ferramenta *SpaCy Annotation Tool* desenvolvida por Murugavel (2020) que permite a atribuição dos rótulos personalizados no texto.

Para realização das anotações, os dados foram divididos em arquivos com 10-20 prontuários no formato *txt*. Posteriormente foram carregados na ferramenta e marcados conforme imagem ilustrativa abaixo. As marcações foram realizadas por alunos do quarto ano do curso de medicina da Faculdade da Universidade Estadual Paulista "Júlio de Mesquita Filho".

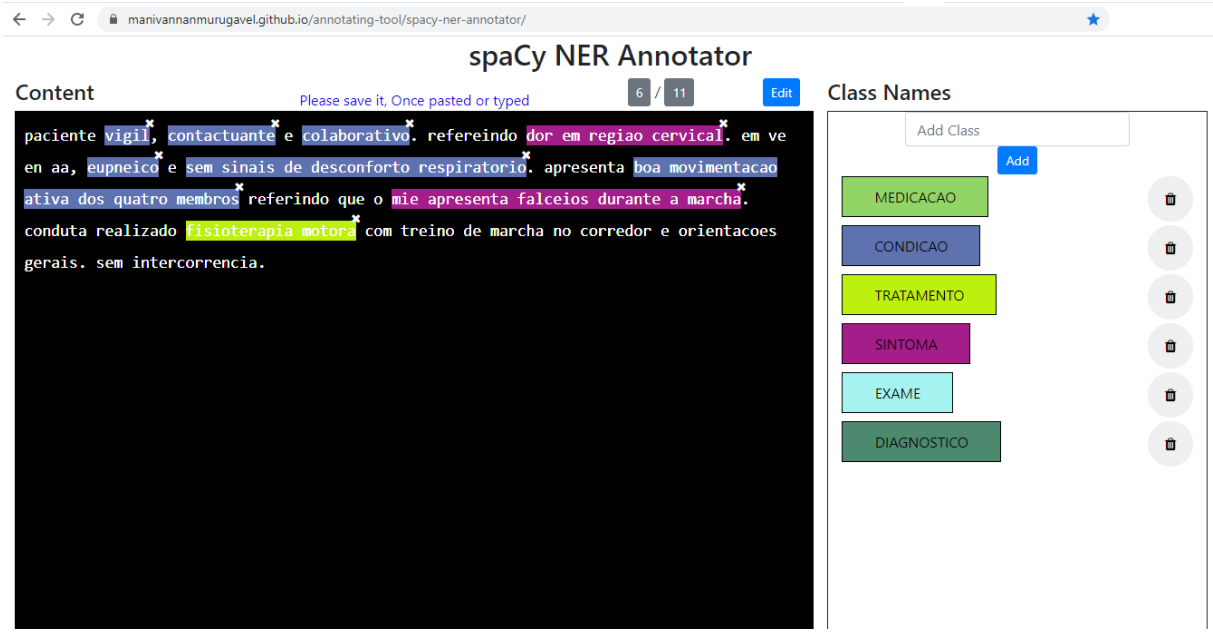


Figura 2. Ferramenta de anotações de entidades. Fonte: (MURUGAVEL, 2020)

Após as marcações, os arquivos foram extraídos no formato *json* e convertidos para uma lista de tuplas, formato de entrada do *spaCy*. Cada tupla deve conter os índices inicial e final da entidade nomeada no texto e a categoria ou rótulo da entidade nomeada. Abaixo alguns exemplos dos dados de treino e teste do *spaCy*.

```
TRAIN_DATA = [
              ('refere melhora miccao com combodart toma marevan e furosemida 6cp/dia pedido urina',
               {'entities': [(26, 35, 'MEDICACAO'), (41, 48, 'MEDICACAO'), (51, 61, 'MEDICACAO'), (77, 82, 'EXAME')]}),
              ('checo gasometria venosa ph 7,24 bic 16,1 k 3,79 lac 1,1 hb 13,7 conduta prescrevo bic 60ml',
               {'entities': [(6, 23, 'EXAME'), (82, 85, 'MEDICACAO')
              ]
```

```
TEST_DATA = [
              ('atendimento psa qp dor de ouvido ha 2 dias hpma dor ha 1 dia em ouvido esquerdo, hd otite aguda',
               {'entities': [(19, 32, 'SINTOMA'), (48, 51, 'SINTOMA'), (84, 95, 'DIAGNOSTICO')]})
              ]
```

Figura 3. Exemplo de tuplas para treino e teste do modelo - formato *spaCy*.

3.0.7 Descrição do Modelo e suas Configurações

Inicialmente foi criado um modelo vazio com a função $spaCy.blank$ e selecionado o idioma português. Como é um modelo vazio, começamos a construção com o $nlp.add\_pipe$ para adicionar o componente de entidades nomeadas no $doc.ents$ ao *pipeline*.

O *pipeline* de processamento é uma série de funções aplicadas sequencialmente para a transformação de texto em *token*. Nesse processo é criado um objeto *Doc* (abreviação de documento) que será processado em várias etapas distintas e considera as palavras em seu

contexto. Através do *Doc* é possível acessar informações do texto de uma maneira estruturada (SPACY, 2021).

Para dar início ao treinamento de um novo modelo com pesos aleatórios, utilizamos a função $nlp.begin\_training$. Para prevenir que o modelo convirja para uma solução sub-ótima utilizamos o comando *random.shuffle* para embaralhar aleatoriamente os dados do treinamento antes de cada iteração. Desta forma asseguramos que o modelo não faça generalizações com base na ordem dos exemplos passados durante o treinamento e assim é possível obter uma melhor acurácia.

Cada exemplo do treinamento consiste no texto de um prontuário e suas respectivas anotações. A cada prontuário ele faz uma previsão e em seguida consulta os rótulos para verificar se está correta.

De acordo com os resultados, os pesos são ajustados para que a previsão correta obtenha uma melhor pontuação na próxima iteração. Quanto maior a diferença, mais significativo será o gradiente de erro da função de perda. Para cada iteração, o modelo é atualizado por meio do comando $nlp.update$.

A função de perda utilizada pelo modelo, conforme (THINC, 2021) é a perda de entropia cruzada multiclasse, ou perda logarítmica. A entropia cruzada aumenta à medida que a probabilidade prevista de uma amostra diverge do valor real.

A cada iteração, agrupamos os exemplos em lotes, que são denominados *batches* e o parâmetro para denotar o tamanho do lote é o *minibatch*. O *batch size* ideal, conforme spaCy (2021), dependerá dos componentes do *pipeline*, do comprimento dos documentos, do número de processos e da quantidade de memória disponível.

Na tentativa de otimizar o modelo, fizemos diversas simulações alterando os parâmetros do modelo. Uma das simulações foi a utilização da função *minibatch* (TRAIN_DATA, $size = compounding$ (4.0, 32.0, 1.001)) que gera uma série infinita de valores compostos. Na função determinamos o valor de *start*, o valor de *stop* e o *compound*, sendo este último o fator de composição para a série.

Foram testados diferentes valores para o hiperparâmetro *dropout* (taxa de abandono) e uma das alternativas avaliadas foi a utilização da função $dropout = decaying$ (0.6, 0.2, 1e-4), com o intuito de definir uma alta taxa de abandono no início e decair para um valor mais razoável para evitar o *overfitting* imediato da rede. Vários autores utilizam o *dropout* de 0.5, como Slatton (2014), Srivastava et al. (2014), Hinton e Osindero (2006), já o spaCy (2021), recomenda valores próximos a 0.2. Por isso foram realizados testes para determinar os valores que seriam utilizados no modelo.

Já o otimizador define como o modelo deve alterar seus pesos para reduzir o erro atual, que é calculado quando comparamos os dados previstos com os resultados reais obtidos. Foram testados os otimizadores ADAM, com as configurações padrão do *spaCy* ($learn\_rate = 0.001$,

$beta1 = 0.9$, $beta2 = 0.999$, $eps = 1e-08$, $L2 = 1e-6$, $grad\_clip = 1.0$, $use\_averages = True$, $L2\_is\_weight\_decay = True$) e o SGD ($learn\_rate = 0.001$, $L2 = 1e{-}6$, $grad\_clip = 1.0$).

3.0.8 Análise de Desempenho do Modelo para o Reconhecimento de Entidades Nomeadas

Os hiperparâmetros que fazem a rede alcançar o menor valor de perda e apresentam as melhores métricas *F-Score, Recall e Precision* serão considerados como de melhores desempenho. Por isso, após o treinamento do modelo foi utilizada a função $scorer.score$ para verificar as métricas do modelo na base de teste.

Para acompanhar o desempenho qualitativo do modelo foram utilizados alguns recursos como: listagem das entidades e seus respectivos rótulos.

```
# lista de entidades encontradas com cada rótulo
def show_ents(doc):
  if doc.ents:
    for ent in doc.ents:
      #print(ent.text + ' | ' + str(ent.start_char) + ' - ' + str(ent.end_char) + ' | ' + (ent.label_))
      print(ent.text + ' | ' + (ent.label_)+ ';')
    else:
      print('Mais nenhuma entidade encontrada')
doc = prdnlp(str(ACC))
saida = show_ents(doc)
```

```
furosemida | MEDICACAO;
tiamazol | MEDICACAO;
hidrocortisona | MEDICACAO;
lugol | MEDICACAO;
hipertireoidismo | DIAGNOSTICO;
bradicardia | SINTOMA;
amiodarona | MEDICACAO;
hanseniase | DIAGNOSTICO;
tapazol | MEDICACAO;
propranolol | MEDICACAO;
digoxina | MEDICACAO;
xarelto | MEDICACAO;
beg | CONDICAO;
corado | CONDICAO;
hidratado | CONDICAO;
```

Figura 4. Exemplo de lista de entidades extraídas com seus respectivos rótulos.

Também foi possível visualizar um resumo com a quantidade de entidades extraídas de cada categoria, durante as validações e testes.

```
# quantidade de entidades encontradas
print('MEDICACAO: ' + str(len([ent for ent in doc.ents if ent.label_ == "MEDICACAO"])))
print('CONDICAO: ' + str(len([ent for ent in doc.ents if ent.label_ == "CONDICAO"])))
print('DIAGNOSTICO: ' + str(len([ent for ent in doc.ents if ent.label_ == "DIAGNOSTICO"])))
print('SINTOMA: ' + str(len([ent for ent in doc.ents if ent.label_ == "SINTOMA"])))
print('TRATAMENTO: ' + str(len([ent for ent in doc.ents if ent.label_ == "TRATAMENTO"])))
print('EXAME: ' + str(len([ent for ent in doc.ents if ent.label_ == "EXAME"])))
```

```
MEDICACAO: 734
CONDICAO: 814
DIAGNOSTICO: 861
SINTOMA: 1170
TRATAMENTO: 233
EXAME: 339
```

Figura 5. Exemplo de lista resumo com a quantidade de entidades extraídas de cada categoria.

Outra forma utilizada foi uma análise visual, verificando o texto de 1 prontuário por vez com a função *displacy*.

```
# análise de 1 registro
test_text = "PS mae refere que crianca foi picada por inseto hoje"
doc = prdnlp(test_text)
for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
# exibição gráfica das entidades na frase
displacy.render(doc,style='ent',jupyter=True)
```

```
alergia 82 89 CONDICAO
beg 100 103 CONDICAO
corado 105 111 CONDICAO
hidratado 113 122 CONDICAO
afebril 124 131 CONDICAO
eupneico 133 141 CONDICAO
hiperemia 227 236 SINTOMA
calor 239 244 SINTOMA
hidroxizine 287 298 MEDICACAO
compressa fria local 306 326 TRATAMENTO
```

PS mae refere que crianca foi picada por inseto hoje, por volta das 12h. Esta com alergia CONDICAO local. EF beg CONDICAO , corado CONDICAO
central puntiforme, com pequena vesicula, halo de cerca de 4cm com hiperemia SINTOMA e calor SINTOMA , em regiao do maleolo medial esquerdo. C

Figura 6. Exemplo de análise gráfica do texto de 1 prontuário, utilizando a função *displacy*.

3.0.9 Pós-processamento dos dados

Nesta etapa geramos uma tabela com todos os campos já existentes no prontuário, além das novas colunas que criamos após a aplicação do modelo. Cada coluna é uma classe que foi extraída (sintoma, diagnóstico, medicação, exame, condição, tratamento) preenchida com o que foi "minerado" do campo narrativo. Abaixo a tabela gerada com a inclusão das classes em colunas.

RACACOR	ESCOLARIDADE	NMPROFISSAO	TPSANGUINEO	SITUACAO	NASCIMENTO	ESTADOCIVIL	DSEVOLUCAO	ESPECIALIDADE	CID	CIDDESCRICAO	SINTOMA	DIAGNOSTICO	MEDICACAO	EXAME	CONDICAO	TRATAMENTO
IRANCA	SUPERIOR COMPLETO	PROFESSOR	NaN	NaN	21/04/1951	CASADO	10h paciente admitida no ambulatorio de quimio...	ENFERMAGEM	Z512	OUTRA QUIMIOTERAPIA	NaN	NaN	paclitaxel	NaN	consciente, orientada, calma, comunicativa, co...	quimioterapia
IRANCA	MEDIO (2Âº GRAU) INCOMPLETO	ESTUDANTE	NaN	NaN	26/05/2004	CASADO	pa maternidade 03 07 2020 16 anos p...	OBSTETRICIA	Z359	SUPERVISAO NAO ESPECIFICADA DE GRAVIDEZ DE ALT...	cefaleia, alergias, dor	peritonite	dramin	usg	beg, chaaa, lote	NaN
IRANCA	MEDIO (2Âº GRAU) COMPLETO	EMPREGADO DOMESTICO	NaN	NaN	15/06/1995	DIVORCIADO	paciente com queixas contipacao ha + 1 semana ...	PLANTONISTA GENERALISTA 1	Z000	EXAME MEDICO GERAL	dor abdominal, vomito, febre, dor na palpacao,...	NaN	NaN	rx	orientado, chaaa	NaN
IRANCA	FUNDAMENTAL 5A A 8A COMPLETO	CAMINHONEIRO	NaN	NaN	17/12/1959	CASADO	psa paciente admitido na se proveniente do cm...	PLANTONISTA GENERALISTA 1	I213	INFARTO AGUDO TRANSMURAL DO MIOCARDIO, DE LOCA...	dor precordial, edemas	NaN	NaN	NaN	orientado, contactuante, hidratado, corado	NaN

Tabela 2. Tabela dos prontuários acrescida de colunas dos dados extraídos (cada célula possui um vetor de entidades extraídas do seu respectivo prontuário).

Assim, realizamos a tokenização destes textos, processo que envolve a separação de cada palavra *tokens* em blocos constituintes do texto, para depois convertê-los em vetores numéricos.

O objetivo é obter uma matriz que as informações extraídas estejam em colunas, e as linhas contenham valores binários que indiquem a existência do rótulo correspondente àquela coluna. Abaixo o exemplo dos dados transformados da classe Medicação.

NMPROFISSAO	TPSANGUINEO	SITUACAO	NASCIMENTO	ESTADOCIVIL	DSEVOLUCAO	ESPECIALIDADE	CID	CIDDESCRICAO	MEDICACAO	enalapril	paclitaxel	dramin	metronidazol	taxol	herceptin
PROFESSOR	NaN	NaN	21/04/1951	CASADO	10h paciente admitida no ambulatorio de quimio...	ENFERMAGEM	Z512	OUTRA QUIMIOTERAPIA	paclitaxel	0	1	0	0	0	0
ESTUDANTE	NaN	NaN	26/05/2004	CASADO	pa maternidade 03 07 2020, 16 anos procedente...	OBSTETRICIA	Z359	SUPERVISAO NAO ESPECIFICADA DE GRAVIDEZ DE ALT...	dramin	0	0	1	0	0	0
ESTUDANTE	NaN	NaN	14/02/2006	SOLTEIRO	cirurgia pediatrica 14 08 2013 9 po abaixamen...	CIRURGIA PEDIATRICA	K593	MEGACOLON NAO CLASSIFICADO EM OUTRA PARTE	metronidazol	0	0	0	1	0	0
COMERCIANTE	NaN	NaN	27/06/1969	SOLTEIRO	abr 2012 cdi mama d ct3n1m0 iiia re rp herb2 ...	ONCOLOGIA CLINICA	C509	NEOPLASIA MALIGNA DA MAMA, NAO ESPECIFICADA	taxol, herceptin, taxol, herceptin, herceptin,...	0	0	0	0	1	1
)ESEMPREGADO	NaN	NaN	13/05/1994	SOLTEIRO	amb. de disturbios do calcio 07 07 17 caso no...	ENDOCRINOLOGIA	M819	OSTEOPOROSE NAO ESPECIFICADA	alendronato, acido zoledronico, dipirona, cicl...	0	0	0	0	0	0

Tabela 3. Matriz com a tokenização das entidades da classe MEDICAÇÃO.

O Reconhecimento de Entidades Nomeadas considera o contexto para a classificação das entidades, e desta forma, é possível classificar o texto mesmo com acrônimos, erros gramaticais ou erros de digitação. Por isso, temos 2.578 palavras/expressões únicas identificadas como medicamentos. No entanto, para seguirmos com as análises multivariadas foi necessário uma revisão manual, assegurando a não repetição pelos motivos citados acima. Para ilustrar, apresentamos abaixo o exemplo do medicamento propranolol que foi escrito de 15 formas distintas.

Após a revisão e padronização dos dados da base obtivemos 1.349 nomes de medicamentos que foram considerados como variáveis categóricas na Análise de Agrupamento que será melhor detalhada na seção 3.10.

0	popranolol	propranolol
1	porpranolol	propranolol
2	prapranolol	propranolol
3	pronalol	propranolol
4	propanalol	propranolol
5	propanol	propranolol
6	propanolol	propranolol
7	proparanolol	propranolol
8	propralol	propranolol
9	propranol	propranolol
10	propranolo	propranolol
11	propranolol	propranolol
12	proranolol	propranolol
13	prorpanolol	propranolol
14	prpranolol	propranolol

Tabela 4. Exemplo do medicamento propranolol que foi escrito de 15 formas distintas nos prontuários da amostra.

3.0.10 Métodos Estatísticos Multivariados - Análise de Agrupamentos

Com o intuito de avaliar os dados extraídos e entender o potencial de medir, explicar ou predizer o grau de relacionamento e impacto das variáveis, aplicamos uma Análise de Agrupamento na base gerada de medicamentos, considerando também as informações sociodemográficas presentes no prontuário, abaixo mencionadas.

- IDADE (DTENTRADA - NASCIMENTO)/365,25)
- SEXO (0 = feminino; 1 = masculino)
- RAÇA/COR (0 = não sabe/não declararam; 1 = branca; 2 = parda; 3 = preta; 4 = amarela; 5 = indígena)
- ESCOLARIDADE (0 = não sabe/não declararam; 1 = não alfabetizado; 2 = somente alfabetizado; 3 = fundamental 1º- 4º ano completo ou incompleto (primário/admissão); 4 = fundamental 5º- 8º ano completo ou incompleto (ginásio/1º grau); 5 = médio completo ou incompleto (2º grau); 6 = superior completo ou incompleto, mestrado ou doutorado (3º grau/pós-graduação);
- ESTADO CIVIL (0 = não sabe/não declararam; 1 = solteiro; 2 = casado ou união estável; 3 = desquitado ou divorciado; 4 = viúvo)

Não foram diretamente consideradas na Análise de Agrupamento as variáveis: DTENTRADA (data de entrada no hospital), DTSAIDA (data de saída do hospital), NMPROFISSAO (profissão), TPSANGUINEO (tipo sanguíneo), SITUACAO (óbito), NASCIMENTO (data de nascimento), ESPECIALIDADE (especialidade médica que atendeu o paciente), CID e CIDDESCRICAO (Classificação Estatística Internacional de Doenças e Problemas Relacionados com a Saúde). Estas variáveis foram excluídas devido ao foco do trabalho e, em alguns casos, devido à existência de dados faltantes, que requereria a aplicação de técnicas estatísticas de imputação de dados.

Foi escolhido o *framework R Studio*, que utiliza linguagem de programação R para desenvolver esta parte do trabalho. Foram baixados os pacotes necessários utilizando *install.packages* e *library*. Os dados foram transformados com a função *mutate_if* encontrada no pacote *dplyr*.

Primeiro, foi analisada a dissimilaridade entre as observações no conjunto de dados utilizando a distância de Gower, devido a presença de dados mistos. Para este fim, foi utilizada a função *daisy* que calcula as dissimilaridades de pares.

Em seguida, foi aplicado o algoritmo *Partition Around Medoids* (PAM), método que apresentou melhores resultados e é o mais adequado quando é utilizada a distância de Gower.

A quantidade de clusters foi definida com base na análise do *Silhouette Width*, métrica de validação que determina quão semelhante uma observação é do seu próprio *cluster* em comparação com o *cluster* vizinho mais próximo.

4 Resultados e Avaliação do Desempenho do Modelo

Neste capítulo é apresentado um estudo com dados extraídos pelo modelo, utilizando toda metodologia exposta acima e os conceitos apresentados na fundamentação teórica. Também será exposto os resultados da avaliação de desempenho e a análise e discussão desses resultados

4.0.1 Resultados e Desempenho do Modelo

Nos 30.000 prontuários da amostra, existem 13.823 (46,1%) pacientes do sexo masculino e 16.177 (53,9%) do sexo feminino. A idade variou de 0 a 104 anos, com média de 45,6 anos, mediana de 49,2 anos e desvio-padrão de 25,3 anos.

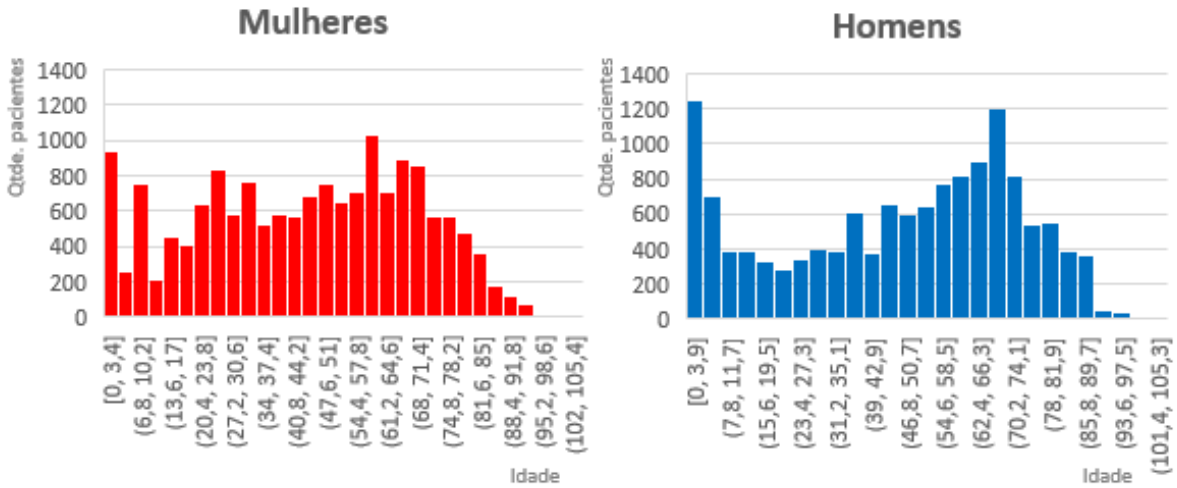


Figura 7. Histograma por idade e sexo dos 30.000 pacientes da amostra.

Observa-se que as mulheres têm uma distribuição mais uniforme ao longo da vida em relação a frequência que recorrem ao atendimento médico. Já os homens possuem maiores concentrações próximo ao nascimento e após os 50,0 anos.

A figura 8 apresenta a quantidade de pacientes atendidos por ano e mês, considerando a variável "Data de Entrada" como referência. Os meses com maiores quantidades de atendimentos são janeiro, julho e agosto, meses próximos ou que coincidem com recessos e férias escolares.

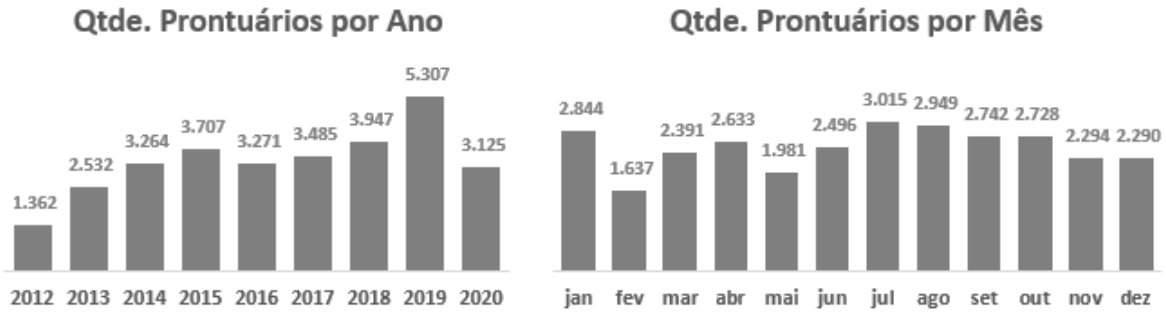


Figura 8. Quantidade de pacientes da amostra atendidos por ano/mês.

Para descrever melhor o grupo em análise, a tabela abaixo apresenta algumas variáveis demográficas e socioeconômicas.

Variáveis	Categorias	n	%
Demográficas			
Sexo	Masculino	13.823	46,1
	Feminino	16.177	53,9
Faixa etária	0-12 anos	4.494	15,0
	13-19 anos	1.431	4,8
	20-60 anos	13.730	45,8
	61 anos e mais	10.345	34,5
Cor da pele	Branca	26.348	87,8
	Preta	1.016	3,4
	Parda	2.311	7,7
	Amarela	39	0,1
	Indígena	2	0,0
	Não sabem/Não declararam	284	0,9
Socioeconômicas			
Escolaridade	Não alfabetizado	3.029	10,1
	Somente alfabetizado	1.993	6,6
	Primário/admissão	4.598	15,3
	Ginásio/1° grau	10.225	34,1
	2° grau	6.946	23,2
	3° grau/ superior/pós-graduação	2.586	8,6
	Não sabem/Não declararam	623	2,1
Estado civil	Solteiro	11.360	37,9
	Casado/União-estável	13.031	43,4
	Divorciado/Desquitado	2.356	7,9
	Viúvo	3.017	10,1
	Não sabem/Não declararam	236	0,8

Tabela 5. Variáveis demográficas e socioeconômicas dos 30.000 prontuários.

Este estudo é composto por diferentes tipos de narrativas de múltiplas especialidades clínicas. Abaixo uma tabela com as especialidades mais presentes na amostra.

Especialidades	n	%
Plantonista generalista 1	4.009	13,4
Pediatria	3.886	13,0
Clínica médica geral	3.684	12,3
Cirurgia geral	1.977	6,6
Ortopedia/Traumatologia	1.319	4,4
Nefrologia	1.295	4,3
Obstetrícia	1.263	4,2
Cardiologia	1.165	3,9
Neurologia clínica	1.159	3,9
Infectologia	697	2,3
Oftalmologia	632	2,1
Otorrinolaringologia	617	2,1
Outros	8.297	27,7

Tabela 6. Descrições das Especialidades mais frequentes nos 30.000 prontuários do estudo.

Há uma alta dispersão nos códigos de Classificação Estatística Internacional de Doenças e Problemas Relacionados com a Saúde (CID), sendo o maior deles presente em apenas 1,8% dos prontuários. Abaixo uma tabela das descrições dos CIDs mais frequentes neste banco de dados.

CID Descrição	n	%
Exame médico geral	527	1,8
Dor aguda	513	1,7
Outras septicemias especificadas	509	1,7
Insuficiência cardíaca congestiva	486	1,6
Infarto agudo do miocárdio não especificado	409	1,4
Doenca renal em estádio final	368	1,2
Acidente vascular cerebral, não especificado como hemorrág	337	1,1
Estado de mal epiléptico, não especificado	325	1,1
Outros infartos cerebrais	296	1,0
Pneumonia bacteriana não especificada	294	1,0
Exame dos olhos e da visão	290	1,0
Broncopneumonia não especificada	283	0,9
Outros	25.363	84,5

Tabela 7. Descrições dos CIDs mais frequentes nos 30.000 prontuários do estudo.

4.0.2 Hiperparâmetros

A figura abaixo mostra uma visão geral do desempenho da rede por época, em relação à função de perda. É possível perceber que quando consideramos o $dropout = decaying$ (0.6, 0.2, 1e-4), obtivemos os menores valores de perdas em ambas as situações, com e sem a função *minibatch - size compounding* e que a partir de determinada época, aproximadamente 35, a função

da perda apresentou um comportamento quase linear, diferente da configuração 0.5, que precisou de uma quantidade maior de iterações para estabilizar o modelo. Também é importante ressaltar que o *dropout* 0.2 apresentou resultados semelhantes ao *decaying(0.6, 0.2, 1e-4)*, mas demorou um pouco mais para estabilizar o modelo. Outro fator determinante nas escolhas dos parâmetros foi o tempo de execução de treinamento para 1.000 ou mais épocas, significativamente maior que os demais.

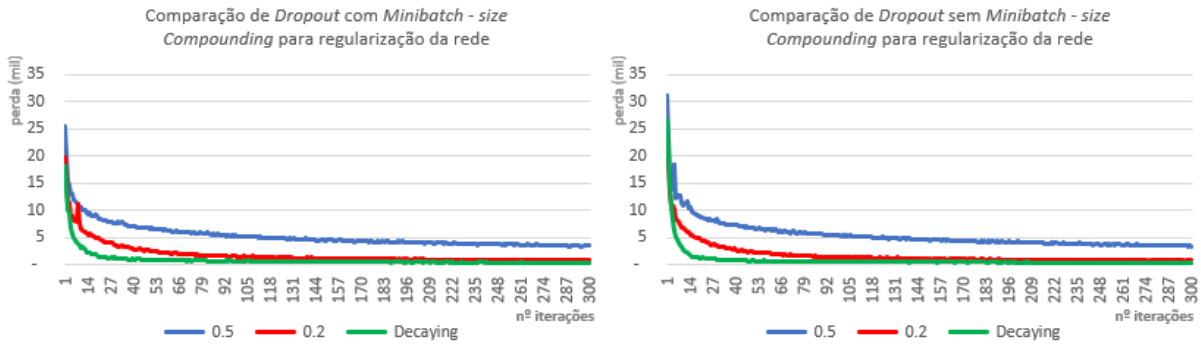


Figura 9. Simulação para os hiperparâmetros *dropout* e *batch size* com 300 iterações, considerando as funções *decaying* e *minibatch - size compounding*.

O otimizador ADAM com os hiperparâmetros recomendados na literatura, citados na seção 2.3, apresentou melhores resultados que o SGD e por isso foi o utilizado no modelo.

4.0.3 Entidades Extraídas

Alguns exemplos de classificações feitas pelo modelo desenvolvido estão apresentados abaixo. Todos são trechos retirados do campo "evolução" dos prontuários em análise.

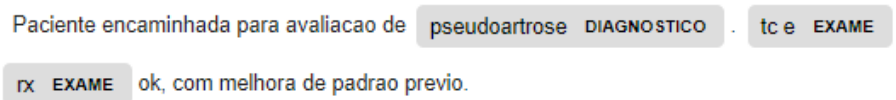


Figura 10. Exemplo de Extração de Entidades Nomeadas com a presença de abreviatura.

No primeiro exemplo temos a extração bem sucedida das entidades Diagnóstico e Exame, mesmo nos casos em que os nomes dos exames estão abreviados ("tc" para tomografia computadorizada e "rx" para raio-x).

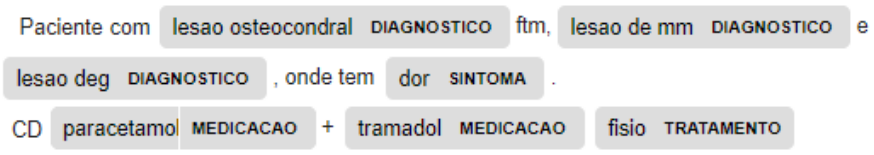


Figura 11. Exemplo de Extração de Entidades Nomeadas com a presença de símbolos.

No segundo exemplo, conseguimos observar a capacidade de aprendizado, com o modelo conseguindo detectar diferentes tipos de lesões como Diagnóstico, além de entender os medicamentos separados pelo sinal "+" e a abreviação do tratamento "fisio".

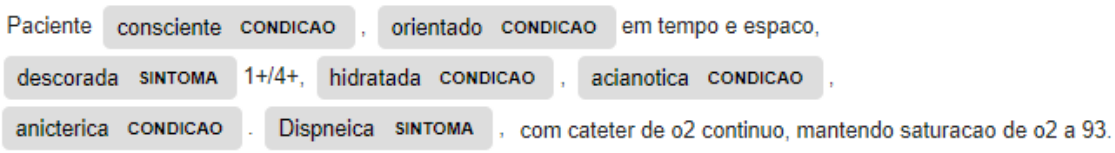


Figura 12. Exemplo de Extração de Entidades Nomeadas com detecção de entidades alternadas.

No terceiro exemplo é possível visualizar a capacidade de detecção de Condição e Sintoma de forma alternada, mostrando a eficiência na utilização do método neste tipo de análise.

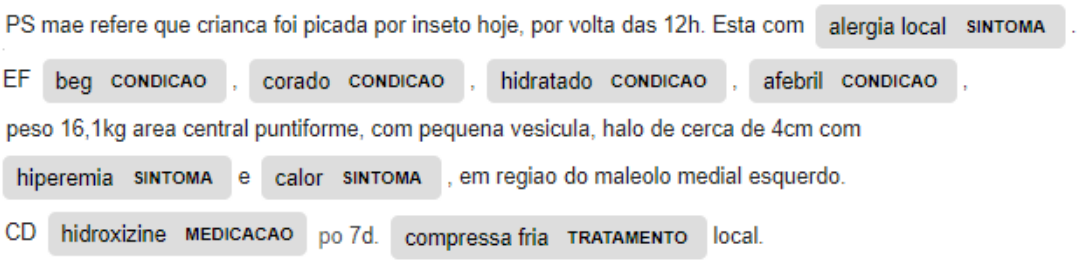


Figura 13. Exemplo de Extração de Entidades Nomeadas com a presença de diversas entidades distintas.

No quarto exemplo temos a extração de quatro das seis entidades mapeadas do modelo (Sintoma, Condição, Medicação e Tratamento).

4.0.4 Análise da Qualidade dos Dados e Validação do Modelo

Os seguintes resultados foram obtidos quando avaliamos as correspondências exatas, considerando a identificação das fronteiras das entidades, que é definida do primeiro ao último *token*.

Entidades	F-Score	Precision	Recall
Condição	82,652	90,298	76,201
Diagnóstico	49,272	54,424	45,011
Exame	54,664	72,609	43,832
Medicação	80,966	87,474	75,360
Sintoma	58,863	68,768	51,451
Tratamento	47,312	57,592	40,146
Modelo	**63,867**	**72,725**	**56,932**

Tabela 8. Métricas de avaliação de extração de entidades nomeadas do modelo.

Com base nas referências apresentadas na seção 2.4 de estudos na língua portuguesa e considerando a complexidade da base de dados utilizada, podemos concluir que os resultados foram significativos, com um *F-Score* de 63,9%. Contudo, observamos ainda a possibilidade de melhoria no desempenho com uma base maior de treinamento ou ajustes nos hiperparâmetros do modelo.

	200 PRONTUÁRIOS			1.200 PRONTUÁRIOS					
Entidades	**Qtde.**	**%**	**Média**	**Qtde.**	**%**	**Média**	**F-Score**	**Precision**	**Recall**
Condição	631	21%	3,2	3.304	19%	2,8	-12,397	-2,009	-21,758
Diagnóstico	606	20%	3,0	3.970	23%	3,3	-7,091	-4,066	-9,374
Exame	273	**9%**	1,4	1.162	**7%**	1,0	14,664	25,551	9,050
Medicação	359	12%	1,8	2.928	17%	2,4	-0,284	-5,383	3,138
Sintoma	943	**31%**	4,7	5.267	**30%**	4,4	-9,848	0,476	-17,684
Tratamento	214	**7%**	1,1	933	**5%**	0,8	-16,973	-2,408	-29,085
	3.026		**15,1**	**17.564**		**14,6**	**-5,753**	**1,910**	**-11,532**

Tabela 9. Camparativo das métricas de avaliação de extração de entidades nomeadas do modelo com 200 prontuários e 1.200 prontuários no *Corpus* Clínico.

Ao comparar os resultados do modelo de treinamento inicial, com 200 prontuários e o atual, com 1200 prontuários, vemos uma melhora significativa na *Precision*, ou seja, há um aumento da probabilidade de que um positivo seja real. Também conseguimos perceber um desbalanceamento na quantidade de entidades das classes Exame e Tratamento presentes nos prontuários, o que interfere no nosso resultado final. Este fator é explicado pela quantidade média de cada entidade por prontuário (em um prontuário temos em média: 2,8 condições relatadas; 3,3 diagnósticos citados; 1,0 exame prescrito ou avaliado; 2,4 medicações consumidas ou preescritas;

4,4 sintomas relatados e 0,8 tratamentos ressaltados), sendo menores nas entidades citadas como desbalanceadas.

4.0.5 Métodos Estatísticos Multivariados - Análise de Agrupamentos

Dos 28.800 prontuários avaliados (não foram considerados os 1.200 prontuários utilizados na construção (treino/teste) do modelo), 17.382 continham informações acerca de medicamentos prescritos ou em uso pelos pacientes.

Depois de calcular a *Silhouette Width* para *clusters* variando de 2 a 10 para o algoritmo PAM, obtivemos que 4 *clusters* produzem o valor mais alto, sendo o número ideal de grupos neste caso.

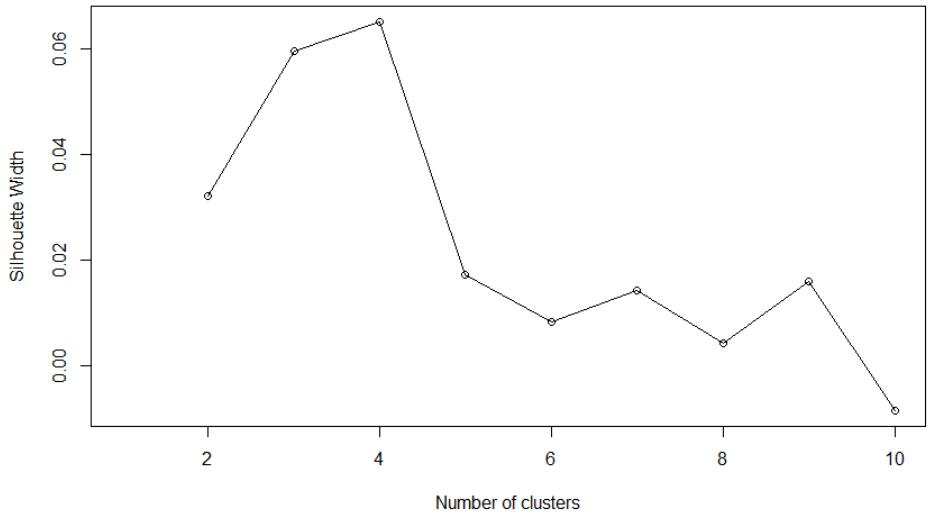


Figura 14. Análise do *Silhouette Width* demonstra 4 *clusters* como sendo o ideal.

Após executar o algoritmo e selecionar a quantidade de *clusters*, o resumo do resultado com a interpretação de cada um dos grupos gerados é apresentado abaixo.

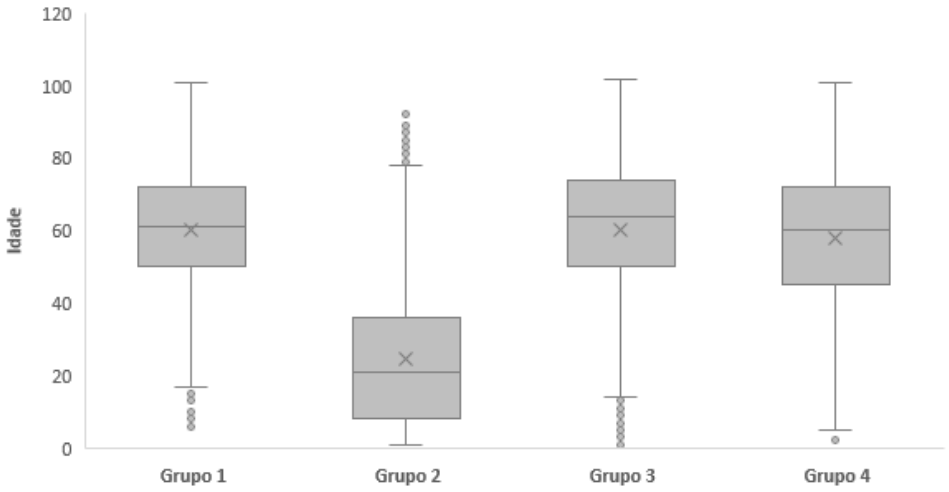
Grupo 1 - Composto por mulheres com idade média de 60,4, mediana de 61,0 e desvio padrão de 16,0 anos. Possuem ginásio/primeiro grau de escolaridade e são casadas. Têm um consumo predominante dos medicamentos: AAS, Losartana, Omeprazol e Sinvastatina.

Grupo 2 - Grupo misto (homens e mulheres) com idade média de 24,6, mediana de 21,0 e desvio padrão de 20,5 anos. Possuem segundo grau de escolaridade e são solteiros. Têm um

consumo predominante dos medicamentos: Dipirona, Prednisolona e Amoxicilina.

Grupo 3 - Composto por homens com idade média de 60,4, mediana de 64,0 e desvio padrão de 18,7 anos. Possuem ginásio/primeiro grau de escolaridade e são casados. Têm um consumo predominante dos medicamentos: AAS, Sinvastatina, Omeprazol, Furosemida e Losartana.

Grupo 4 - Grupo misto (homens e mulheres) com idade média de 58,0, mediana de 60,0 e desvio padrão de 19,5 anos. Possuem ginásio/primeiro e segundo grau de escolaridade e são casados. Têm um consumo predominante dos medicamentos: Norfloxacino, Clopidogrel e Ciprofloxacino.



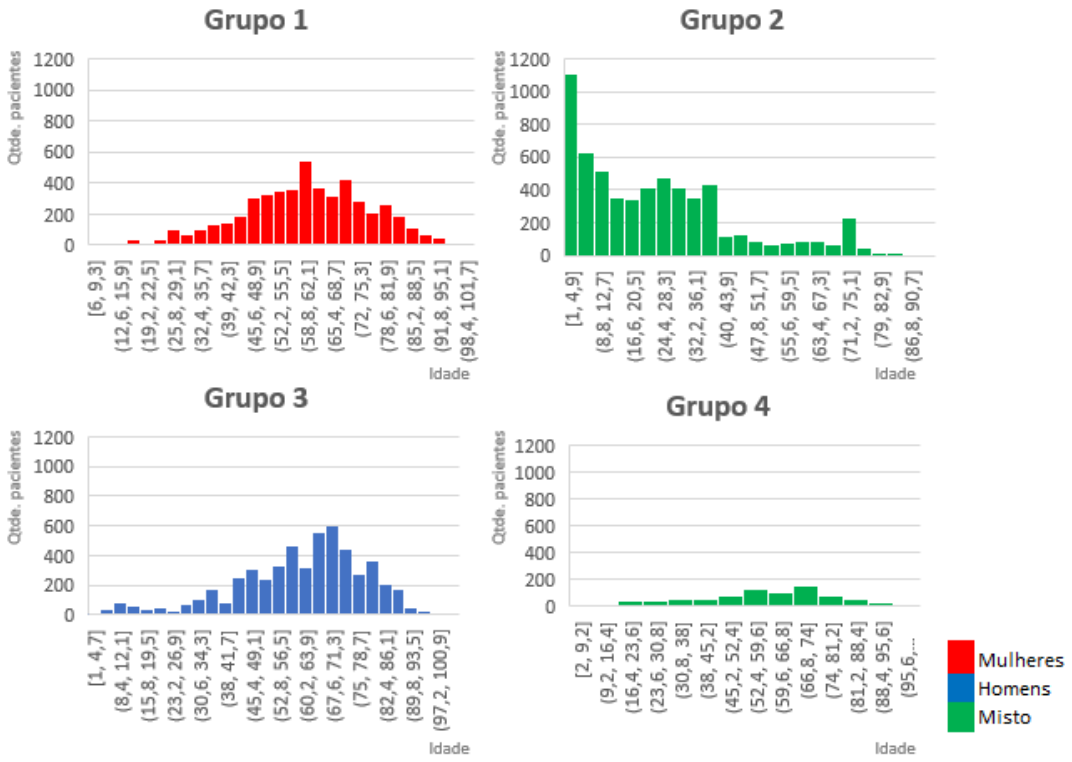


Figura 15. Boxplot e Histograma com estratificação por sexo e idade de cada grupo.

A partir do boxplot e dos histogramas acima, é mais fácil visualizar que o Grupo 2 é composto predominantemente por crianças, adolescentes e adultos. Já os demais grupos têm uma prevalência maior de idosos. Também é possível perceber que o Grupo 4 possui uma quantidade bem inferior de pacientes, se comparado aos demais.

Quando aprofundamos nas análises em relação ao medicamento AAS, o mais consumido pelos Grupos 2 e 3, percebemos algumas particularidades no perfil dos pacientes que fazem uso recorrente do mesmo. Primeiramente, observamos que 1.041 (52,5%) pacientes são do sexo masculino e 942 (47,5%) do sexo feminino. A idade média dos seus usuários é de 66,4 anos, mediana de 68,0 anos e desvio-padrão de 14,2 anos. Também é possível observar que uma quantidade significativa de pacientes iniciam sua utilização próximos dos 45,0 anos, independente do sexo. No entanto, mulheres tendem a iniciar o consumo com uma idade mais avançada em relação aos homens (após os 55,0 anos). Outro fator interessante é que 2,1% das mulheres consomem AAS antes dos 30,0 anos, diferente dos homens, com apenas 0,9%.

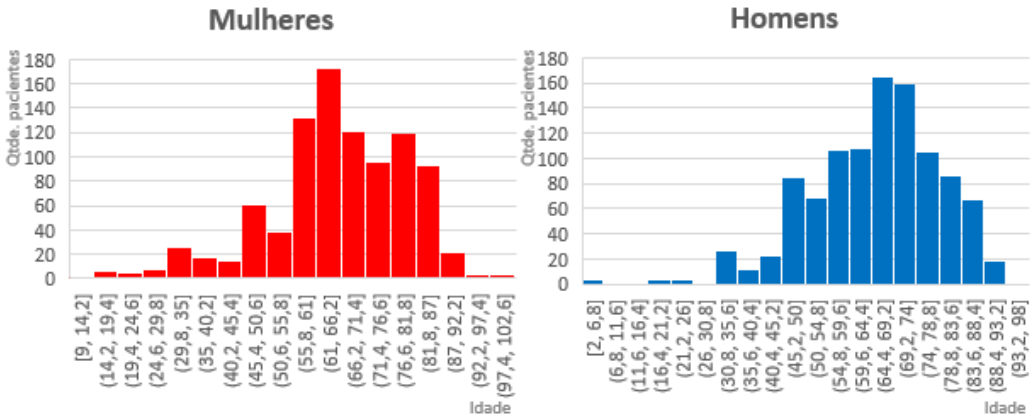


Figura 16. Histograma com estratificação por sexo e idade dos pacientes que consomem AAS.

5 Conclusão

Neste trabalho, construímos um modelo de extração de dados por meio de Reconhecimento de Entidades Nomeadas, utilizando Redes Neurais Convolucionais através da biblioteca *spaCy* em *Python*. Também desenvolvemos um corpus semanticamente anotado usando textos clínicos em português de várias especialidades médicas. Desta forma, conseguimos demonstrar que é possível treinar modelos para extrair informações de textos clínicos hospitalares.

Foi evidenciado como a utilização de NER em dados clínicos realmente é uma ferramenta poderosa na fomentação de estudos, principalmente epidemiológicos, pois consegue extrair informações que não existem em campos estruturados em bases de dados de prontuários médicos.

O modelo foi elaborado levando em consideração 6 classes de entidades presentes em prontuários médicos de complexidade significativa de extração e é possível observar resultados satisfatórios apresentados na seção 4.4 , atingindo 72,7% de *precision*.

Também é possível observar que, com o aumento da base de treinamento de 200 para 1.200 prontuários, obtivemos melhores resultados mesmo com o desbalanceamento da quantidade de entidades na amostra. A classe Condição do Paciente chagou a atingir 90,3% de *precision*.

Os resultados dessa aplicação se mostraram relevantes e sustentam que estudos futuros podem ser realizados com o objetivo de traçar perfis epidemiológicos da população atendida, trazendo mais informações para os responsáveis por gerir os recursos, de forma que estes sejam melhor aplicados, evitando desperdícios ou escassez de medicamentos, de insumos ou de profissionais. Além disso, com a obtenção de mais informações sobre os atendimentos é possível preparar melhor o sistema de saúde para surtos epidemiológicos ou problemas sazonais, mitigando seus impactos.

Com a simulação dos hiperparâmetros foi possível evidenciar a convergência mais rápida da rede com a utilização da função *decaying* na determinação do *dropout* de unidades como técnica de regularização e a alteração não significativa nos resultados com a aplicação do *minibatch - size compounding*, além da quantidade ideal de iterações considerando a otimização do aprendizado da rede e seu tempo de execução. Também foram testados os otimizadores recomendados na literatura e o ADAM foi selecionado por sua melhor performance na situação em questão.

Ao realizar uma análise exploratória dos dados extraídos da classe Medicação, foi possível traçar o perfil dos 4 grupos que melhor descrevem o comportamento dos pacientes em relação aos

medicamentos consumidos ou prescritos pelo hospital, sendo o AAS o mais consumido por estes pacientes no período de análise. Também foi possível entender o comportamento destes pacientes que estavam utilizando o AAS, considerando os fatores idade e sexo. Essa demonstração prática reforçou que a aplicação de Análises Multivariadas nos dados extraídos é promissora revelando comportamentos e informações até então desconhecidas relacionadas com as Entidades aqui trabalhadas.

Este trabalho apresenta uma extração de informações de textos clínicos de língua portuguesa, contribuindo para a área médica com uma coleção de 1.200 textos clínicos, com entidades nomeadas anotadas manualmente que poderão ser utilizados em trabalhos futuros.

Como possíveis melhorias, recomenda-se adicionar mais anotadores de forma a elevar a qualidade e reduzir a parcialidade. Também é importante o treinamento constante do algoritmo para que seja continuamente aprimorado, além da retroalimentação do modelo com a perspectiva de outros profissionais da saúde. Além disso recomenda-se a construção de outras análises estatísticas considerando as demais classes de entidades já extraídas neste trabalho, por fornecer respostas probabilísticas que permitem traçar perfis, entender comportamentos ou até predizer doenças ou diagnósticos, além de ser possível identificar doenças mais prevalentes, possibilitando tratamentos preventivos.

Referências

AIHUB, T. M. *Named Entity Recognition using Spacy and Tensorflow*. 2020. Disponível em: <https://aihub.cloud.google.com>. 7, 8

ANANIADOU, S.; KELL, D. B.; TSUJII, J.-i. Text mining and its potential applications in systems biology. *Trends in biotechnology*, Elsevier, v. 24, n. 12, p. 571–579, 2006. 1

ARANHA, C.; PASSOS, E. A tecnologia de mineração de textos. *Revista Eletrônica de Sistemas de Informação*, v. 5, n. 2, 2006. 2

BAKER, R. S. J. D. Data mining for education. *International encyclopedia of education*, Elsevier Oxford, UK, v. 7, n. 3, p. 112–118, 2010. 26

BALDI, P. Autoencoders, unsupervised learning, and deep architectures. In: *Proceedings of ICML workshop on unsupervised and transfer learning*. [S.l.: s.n.], 2012. p. 37–49. 6

BERKHIN, P.; BECHER, J. D. Learning simple relations: Theory and applications. In: SIAM. *Proceedings of the 2002 SIAM International Conference on Data Mining*. [S.l.], 2002. p. 420–436. 26

BISHOP, M.; LOONEY, C. 006.4—computer systems. pattern recognition. *The British National Bibliography*, British Library, Bibliographic Services Division., v. 1, p. 30, 1998. 6

BONNIN, R. *Building Machine Learning Projects with TensorFlow*. [S.l.]: Packt Publishing Ltd, 2016. 12, 17

BOUREAU, Y.-L.; PONCE, J.; LECUN, Y. A theoretical analysis of feature pooling in visual recognition. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. [S.l.: s.n.], 2010. p. 111–118. 13, 22

BRAGA, L. P. V. B. *Introdução à Mineração de Dados-2a edição: Edição ampliada e revisada*. [S.l.]: Editora E-papers, 2005. 5

BURKE, D. S. et al. Measurement of the false positive rate in a screening program for human immunodeficiency virus infections. *New England Journal of Medicine*, Mass Medical Soc, v. 319, n. 15, p. 961–964, 1988. 23, 24

CASTRO, L. N. d.; FERRARI, D. G. Introdução à mineração de dados: conceitos básicos, algoritmos e aplicações. *São Paulo: Saraiva*, 2016. 5

CHEN, D.; MANNING, C. D. A fast and accurate dependency parser using neural networks. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. [S.l.: s.n.], 2014. p. 740–750. 13, 14

CHEN, H. et al. A textual database/knowledge-base coupling approach to creating computer-supported organizational memory. *MIS Department, University of Arizona*, v. 5, 1994. 4

CHIEU, H. L.; NG, H. T. Named entity recognition with a maximum entropy approach. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. [s.n.], 2003. p. 160–163. Disponível em: <https://www.aclweb.org/anthology/W03-0423>. 9, 10

COSTA, H. G. Modelo para webibliomining: proposta e caso de aplicação. *Revista da FAE*, v. 13, n. 1, p. 115–126, 2010. 5

DAWAR, K.; SAMUEL, A. J.; ALVARADO, R. Comparing topic modeling and named entity recognition techniques for the semantic indexing of a landscape architecture textbook. In: IEEE. *2019 Systems and Information Engineering Design Symposium (SIEDS)*. [S.l.], 2019. p. 1–6. 9

DENG, L.; YU, D. Deep learning: methods and applications. *Foundations and Trends in Signal Processing*, Now Publishers, Inc., v. 7, n. 3–4, p. 197–387, 2014. 6, 7

EFRON, B. *The jackknife, the bootstrap and other resampling plans*. [S.l.]: SIAM, 1982. 25, 26

FALCÃO, A. E. J. et al. Indecs: método automatizado de classificação de páginas web de saúde usando mineração de texto e descritores em ciências da saúde (decs). *Journal of Health Informatics*, v. 1, n. 1, 2009. 1

GOODFELLOW, I. et al. *Deep learning*. [S.l.]: MIT press Cambridge, 2016. v. 1. 5, 7, 14, 15, 16, 17

GOTH, G. Analyzing medical data. *Communications of the ACM*, ACM New York, NY, USA, v. 55, n. 6, p. 13–15, 2012. 1

GUPTA, D. Fundamentals of deep learning–introduction to recurrent neural networks. *Analytics Vidhya (2107)*, 2017. 14, 15, 17

HAYKIN, S. S. *Neural Networks and learning machines/Simon Haykin.* [S.l.]: New York: Prentice Hall,, 2009. 5, 6

HCFMB. *HOSPITAL DAS CLINICAS DA FACULDADE DE MEDICINA DE BOTUCATU (HCFMB)*. 2019. Disponível em: <http://www.hcfmb.unesp.br/>. 29

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. 22, 23

HEARST, M. A. Untangling text data mining. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. [S.l.], 1999. p. 3–10. 4

HINTON, G.; OSINDERO, S. The, y. 2006, a fast learning algorithm for deep belief nets. *Neural computation*, v. 18, n. 7, 2006. 6, 33

JONES, M. T. *Arquiteturas de aprendizado profundo: O surgimento da inteligência artificial*. 2017. Disponível em: <http://www.ibm.com/developerworks/br/library/cc-machine-learning-deeplearning-architectures/cc-machine-learning-deep-learning-architectures-pdf.pdf/>. 7

JURAFSKY, D.; MARTIN, J. H. Speech and language processing (draft). *Chapter A: Hidden Markov Models (Draft of September 11, 2018). Retrieved March*, v. 19, p. 2019, 2018. 9, 10

JUSOH, S.; ALFAWAREH, H. M. Techniques, applications and challenging issue in text mining. *International Journal of Computer Science Issues (IJCSI)*, International Journal of Computer Science Issues (IJCSI), v. 9, n. 6, p. 431, 2012. 4

KASSAMBARA, A. *Practical guide to cluster analysis in R: Unsupervised machine learning*. [S.l.]: Sthda, 2017. v. 1. 26, 27, 28

KAUFMAN, L.; ROUSSEEUW, P. J. *Finding groups in data: an introduction to cluster analysis*. [S.l.]: John Wiley & Sons, 2009. v. 344. 26, 27, 28

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 21

KOHANE, I. S. Using electronic health records to drive discovery in disease genomics. *Nature Reviews Genetics*, Nature Publishing Group, v. 12, n. 6, p. 417–428, 2011. 1

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, AcM New York, NY, USA, v. 60, n. 6, p. 84–90, 2017. 13, 14

KULKARNI, T. D. et al. Deep convolutional inverse graphics network. *Advances in neural information processing systems*, v. 28, p. 2539–2547, 2015. 11

LAFFERTY, J.; MCCALLUM, A.; PEREIRA, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001. 10

LEOPOLD, H. et al. Using hidden markov models for the accurate linguistic analysis of process model activity labels. *Information Systems*, Elsevier, v. 83, p. 30–39, 2019. 9

LI, J. et al. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, 2020. 8, 23

LOH, S.; GARIN, R. S. Web intelligence–inteligência artificial para descoberta de conhecimento na web. *Oficina de Inteligência Artificial*, v. 5, p. 11–34, 2001. 5

LOPES, F.; TEIXEIRA, C.; OLIVEIRA, H. G. Contributions to clinical named entity recognition in Portuguese. In: *Proceedings of the 18th BioNLP Workshop and Shared Task*. Florence, Italy: Association for Computational Linguistics, 2019. p. 223–233. Disponível em: <https://www.aclweb.org/anthology/W19-5024>. 8

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: *Proc. icml*. [S.l.: s.n.], 2013. v. 30, n. 1, p. 3. 13, 14

MAIA, L. B. et al. Evaluation of melanoma diagnosis using deep features. In: IEEE. *2018 25th international conference on systems, signals and image processing (IWSSIP)*. [S.l.], 2018. p. 1–4. 11

MALI, M.; ATIQUE, M. Applications of text classification using text mining. *International Journal of Engineering Trends and Technology*, v. 13, n. 5, p. 209, 2014. 5

MCCALLUM, A.; LI, W. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. 2003. 11

MONTAVON, G.; SAMEK, W.; MÜLLER, K.-R. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, Elsevier, v. 73, p. 1–15, 2018. 2

MURUGAVEL, M. *Spacy Annotation Tool*. 2020. Disponível em: <https://manivannanmurugavel.github.io/annotating-tool/spacy-ner-annotator/>. 31, 32

NETO, J. F. d. S. Reconhecimento de entidades nomeadas para o português usando redes neurais. Pontifícia Universidade Católica do Rio Grande do Sul, 2019. 25

NIELSEN, M. A. *Neural Networks and deep learning*. [S.l.]: Determination press San Francisco, CA, USA:, 2015. v. 2018. 6

OLSON, D. L.; DELEN, D. Performance evaluation for predictive modeling. In: *Advanced data mining techniques*. [S.l.]: Springer, 2008. p. 137–147. 23, 24, 25, 26

PAKHOMOV, S. et al. Electronic medical records for clinical research: application to the identification of heart failure. *Am J Manag Care*, v. 13, n. 6 Part 1, p. 281–288, 2007. 2

PEI, W.; GE, T.; CHANG, B. An effective neural network model for graph-based dependency parsing. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. [S.l.: s.n.], 2015. p. 313–322. 13, 14

PEISSIG, P. L. et al. Importance of multi-modal approaches to effectively identify cataract cases from electronic health records. *Journal of the American Medical Informatics Association*, BMJ Group BMA House, Tavistock Square, London, WC1H 9JR, v. 19, n. 2, p. 225–234, 2012. 2

PERES, R. Algoritmo back-propagation. Revista Programar, 2017. 5

PETERS, A. C. et al. Semclinbr–a multi institutional and multi specialty semantically annotated corpus for portuguese clinical nlp tasks. *arXiv preprint arXiv:2001.10071*, 2020. 8

PINTO, V. B. Prontuário eletrônico do paciente: documento técnico de informação e comunicação do domínio da saúde 10.5007/1518-2924.2006 v11n21p34. *Encontros Bibli: revista eletrônica de biblioteconomia e ciência da informação*, v. 11, n. 21, p. 34–48, 2006. 1

PIRES, A.; DEVEZAS, J.; NUNES, S. Benchmarking named entity recognition tools for portuguese. *Proceedings of the Ninth INForum: Simpósio de Informática*, p. 111–121, 2017. 24

PODANI, J.; SCHMERA, D. On dendrogram-based measures of functional diversity. *Oikos*, Wiley Online Library, v. 115, n. 1, p. 179–185, 2006. 26

PONTI, M. A.; COSTA, G. B. P. da. Como funciona o deep learning. *arXiv preprint arXiv:1806.07908*, 2018. 5

PYPI. *Python Package Index (PyPI)*. 2021. Disponível em: <https://pypi.org/>. 31

RABINER, L.; JUANG, B. An introduction to hidden markov models. *ieee assp magazine*, IEEE, v. 3, n. 1, p. 4–16, 1986. 9

RAMACHANDRAN, P.; ZOPH, B.; LE, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 14, 15, 16, 17

RASCHKA, S.; MIRJALILI, V. *Python machine learning*. [S.l.]: Packt Publishing Ltd, 2017. 17

RATNAPARKHI, A. Maximum entropy models for natural language ambiguity resolution. 1998. 9, 10

REYNOLDS, A. P.; RICHARDS, G.; RAYWARD-SMITH, V. J. The application of k-medoids and pam to the clustering of rules. In: SPRINGER. *International Conference on Intelligent Data Engineering and Automated Learning*. [S.l.], 2004. p. 173–178. 27, 28

ROQUE, F. S. et al. Using electronic patient records to discover disease correlations and stratify patient cohorts. *PLoS computational biology*, Public Library of Science, v. 7, n. 8, 2011. 2

RUDER, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 17, 19, 20, 21, 22

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks*, Elsevier, v. 61, p. 85–117, 2015. 6

SCHNEIDER, E. T. R. et al. BioBERTpt - a Portuguese neural language model for clinical named entity recognition. In: *Proceedings of the 3rd Clinical Natural Language Processing Workshop*. Online: Association for Computational Linguistics, 2020. p. 65–72. Disponível em: <https://www.aclweb.org/anthology/2020.clinicalnlp-1.7>. 8

SETTLES, B. Biomedical named entity recognition using conditional random fields and rich feature sets. In: *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*. [S.l.: s.n.], 2004. p. 107–110. 10, 11

SHELAR, H. et al. Named entity recognition approaches and their comparison for custom ner model. *Science & Technology Libraries*, Taylor & Francis, v. 39, n. 3, p. 324–337, 2020. 8

SILVA, L. A. da; PERES, S. M.; BOSCARIOLI, C. *Introdução à mineração de dados: com aplicações em R*. [S.l.]: Elsevier Brasil, 2017. 5

SLATTON, T. G. A comparison of dropout and weight decay for regularizing deep neural networks. 2014. 33

SMITH, L. N. Cyclical learning rates for training neural networks. In: IEEE. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. [S.l.], 2017. p. 464–472. 19

SONG, M. Opinion: Text mining in the clinic. *The Scientist*, 2013. 1

SPACY. *Language Processing Pipelines*. 2021. Disponível em: <https://spacy.io/usage/processing-pipelines>. 33

SPASIC, I. et al. Text mining and ontologies in biomedicine: making sense of raw text. *Briefings in bioinformatics*, Henry Stewart Publications, v. 6, n. 3, p. 239–251, 2005. 1

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014. 22, 33

SULLIVAN, D. The need for text mining in business intelligence. *DM REVIEW*, POWELL PUBLISHING INC, v. 10, p. 12–16, 2000. 4

TAN, A.-H. Text mining: The state of the art and the challenges. In: SN. *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*. [S.l.], 1999. v. 8, p. 65–70. 4

THINC. *Loss Calculators*. 2021. Disponível em: <https://thinc.ai/docs/api-loss>. 33

THOMAS, J.; MCNAUGHT, J.; ANANIADOU, S. Applications of text mining within systematic reviews. *Research Synthesis Methods*, Wiley Online Library, v. 2, n. 1, p. 1–14, 2011. 5

THURAISINGHAM, B. *Data mining: technologies, techniques, tools, and trends*. [S.l.]: CRC press, 2014. 4

VYCHEGZHANIN, S.; KOTELNIKOV, E. Comparison of named entity recognition tools applied to news articles. In: IEEE. *2019 Ivannikov Ispras Open Conference (ISPRAS)*. [S.l.], 2019. p. 72–77. 8

WANG, Y.-Y.; DENG, L.; ACERO, A. Spoken language understanding. *IEEE Signal Processing Magazine*, IEEE, v. 22, n. 5, p. 16–31, 2005. 9

XU, R.; WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on neural networks*, Ieee, v. 16, n. 3, p. 645–678, 2005. 26, 27

ZEILER, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 19

ZHANG, Y.; WALLACE, B. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015. 11, 12, 13, 14, 22

ZHOU, S. K.; GREENSPAN, H.; SHEN, D. *Deep learning for medical image analysis*. [S.l.]: Academic Press, 2017. 6, 7

ZWEIGENBAUM, P. et al. Frontiers of biomedical text mining: current progress. *Briefings in bioinformatics*, Oxford University Press, v. 8, n. 5, p. 358–375, 2007. 1