UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Amanda Furtado Brinhosa

Estruturação automática de laudos médicos utilizando processamento de linguagem natural e aprendizado de máquina para extração de informações clínicas

Florianópolis
2020

Amanda Furtado Brinhosa

Estruturação automática de laudos médicos utilizando processamento de linguagem natural e aprendizado de máquina para extração de informações clínicas

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Eric Antonelo, Dr.
Supervisor: Nicholas Drabowski, Eng.

Florianópolis
2020

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Brinhosa, Amanda
Estruturação automática de laudos médicos utilizando processamento de linguagem natural e aprendizado de máquina para extração de informações clínicas / Amanda Brinhosa ; orientador, Eric Antonelo, 2020.
p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2020.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Aprendizado de máquina. 3. Laudos médicos. 4. Extração de informações clínicas. 5. Processamento de linguagem natural. I. Antonelo, Eric. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Amanda Furtado Brinhosa

Estruturação automática de laudos médicos utilizando processamento de linguagem natural e aprendizado de máquina para extração de informações clínicas

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de graduação em Engenharia de Controle e Automação

Florianópolis, 29 de julho de 2020.

Prof. Hector Bessa Silveira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Eric Aislan Antonelo, Dr.
Orientador - UFSC/CTC/DAS

Nicholas Roberto Drabowski, Eng.
Supervisor - Empresa Portal Telemedicina

Prof. Fabio Luis Baldissera, Dr.
Avaliador - UFSC/CTC/DAS

Prof. Fabio Luis Baldissera, Dr.
Presidente da Banca - UFSC/CTC/DAS

AGRADECIMENTOS

Aos meus pais, Márcia e Sérgio, e ao meu irmão Gustavo, agradeço por todo apoio, por sempre alimentarem meus sonhos e acreditarem no meu potencial.

Agradeço também à Portal Telemedicina e a todos os meus colegas de trabalho por confiarem no meu trabalho e por compartilharem o propósito de salvar vidas através da tecnologia. Sou grata especialmente ao meu supervisor Nicholas, que me incentivou, inspirou e ensinou tanto durante essa trajetória.

Ao meu orientador Eric, muito obrigada por todo o auxílio e por ter aceitado essa empreitada mesmo em tempos tão difíceis. Aproveito para agradecer a todos os professores que participaram da minha caminhada, pois com certeza cada um contribuiu para que eu chegasse até aqui com tantos aprendizados.

A todos os meus amigos e colegas de curso, os quais divido tantas memórias, minha gratidão. A jornada foi mais leve graças a vocês.

Por fim, agradeço à Universidade Federal de Santa Catarina por ter me proporcionado tantas experiências incríveis.

RESUMO

Este documento tem por finalidade apresentar uma abordagem para a estruturação automática de laudos médicos em português por meio da extração de informações clínicas utilizando processamento de linguagem natural (NLP) e aprendizado de máquina. Foram propostas duas arquiteturas, Bi-LSTM-CRF e CRF baseada em *features*, além da técnica de *bootstrapping*, utilizada para criação de conjunto de dados anotados a partir de um conjunto pequeno. As métricas *precision*, *recall* e *F1-score*, obtidas a partir do conjunto de teste, foram todas acima de 80%. As análises de robustez mostraram que a solução é promissora e poderá ser aprimorada. A explicabilidade dos modelos de CRF, que apresenta características que o algoritmo observou durante o treinamento, também auxiliou no entendimento do que foi realizado. As implementações são escaláveis e adaptáveis, não somente para novas modalidades médicas, mas também para outras tarefas dentro da empresa, o que não acontece com o microsserviço de estruturação atual, baseado em expressões regulares. Além disso, os resultados mostraram uma potencial otimização de tempo de desenvolvimento e custo em produção, simplificando a estrutura em nuvem vigente.

Palavras-chave: Aprendizado de máquina. Laudos médicos. Extração de informações clínicas. Processamento de linguagem natural.

ABSTRACT

This document presents an approach for automatic structuring of medical reports in portuguese to extract clinical information using natural language processing (NLP) and machine learning. Two architectures were proposed, Bi-LSTM-CRF and Feature-based CRF, in addition to the bootstrapping technique, used to create annotated data sets from a small set. The metrics precision, recall and F1-score, obtained from the test set, were all above 80%. The robustness analysis showed that the solution is promising and could be improved. The explainability for CRF models, that presents characteristics that the model learned during the training process, also helped to understand what was done. The implementations are scalable and adaptable, not only for new medical modalities, but also for other tasks within the company, which is not the case with the current structuring microsservice that uses regular expressions. Furthermore, the results presented a potential optimization in development time and in operational production cost, simplifying the current cloud structure.

Keywords: Machine learning. Medical reports. Clinical information extraction. Natural language processing.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AILA	*Artificial Intelligence for Life Analytics*
IA	Inteligência Artificial
NLP	*Natural Language Processing*
FHIR	*Fast Healthcare Interoperability Resources*
GCP	*Google Cloud Platform*
RNN	*Recurrent Neural Networks*
LSTM	*Long Short-Term Memory*
CRF	*Conditional Random Fields*
Bi-LSTM-CRF	*Bidirectional LSTM-CRF*
BERT	*Bidirectional Enconder Representations from Transformers*
IHTSDO	*International Health Terminology Standards Development Organization*
CID	Classificação Internacional de Doenças
NLTK	*Natural Language Toolkit*
IOB	*Inside, outside, beginning*
SNOMED-CT	*Systematized Nomenclature of Medicine Clinical Terms*
RMSProp	*Root Mean Square Propagation*
Adam	*Adaptive Moment Estimation*
L-BFGS	*Limited-memory Broyden-Fletcher-Goldfarb-Shanno*
L2-SGD	*Stochastic Gradient Descent with L2 regularization*
AP	*Averaged Perceptron*
PA	*Passive Aggressive*
AROW	*Adaptive Regularization Of Weight Vector*
TP	*True Positives* ou Verdadeiros Positivos
FP	*False Positives* ou Falsos Positivos

FN	*False Negatives* ou Falsos Negativos
TN	*True Negatives* ou Verdadeiros Negativos
CSV	*Comma-separated values*
BOS	*Beginning of sentence* ou Início da Sentença
EOS	*End of sentence* ou Final da Sentença

SUMÁRIO

1 INTRODUÇÃO

É de conhecimento geral que, atualmente, a quantidade de dados disponíveis cresce significativamente a cada ano. Com isso, diversas oportunidades de negócio estão surgindo e as empresas que conseguem utilizar essas informações de modo inteligente se destacam dos concorrentes. Porém, para que se possa gerar valor a partir dos dados, os mesmos precisam estar formatados e estruturados.

Na área da saúde, a quantidade de dados não estruturados, mas disponíveis, torna as análises ainda mais delicadas. Pode-se citar, por exemplo, que uma grande parte das prescrições médicas no Brasil continuam sendo escritas à mão, o que prejudica o levantamento de indicadores para guiar decisões de gestão em saúde. Outro ponto é que, além da falta de interconectividade entre os equipamentos, existe ainda a falta de interoperabilidade entre os mais variados sistemas (NACIONAL, 2020), ou seja, há a dificuldade dos mesmos se comunicarem entre si de maneira transparente. Na maioria das vezes, as informações de um paciente estão espalhadas em múltiplas fontes diferentes.

Apesar das dificuldades citadas, muitas empresas estão buscando soluções para lidar com o problema de interoperabilidade. A integração entre bancos de dados e sistemas está sendo efetivamente realizada através de protocolos e padrões, como o *Fast Healthcare Interoperability Resources* (FHIR), *Systematized Nomenclature of Medicine Clinical Terms* (SNOMED-CT) e outros.

Todavia, mesmo com todos os avanços comentados, ainda existem milhares de dados médicos armazenados e disponíveis, porém não estruturados. Nesta área, uma grande oportunidade de pesquisa refere-se aos laudos de exames. A maior parte hoje está em texto livre, sem padronizações, apesar do vocabulário específico.

Neste contexto, a empresa Portal Telemedicina, que tem o propósito de unir a tecnologia e saúde para solucionar o acesso restrito à medicina de qualidade, traz à tona a necessidade de organizar e aproveitar ao máximo os dados que têm disponíveis. Até o presente momento, a Portal possui um microsserviço[1] de estruturação de laudos do tipo especialista, que utiliza um modelo com expressões regulares[2] para detectar padrões nos textos. O mesmo foi concretizado no âmbito da disciplina de Projeto Integrador, pela própria autora deste trabalho em conjunto com outros dois graduandos do curso, e posteriormente melhorado e integrado com o sistema de telemedicina da empresa.

No entanto, existem diversos pontos a serem levados em consideração. Um microsserviço especialista demanda que uma equipe experiente trabalhe nas várias

[1] Microsserviços são uma abordagem de arquitetura de software que definem serviços individuais, ou seja, possuem uma função específica, bem definida e independente. Geralmente compõem grandes e complexas aplicações.

[2] Expressões Regulares são padrões de caracteres que associam sequências de caracteres no texto. Podem ser usadas, por exemplo, para identificar uma dada palavra em uma frase.

áreas que o compõe e uma cautela ainda maior, visto que as expressões regulares criadas exigem um conhecimento a priori dos padrões de escrita dos laudos. Por esse motivo, pode-se afirmar que ele possui um formato pouco flexível, dificultando a manutenção e, por não ser generalizável, ele deve ser repensado para cada modalidade médica nova.

O projeto de fim de curso desenvolvido se encaixa no contexto da resolução destes problemas através da utilização de técnicas avançadas de processamento de linguagem natural (NLP), tornando possível aprofundar o conhecimento sobre a modalidade em análise e detectando novos padrões entre os achados médicos descritos nos laudos. Muitos outros microsserviços poderiam ser beneficiados ou derivados desta abordagem, incluindo a expansão da base de dados compatível com os algoritmos de IA da empresa.

Sob o ponto de vista da sociedade, podem-se citar como benefícios a maior rapidez e acurácia na entrega dos resultados de exames, promovendo uma melhoria dos índices de desempenho relacionados à saúde e reduzindo o custo de laudos médicos. Para os profissionais de saúde destacam-se os ganhos em comodidade, agilidade no recebimento de laudos, além da redução de erros de diagnóstico e, consequentemente, de tratamento.

No caso do microsserviço de estruturação de laudos atual, são analisados exames de eletrocardiograma, eletroencefalograma e espirometria de forma bastante simplificada. Uma vez que está sendo desenvolvido pela empresa um projeto para detecção e acompanhamento de nódulos cancerígenos nas modalidades de raio-X e tomografia, é imprescindível que modelos de NLP, por serem mais flexíveis e possuírem maior potencial que expressões regulares para descobrir novos padrões automaticamente, sejam criados e agregados o mais rapidamente possível, auxiliando nessa tarefa.

Também há uma grande relevância do projeto no contexto da pandemia de COVID-19. Laudos de radiografia possuem muitas informações que precisam ser extraídas e combinadas para serem utilizadas como suporte à criação de novas soluções.

Uma vez que o curso de Engenharia de Controle e Automação promove a interdisciplinaridade, acredita-se que este trabalho poderá contribuir significativamente na área de saúde. No contexto apresentado, as habilidades adquiridas nas disciplinas de inteligência artificial, banco de dados, programação, estatística, fundamentos de biomédica e metodologia de sistemas são relevantes.

1.1 OBJETIVOS

O objetivo geral deste projeto de fim de curso é desenvolver e avaliar algoritmos de estruturação automática de laudos médicos de exames de raio-X baseado em NLP juntamente com um processo que o torne facilmente adaptável para novas modalidades

médicas futuramente. Tem-se como objetivos específicos, incluindo aspectos técnicos e de negócio, para este projeto:

- tornar o microsserviço de estruturação de laudos atual mais flexível empregando modelos de NLP, substituindo as expressões regulares;
- tornar o microsserviço de estruturação de laudos atual facilmente expansível para novas modalidades médicas e melhorar e atualizar as que já estão implementadas;
- definir quais são as métricas adequadas de aprendizado de máquina e seus respectivos valores, quando aplicáveis (por exemplo, modelos com acurácia maior que 80%);
- definir métricas adequadas de pesquisa e desenvolvimento dentro da empresa;

1.2 ESTRUTURA DO DOCUMENTO

Este relatório está divido em seis capítulos. No capítulo 2 é feita uma contextualização mais detalhada sobre o problema e a empresa. No capítulo 3 é apresentada a revisão da literatura em conjunto com a fundamentação teórica. Já o capítulo 4 trata dos métodos de resolução do problema, sendo a implementação efetiva apresentada no capítulo 5. Por fim, tem-se os capítulos 6 e 7 que cobrem os resultados, discussão e conclusões.

2 CONTEXTUALIZAÇÃO

2.1 A PORTAL TELEMEDICINA

A Portal Telemedicina foi fundada em 2013, com a missão de salvar vidas e garantir o acesso universal à medicina. Em vista da necessidade de uma mudança na saúde brasileira, a Portal oferece uma solução completa de telemedicina, conectando os melhores especialistas a clínicas e hospitais. Laudos à distância são realizados em poucos minutos e por um valor acessível, possibilitando que até mesmo lugares remotos tenham acesso à medicina de qualidade.

A plataforma de telediagnóstico é integrada aos equipamentos médicos por meio de multiprotocolos de comunicação e internet das coisas, facilitando a operação, uma vez que os dados são enviados automaticamente para o sistema central de dados. Além disso, a inteligência artificial é uma grande aliada nesta operação, empoderando médicos a laudar com mais rapidez e precisão.

Atualmente estas soluções já estão presentes em centenas de cidades brasileiras e também na África. Os clientes ativos fazem parte de diferentes segmentos, como clínicas, medicina ocupacional, saúde corporativa, hospitais, assistência domiciliar, unidades móveis, convênios, empresas/indústrias e médicos especialistas. Com tanto impacto, a empresa tem conseguido grandes parcerias (incluindo o Google).

2.2 *ARTIFICIAL INTELLIGENCE FOR LIFE ANALYTICS* (AILA)

A plataforma de Inteligência Artificial (IA) denominada *Artificial Intelligence for Life Analytics* ou AILA, desempenha um importante papel no serviço de telemedicina fornecido pela empresa. Os modelos de IA atuais realizam um complexo processamento das imagens dos exames e retornam os achados médicos detectados e suas probabilidades (por exemplo, chance de 90% de um dado exame de raio-X apresentar cardiomegalia) e também a gravidade total do exame.

Dependendo do valor de gravidade retornado, o exame do paciente em questão pode ser priorizado na fila para ser laudado. Todas estas informações retornadas pelos modelos de IA auxiliam não somente o médico que está laudando, mas também facilitam e agilizam o trabalho dos profissionais que estão cuidando do paciente presencialmente.

Localidades em que não existem médicos de uma determinada modalidade ou especialidade, antes de utilizarem o sistema de telediagnóstico da Portal, levavam em média 15 dias para receberem um laudo do exame realizado. Após a implementação do sistema, esse tempo caiu para cerca de 10 minutos.

No momento presente, a plataforma está bem consolidada para o processamento de imagens. Os principais modelos hoje são para detecção de exame alterado

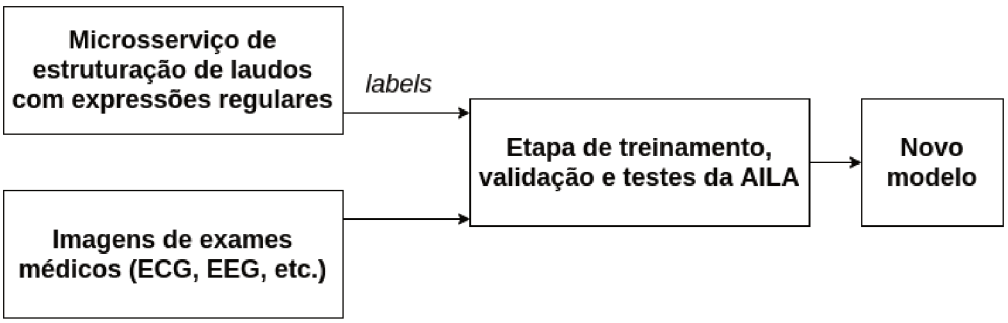
(normal ou anormal) e qualidade técnica do exame realizado (se está em boas condições para ser laudado), sendo que cada modalidade possui o seu modelo específico.

Para a criação de novos modelos, como o treinamento dos algoritmos é do tipo supervisionado, é necessário ter um conjunto de treino, teste e validação com seus respectivos rótulos ou *labels* (0 para o achado médico, como a cardiomegalia, que esteja ausente e 1 para presente). A única forma de tornar isso possível seria extraindo as informações do texto de laudo associado à imagem de exame.

2.3 MICROSSERVIÇO DE ESTRUTURAÇÃO DE LAUDOS ATUAL

Para solucionar o problema da extração de *labels* para o treinamento supervisionado na AILA, foi criado um microsserviço de estruturação de laudos. O mesmo foi implementado no contexto da disciplina de Projeto Integrador com outros dois graduandos do curso e posteriormente validado por especialistas e integrado ao sistema de telediagnóstico completo através da *Google Cloud Platform* (GCP). Na Figura 1 é apresentado o fluxo de trabalho simplificado que é utilizado para geração de novos modelos de IA através da AILA.

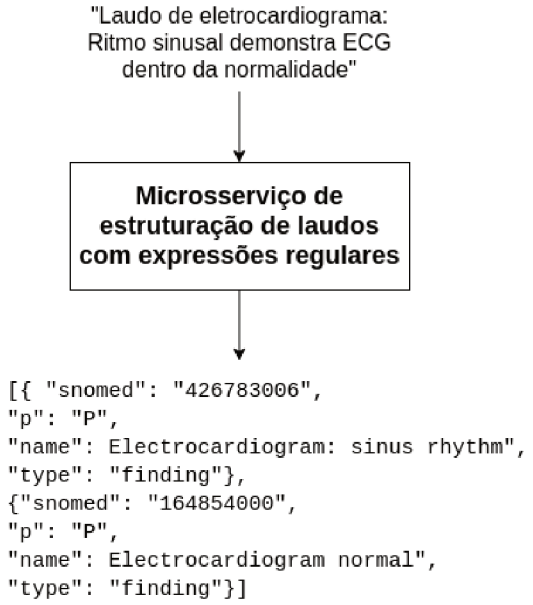
Figura 1 – Fluxo de trabalho simplificado de treinamento, validação e testes da AILA para geração de novos modelos de IA para imagens médicas.



Fonte – Elaborada pela autora (2020).

Deve-se entender como **estruturação** a tarefa do microsserviço de receber um laudo em texto livre e retornar os achados médicos encontrados no mesmo, utilizando a codificação internacional SNOMED-CT. Na Figura 2 tem-se um exemplo para um laudo fictício de eletrocardiograma. A resposta retornada consiste em uma lista de dicionários com: código do achado de acordo com a SNOMED-CT; estado (*p*) do achado no laudo, sendo "P"= presente, "C"= incerto (*clue*) ou "N"= negado ("ausência de"); nome em inglês de acordo com a SNOMED-CT e o tipo, que pode ser um achado médico da categoria *finding* ou *condition* ou ainda *product* (dispositivos, materiais, próteses) ou *procedure* (procedimentos).

Figura 2 – Exemplo de estruturação de um laudo fictício de eletrocardiograma com dois códigos SNOMED-CT identificados.



Fonte – Elaborada pela autora (2020).

A média de laudos é de aproximadamente 3000 por dia, um número que continua crescendo devido à expansão da solução da empresa. Com isso, torna-se cada vez mais relevante e urgente a estruturação destes laudos.

O microsserviço atual trabalha apenas com algumas modalidades médicas e por não ser facilmente escalável, dificulta a sua expansão. Apesar disso, acredita-se que o mesmo irá auxiliar no desenvolvimento do estruturador automático, uma vez que já se tem um prévio conhecimento da tarefa.

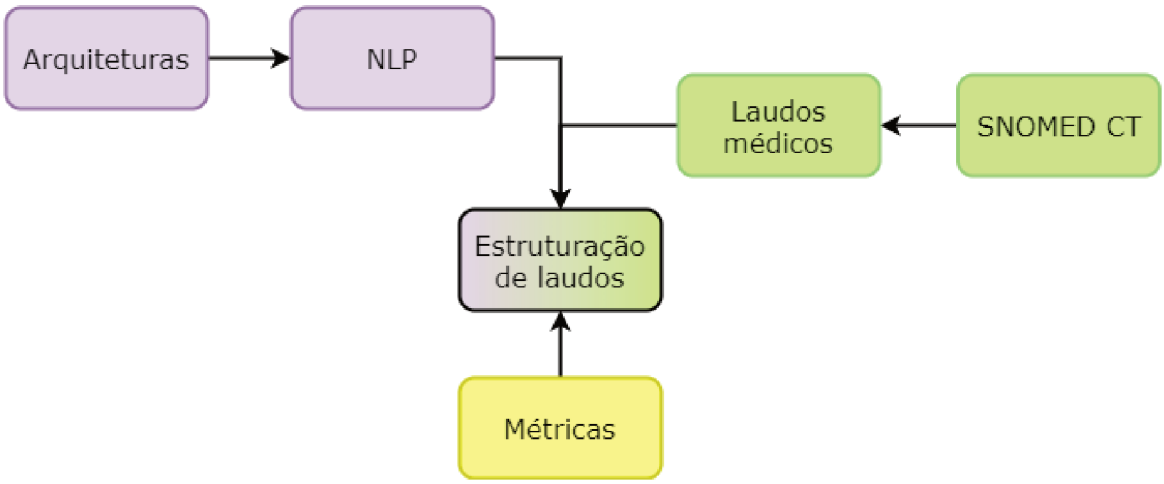
Uma dificuldade a ser enfrentada diz respeito à definição de métricas, já que o campo que envolve NLP ainda não foi explorado na empresa. Além das análises clássicas de aprendizado de máquina, é desejado também saber e estimar quantas horas são necessárias para realizar este tipo de projeto, quantos profissionais precisam ser envolvidos e qual a facilidade de uso e integração em detrimento do custo (métricas de pesquisa e desenvolvimento citadas nos objetivos específicos).

Ademais, a área de inteligência artificial possui um grande potencial para trabalhar com dados mais diversificados (fontes e contextos diferentes), porém possui também muitos desafios quando precisa levar em conta regras de negócio. Trabalhando com metodologias ágeis, modelos de IA têm um fluxo de aprimoramento iterativos até que sejam alcançados bons resultados para serem utilizados de fato. Dessa forma, o objetivo deste PFC está alinhado com estas propostas, fazendo com que dados textuais também possam contribuir com a missão da Portal Telemedicina.

3 FUNDAMENTAÇÃO TEÓRICA E REVISÃO DA LITERATURA

Neste capítulo, são apresentados os aspectos conceituais que compõem o presente projeto de fim de curso. A fundamentação teórica, apoiada pela revisão da literatura, é guiada pelo diagrama mostrado na Figura 3. Cada cor representa uma área de estudo diferente, sendo que a estruturação de laudos é a meta final que liga todos os conceitos apresentados de acordo com os objetivos especificados na Introdução.

Figura 3 – Diagrama de dependência de conceitos



Fonte – Elaborada pela autora (2020).

3.1 PROCESSAMENTO DE LINGUAGEM NATURAL (NLP)

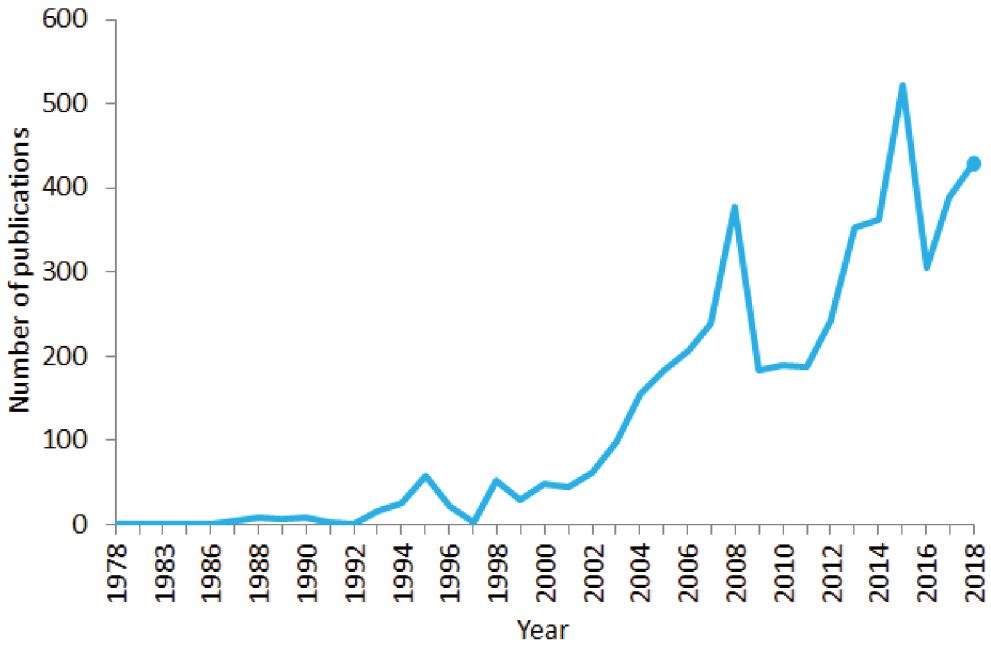
O processamento de linguagem natural, geralmente referido como NLP (do inglês, *Natural Language Processing*), é a parte da ciência da computação, inteligência artificial e linguística que estuda como as máquinas podem compreender, interpretar e manipular as linguagens naturais (CHOWDHURY, 2005). Suas principais problemáticas envolvem geração e compreensão, tanto de texto quanto de fala.

É uma área que tem ganhado cada vez mais visibilidade, porém teve seu destaque com o conhecido Teste de Turing. O mesmo foi introduzido em 1950 por Alan Turing e consiste em verificar a capacidade de uma máquina de exibir um comportamento inteligente equivalente ao de um ser humano. A forma clássica do teste consiste em um jogador humano fazendo perguntas através de um computador sem saber se está falando com outro humano ou uma máquina.

Existe uma diversidade de trabalhos que envolvem NLP, uma vez que há diversos idiomas e dialetos pelo mundo. As aplicações podem ser voltadas para a análise de sentimentos, comumente associadas às redes sociais (WIKIPEDIA, s.d.[b]), (JI *et al.*, 2013), reconhecimento de fala (LUOEN, 2016), *chatbots* (ORACLE, s.d.), entre outros.

O campo de processamento de linguagem natural tem crescido significativamente, principalmente no contexto da indústria de saúde. Através da Figura 4 é possível ter uma ideia do quanto se tem discutido esse assunto. No presente projeto, será estudada a extração de informações clínicas contidas em textos de laudos médicos em português.

Figura 4 – Número de publicações no motor de busca PubMed no período de 1978 a 2018 contendo a expressão "*natural language processing*".



Fonte – (YSE, 2019)

3.1.1 Arquiteturas

Atualmente, existem diversas soluções para problemas que envolvem processamento de linguagem natural. Nos subtópicos a seguir são apresentadas algumas das arquiteturas de IA aplicadas nesta área, suas características principais e funcionamento básico.

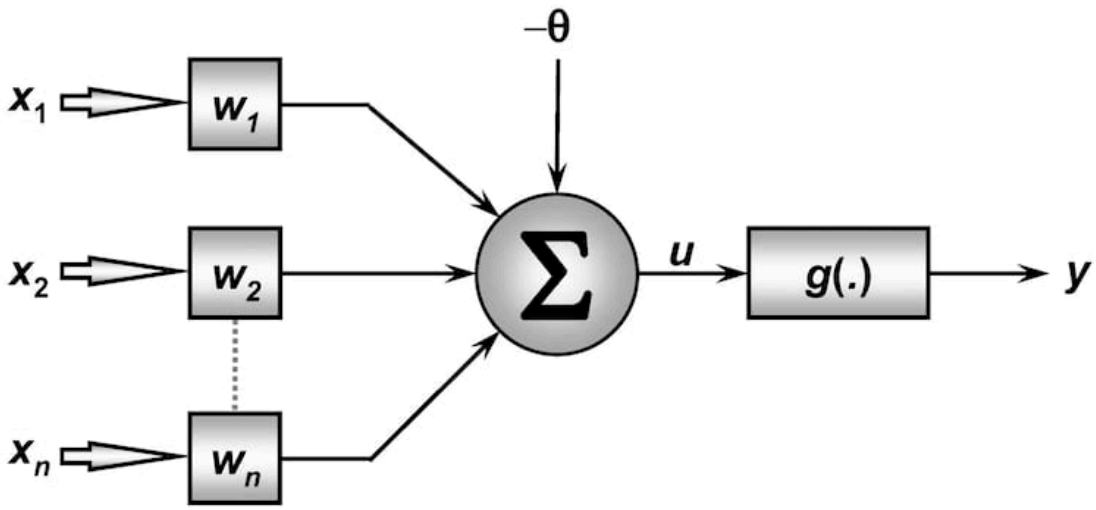
3.1.1.1 Redes neurais artificiais (RNA)

As RNA são um modelo computacional inspirado no cérebro. Possuem a capacidade de realizar o aprendizado de máquina e reconhecer padrões através de um conjunto de neurônios artificiais que captam, processam e distribuem informações.

No tempo presente, existem muitas variações das RNA. Na Figura 5, para fins de entendimento, é ilustrado um de vários neurônios que podem compor uma rede.

Os sinais de entrada, representados pelo conjunto $\{x_1, x_2, ..., x_n\}$, são somados de forma ponderada pelos pesos $\{w_1, w_2, ..., w_n\}$ resultando em um potencial de ativação u. Também faz parte do somatório o elemento θ, que serve para aumentar o grau de liberdade dos ajustes dos pesos (ZAMBIASI, 2011).

Figura 5 – Ilustração de um neurônio componente de uma rede neural artificial.



Fonte – (NET, 2019)

O bloco $g(.)$ corresponde à função de ativação, que define a saída da rede. Tal função pode ser degrau, sigmoide, tangente hiperbólica, entre outras. Pode-se descrever este funcionamento, matematicamente, considerando uma função de ativação linear, como nas equações 1 e 2.

$$u = \sum_{i=0}^{n} x_i w_i - \theta. \tag{1}$$

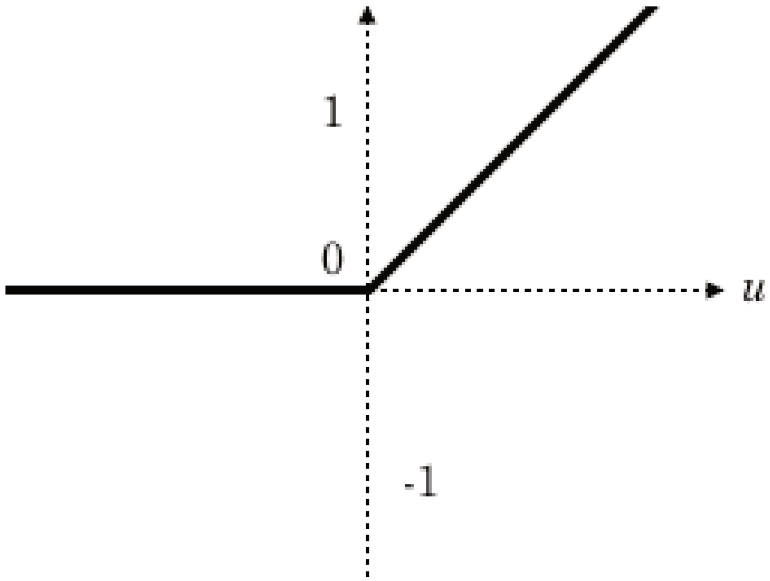
$$y = g(u) = u. \tag{2}$$

Uma função de ativação que vale destacar é a ReLU (unidade linear retificada). Ela é definida como na equação 3 e aplica-se à maioria dos problemas.

$$g(u) = max(0, u). \tag{3}$$

Além de ser não linear, possui a vantagem de ser eficiente e fácil de ser computada, o que evita problemas com o cálculo do gradiente, uma vez que os neurônios não são todos ativados ao mesmo tempo. A Figura 6 ilustra esse comportamento, mostrando que, quando a entrada da função for negativa, ela será convertida em zero e o neurônio não será ativado.

Figura 6 – Função de ativação *ReLU*.



Fonte – (DINIZ, s.d.)

3.1.1.1.1 Redes neurais recorrentes (RNN)

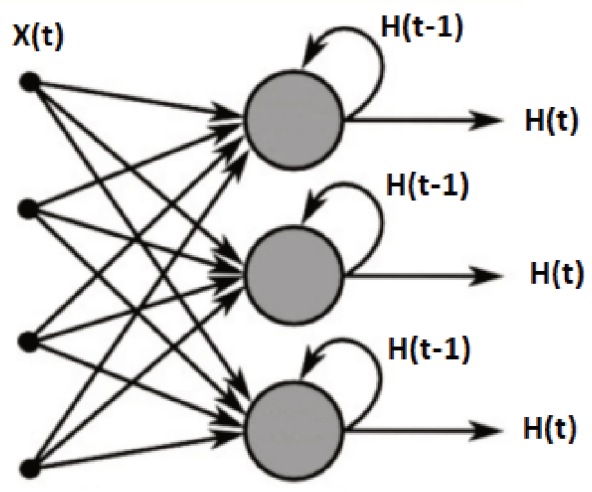
As redes neurais recorrentes, também conhecidas pela sigla RNN (do inglês *Recurrent Neural Networks*), são um tipo de rede neural artificial projetadas para reconhecer padrões em sequências de dados, como texto, genomas e até dados de séries numéricas (ACADEMY, 2019). São estruturas de processamento capazes de representar uma grande variedade de comportamentos dinâmicos, ou seja, possuem uma dimensão temporal, além da espacial.

As RNN têm uma entrada $X(t)$ (tempo presente) e um estado atual $H(t)$, que depende do estado anterior $H(t-1)$. Esse comportamento é ilustrado na Figura 7.

Considerando problemas de NLP, as RNN convencionais, em sua forma mais geral e básica, possuem capacidade para capturar a dependência entre as palavras apenas em uma direção da sentença. Além disso, as mesmas não têm um mecanismo de memória para manter dependências de longo prazo.

Adiciona-se ainda que essa arquitetura pode apresentar problemas com o cálculo do gradiente durante o aprendizado da rede. Porém, seu funcionamento serviu de base para que estruturas mais complexas surgissem.

Figura 7 – Representação gráfica de uma Rede Neural Recorrente, na qual $X(t)$ é a entrada atual, $H(t-1)$ é o estado do tempo anterior e $H(t)$ o estado atual.



Fonte – (ACADEMY, 2019)

3.1.1.1.2 Long Short-Term Memory (LSTM)

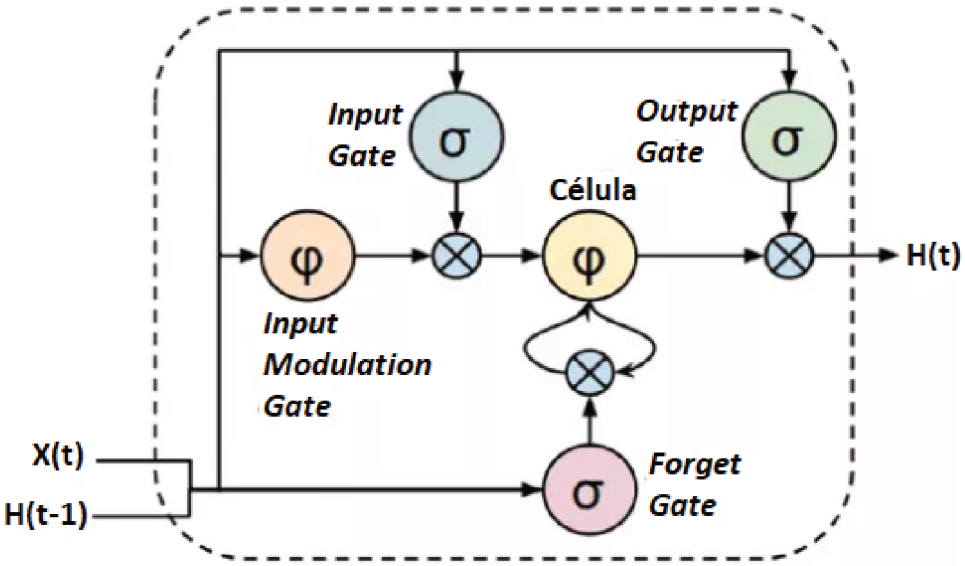
As redes *Long Short-Term Memory* são um tipo de RNN que resolvem o problema de memória de dependências de longo prazo. Elas foram introduzidas por Hochreiter e Schmidhuber em 1997 (HOCHREITER; SCHMIDHUBER, 1997) e aprimoradas em trabalhos posteriores.

As LSTM são compostas por uma estrutura em cadeia e possuem blocos chamados de **célula** de memória. O **estado** da célula consiste na memória de longo prazo.

A informação é retida pela célula e as manipulações de memória são feitas pelos portões, que serão referenciados aqui como ***gates*** (Figura 8). O papel deles é selecionar qual informação é relevante ou não para dada tarefa, ou seja, remover/esquecer ou adicionar/lembrar. Existem três tipos de *gates* (ACADEMY, 2019):

- *forget gate*: recebe a entrada $X(t)$ e o estado anterior $H(t-1)$, passa pela função de ativação σ e célula ϕ, tendo um resultado binário. Isso faz com que remova as informações que não são mais úteis no estado da célula;
- *input gate*: adiciona as informações úteis ao estado da célula através de uma regulação da entrada pela função σ, assim como no *forget gate*, que tem seu resultado multiplicado pela saída do *input modulation gate*;
- *output gate*: tem como papel extrair as informações úteis do estado da célula atual para apresentar na saída, possuindo um processo semelhante ao do *input gate*.

Figura 8 – Estrutura básica de uma célula de memória do tipo LSTM, na qual $X(t)$ é a entrada atual, $H(t-1)$ é estado no tempo anterior e $H(t)$ o estado atual.



Fonte – (ACADEMY, 2019)

Tal arquitetura das LSTM faz também com que o problema no cálculo dos gradientes seja amenizado, uma vez que possui uma estrutura única de gradiente aditivo em oposição às longas multiplicações de pequenos ou grandes valores nas RNN tradicionais (ARBEL, 2018). Além disso, as ativações do *forget gate* são diretamente acessadas na célula e com a constante atualização pelos *gates* tem-se um processo de aprendizado mais eficiente.

Uma aplicação muito comum das LSTM em NLP é o reconhecimento de entidades. Nesse tipo de problema, busca-se identificar se uma ou mais palavras em conjunto correspondem, por exemplo, ao nome de uma cidade. No artigo "*Entity recognition from clinical texts via recurrent neural network*" (LIU *et al.*, 2017) é apresentada uma abordagem em três camadas para extração de entidades em textos clínicos:

- camada de entrada: gera uma representação para cada palavra de uma sentença;
- camada LSTM: gera outra sequência de representação de palavras que captura as informações de contexto de cada palavra nesta frase;
- camada de inferência: produz uma sequência de *labels* que correspondem às entidades.

No caso deste artigo, foi utilizada uma LSTM bidirecional. Seu funcionamento consiste em captar representações da esquerda para a direita e vice-versa, o que

significa que as relações entre termos distantes entre si são assimiladas de forma mais eficaz. Os resultados obtidos no artigo para o reconhecimento de entidades foram satisfatórios, uma vez que superaram outros métodos tradicionais de aprendizado de máquina para esta tarefa (como os algoritmos *Support Vector Machine* e *Conditional Random Fields*).

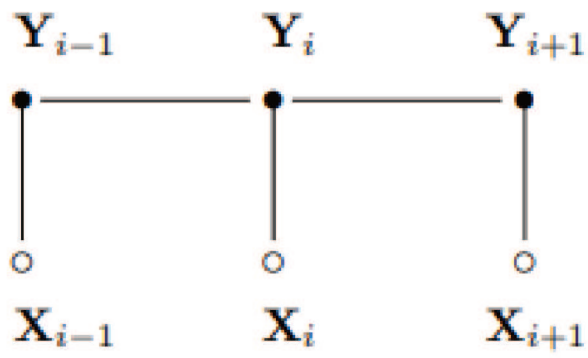
Apesar disso, duas sugestões de aperfeiçoamento foram citadas, que estão de acordo com o objetivo do presente projeto. A primeira seria em relação a integração de conhecimentos já existentes de informações clínicas para melhorar a performance do algoritmo. A segunda diz respeito ao reconhecimento de entidades com formatos e padrões específicos. No caso desse projeto, uma entidade corresponde a um código SNOMED-CT e pode ser composta por uma ou mais palavras, como foi mostrado na Figura 2 do Capítulo 2, na qual são apresentadas duas: ritmo sinusal (com SNOMED-CT sendo "426783006") e eletrocardiograma normal ("164854000").

3.1.1.2 *Conditional Random Fields* (CRF)

As CRF são uma classe de modelos probabilísticos baseados em uma abordagem condicional. No campo de NLP, são bastante utilizadas em tarefas que envolvem informações contextuais, classificação e para etiquetar e segmentar dados em sequência.

No caso de extração de entidades, por exemplo, a ideia principal consiste em calcular a probabilidade de uma certa sequência de *labels* dada uma sequência de palavras. Dessa forma, uma predição pode ser modelada como um grafo que implementa as dependências (representadas pelas ligações entre os nós do grafo) entre as predições.

Figura 9 – Representação gráfica de uma CRF *linear chain*, na qual *X* é a entrada do modelo e *Y* o resultado produzido por ele sobre as dependências entre *X*.



Fonte – (JOHN LAFFERTY; PEREIRA, 2001)

O tipo de grafo utilizado depende da aplicação, mas em processamento de linguagem natural as CRF do tipo *linear chain* são a modelagem mais comum. Na Figura 9 estão ilustradas as relações de dependências nessa configuração, que mostram que as entradas *X* anterior e posterior à atual estão interligadas.

Uma das formas clássicas de implementação das CRF em NLP é utilizando uma série de características das palavras para representá-las, denominadas *features*. Acentuação, pontuação e classe gramatical são alguns exemplos de *features* mais comuns nesse tipo de tarefa.

Apesar disso, escolher quais tipos de *features* são mais adequadas para uma dada atividade, não é algo trivial. Dessa forma, surgiram outros tipos de representação das palavras para aplicar nas CRF e que tornaram-se o estado da arte.

Uma primeira combinação de arquiteturas resulta na Bi-LSTM-CRF, que consiste em uma rede neural composta por camadas de LSTM e CRF. Nesse formato, o modelo utiliza de forma eficiente as informações do tempo passado e futuro através da configuração bidirecional da LSTM e extrai, a nível de sentença, as *labels* devido à CRF (ZHIHENG HUANG; YU, 2015).

Outra abordagem que está cada vez mais popular é a utilização de modelos pré-treinados em conjunto com as CRF. Essa prática é muito útil nos casos em que não se tem uma grande quantidade de dados com *labels* para realizar o treinamento supervisionado.

Recentemente foi publicado um trabalho que aplica o modelo BERT (*Bidirectional Encoder Representations from Transformers*) com CRF para o reconhecimento de entidades em português, porém sua implementação ainda não está disponível (FÁBIO SOUZA; LOTUFO, 2020). O BERT consiste em uma técnica de pré-treinamento para NLP baseado na arquitetura *Transformers*. Seus resultados são tão satisfatórios que após a sua primeira publicação, uma diversidade de variantes foram criadas.

O papel do BERT é representar as palavras através de vetores fazendo uso de um conhecimento prévio em determinada língua. Pode-se entender isto como utilizar uma máquina que sabe falar português e depois especializá-la com o vocabulário técnico (como o médico). A etapa de especialização é justamente realizada através das CRF, que fazem a extração das entidades finais.

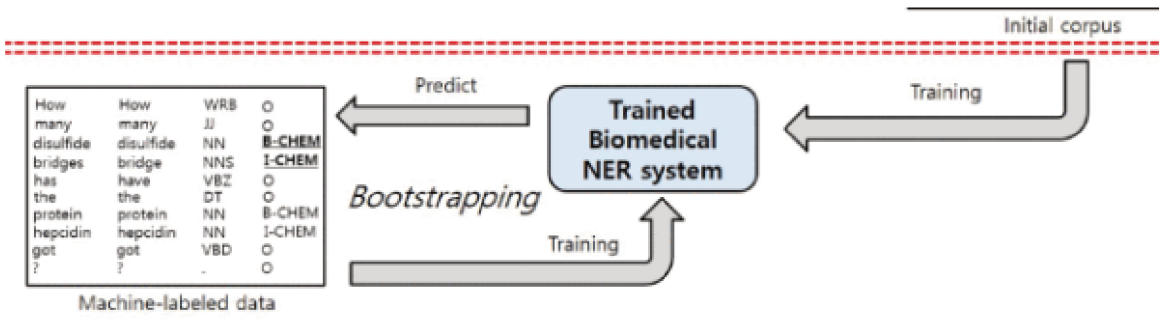
3.1.1.3 Técnica de *bootstrapping*

A literatura também apresenta as CRF sendo aplicadas para auxiliar na anotação de *labels* de forma automática através da técnica de *bootstrapping* (KIM *et al.*, 2019). Dado um pequeno conjunto de dados com anotações feitas por um especialista, é possível usar as CRF para colocar automaticamente *labels* e, através da técnica, aumentar e melhorar a qualidade do conjunto de treino para uma arquitetura a ser especializada em determinada tarefa. Vale reforçar que o *bootstrapping* pode ser aplicado

com outras arquiteturas além da CRF.

A Figura 10 ilustra este funcionamento. Com um conjunto inicial e pequeno de dados anotados, denominado ***Seed corpus***, é realizado o treinamento de uma CRF, indicada na ilustração pelo bloco *Trained Biomedical NER system*. A partir desse primeiro modelo, aplicam-se as *labels* em um conjunto maior e sem anotações (chamado ***Unlabeled corpus***, que torna-se o ***Machine-labeled corpus***). Unindo os dois grupos de dados, *Seed* e *Machine-labeled*, é treinado um novo modelo de CRF.

Figura 10 – Recorte da técnica de *bootstrapping* aplicada para tarefa de reconhecimento de entidades.



Fonte – (KIM *et al.*, 2019)

O conjunto *Unlabeled* é então reaplicado no novo modelo, comparado com o mesmo conjunto anotado no modelo anterior e modificado de acordo com algumas regras. Novamente, treina-se um novo modelo de *Seed* com o *Machine-labeled* atual. Este procedimento, que é denominado *bootstrapping*, é repetido até um número arbitrário de iterações. Sua performance é mensurada indiretamente através da tarefa especializada no qual o conjunto de dados final é aplicado. Esta é uma abordagem inovadora que traz grandes vantagens em projetos nos quais há um número reduzido de anotações e é muito difícil de se obtê-las, como é o caso do presente trabalho.

3.1.1.4 Treinamento de modelos

A seguir são discutidos alguns tópicos referentes ao treinamento de modelos de IA relevantes para o entendimento do trabalho. O primeiro ponto está relacionado ao algoritmo de otimização selecionado para a aprendizagem dos modelos, o que está intrinsecamente ligado à natureza da tarefa.

Para um problema de regressão, a função de custo pode ser escrita como na equação 4, na qual N representa o tamanho do conjunto de dados. Essa função indica quão bem ou mal está indo o modelo na busca pela solução no processo de otimização, uma vez que o termo $(t_i - y_i)$ modela o erro entre a resposta desejada e a predição.

$$J(w) = \frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2. \tag{4}$$

No caso das redes neurais, a aprendizagem das mesmas ocorre à medida que os pesos são modificados de forma iterativa. O ajuste dos pesos *w* pode ser descrito como na equação 5. A taxa de aprendizagem é designada por α , enquanto $J(w)$ representa a função de custo, geralmente referenciada por *loss*, a ser otimizada através de um método iterativo, como o do gradiente descendente (WIKIPEDIA, s.d.[a]), por exemplo.

$$w = w - \alpha \frac{\partial J(w)}{\partial w}. \tag{5}$$

Otimizadores que merecem destaque para redes neurais são RMSprop e Adam, mas existe uma diversidade de possibilidades. A ideia principal é a minimização da função de custo, sendo que cada algoritmo tem as suas particularidades.

Para o RMSProp (*Root Mean Square Propagation*), os gradientes são acumulados em uma janela fixa. Porém, a taxa de aprendizagem é ajustada automaticamente e possui valores diferentes para cada parâmetro. Já o Adam (*Adaptive Moment Estimation*) utiliza o conceito de *momentum* combinado com o RMSProp, que resulta na adição de frações dos gradientes anteriores ao atual (ALGORITHMIA, 2018). Enquanto o *momentum* acelera a busca em direção ao mínimo da função custo, RMSProp impede que essa procura vá para áreas de oscilações, que podem fazer a solução não convergir.

Um ponto muito discutido em IA é o ***overfitting***, que é quando o modelo performa muito bem em dados semelhantes ao de treino, mas em outros nunca vistos, muito mal. É basicamente como se a IA "memorizasse"os dados, sem ter aprendido de fato e sem conseguir generalizar. Uma solução é aumentar o conjunto de dados de treino, utilizando técnicas de augmentação (inserir ruído, inverter ordem das palavras, etc.).

Outra maneira de evitar o *overfitting* é utilizando algum método de **regularização**. É adicionada à função de *loss* um valor que penaliza grandes pesos. Isso significa que, além de ser penalizado por previsões incorretas, o modelo também será penalizado por ter grandes valores de peso, mesmo que suas previsões nos dados de treinamento estejam corretas (ALGORITHMIA, 2018).

Existem três tipos de regularização que valem ser citados: L1, L2 e *dropout*. Na regularização L1, os pesos diminuem em uma quantidade constante para 0, tendendo a concentrar o peso da rede em um número relativamente pequeno de conexões de alta importância. Já na L2, os pesos diminuem em um valor proporcional ao peso (ACADEMY, 2019).

A técnica de *dropout*, muito usada em redes neurais, ao contrário da Regularização L1 e L2, não depende da modificação da função de custo. Na verdade, modifica-se a rede em si.

No *dropout* tradicional, alguns neurônios da rede são temporariamente desativados de forma randômica a cada passo do algoritmo (ACADEMY, 2019). É como se

fossem treinados vários modelos durante um processo em paralelo. Para determinadas arquiteturas, existe ainda o *variational dropout* (GIORGI; BADER, 2019). A diferença é que a técnica se aplica a partes específicas da rede durante múltiplos passos.

Com relação à arquitetura CRF, têm-se outros algoritmos de otimização, apresentados no quadro 1. Nesse caso, podem utilizar a regularização L1 e/ou L2 ou algum método mais específico. Para o *Passive Aggressive*, por exemplo, existe um outro tipo de coeficiente que realiza esse papel.

Quadro 1 – Resumo dos principais algoritmos para a CRF.

Algoritmo	**Sigla**	**Características**
Limited-memory Broyden-Fletcher-Goldfarb-Shanno	L-BFGS	Maximiza o logaritmo da probabilidade dos dados de treino com regularização L1 e/ou L2. Melhora os pesos muito lentamente no início, mas converge rapidamente no final.
Stochastic Gradient Descent with L2 regularization	L2-SGD	Maximiza o logaritmo da probabilidade dos dados de treino com regularização L2. Aproxima dos pesos ótimos rapidamente, mas mostra convergência lenta no final.
Averaged Perceptron	AP	Calcula a média dos pesos em todas as atualizações do processo. É o algoritmo mais veloz no treinamento.
Passive Aggressive	PA	Utiliza o *loss* para atualizar o modelo.
Adaptive Regularization Of Weight Vector	AROW	Também utiliza o *loss*, porém o computa com uma formulação diferente.

Fonte – Elaborado pela autora com base em (OKAZAKI, 2007).

No presente projeto os diferentes otimizadores tiveram seu desempenho avaliado através de experimentos. Alguns dos parâmetros disponíveis foram variados para que seus efeitos no aprendizado do modelo fossem melhor compreendidos.

3.2 MÉTRICAS DE DESEMPENHO

Ao treinar um modelo de inteligência artificial é de suma importância avaliar o seu desempenho através de métricas. As mais clássicas, como acurácia, *precision*, *recall* e *F1-score*, são calculadas através da matriz de confusão (indicada na equação 6), que dispõe sobre o número de classificações de um modelo para cada categoria ou entidade e na qual TP são verdadeiros positivos, FP são falsos positivos, FN os falsos negativos e TN, verdadeiros negativos.

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}. \tag{6}$$

Para trabalhos que envolvem NLP, deve-se levar em conta ainda a forma de interpretação sobre o que é um acerto e um erro. Para reconhecimento de entidades, por exemplo, pode-se considerar uma predição correta como a extração da entidade exata,

mas isso nem sempre irá corresponder à realidade, uma vez que podem existir diversas maneiras de se escrever uma determinada expressão.

A partir das componentes da matriz de confusão, é possível calcular as seguintes métricas:

- acurácia: corresponde à taxa de acertos e é dada por $\frac{TP+TN}{TP+TN+FP+FN}$;
- *recall*: também chamada de sensibilidade, representa a porcentagem de classificações corretas dos que eram de fato positivos e é dada por $\frac{TP}{TP+FN}$;
- *precision*: dos que foram classificados como positivos, qual a taxa de positivos corretos, sendo dada por $\frac{TP}{TP+FP}$;
- *F1-score*: indica a qualidade geral do modelo e é muito utilizada quando conjuntos de dados possuem classes desbalanceadas. É dada por: $2 * \frac{precision*recall}{precision+recall}$.

As métricas apresentadas acima podem ser computadas de duas formas no caso de atividades que envolvam a extração de entidades, por exemplo. Além de calculá-las individualmente, por entidade, pode-se usar a versão **macro**, que contabiliza as métricas de cada entidade com o mesmo peso, ou a versão **micro**, que leva em conta acertos e erros do modelo de forma geral. As métricas podem ser reescritas, considerando *n* entidades, como:

Quadro 2 – Cálculo de métricas micro e macro para *n* entidades em um conjunto de dados.

	Micro	**Macro**
Recall	$\frac{\sum_n TP_n}{\sum_n TP_n+\sum_n FN_n}$	$\frac{\sum_n Recall_n}{n}$
Precision	$\frac{\sum_n TP_n}{\sum_n TP_n+\sum_n FP_n}$	$\frac{\sum_n Precision_n}{n}$
F1-score	$2 * \frac{(MicroPrecision)(MicroRecall)}{(MicroPrecision+MicroRecall}$	$\frac{\sum_n F1-score_n}{n}$

Fonte – Elaborado pela autora com base em (SHMUELI, 2019).

No contexto presente, quando o problema envolve mais de uma *label* e possui conjuntos com a quantidade de exemplos desbalanceados, recomenda-se não utilizar a acurácia como métrica principal, mas sim o *F1-score* (HUILGOL, 2019). Isso se deve ao fato do *F1-score* já levar em conta *precision* e *recall* em seu cálculo, trazendo mais valor à análise.

Para melhor esclarecer a diferença entre os cálculos do Quadro 2, apresenta-se um exemplo. Supondo-se que é desejado identificar duas entidades clínicas em um conjunto de laudos médicos, representadas pelos códigos SNOMED-CT "A", "B", "C", é retornada a seguinte matriz de confusão:

Tabela 1 – Matriz de confusão para exemplo fictício de reconhecimento de entidades.

		Resposta Verdadeira		
		A	**B**	**C**
Predição	**A**	4	6	3
	B	1	2	0
	C	1	2	6

Fonte – Elaborado pela autora com base em (SHMUELI, 2019).

Para as métricas micro, como observam-se todas as entidades ao mesmo tempo, cada predição errada é um falso positivo ou negativo, dependendo do ponto de vista. Por exemplo, se uma amostra para a entidade "A"é classificada como "B", então tal amostra pode ser vista como um falso positivo para "B"e falso negativo para "A". Já no caso da macro, o cálculo é feito para cada entidade da mesma maneira que em um caso binário, seguindo a matriz de confusão e posteriormente se obtém a média. Dessa forma, para o problema fictício, os resultados são apresentados na Tabela 2.

Tabela 2 – Métricas para exemplo fictício de reconhecimento de entidades.

	Micro	**Macro**
Recall	$\frac{4+2+6}{12+6+3+1+0+1+2} = 48\%$	$\frac{0.667+0.20+0.667}{3} = 51.1\%$
Precision	$\frac{4+2+6}{12+6+3+1+0+1+2} = 48\%$	$\frac{0.308+0.667+0.667}{3} = 54.7\%$
F1-score	$2 * \frac{0.48*0.48}{0.48+0.48} = 48\%$	$\frac{0.421+0.308+0.667}{3} = 46.5\%$

Fonte – Elaborado pela autora com base em (SHMUELI, 2019).

Para o presente projeto, ambas as formas, micro e macro, serão contabilizadas. Como cada uma delas observa o modelo de IA sob um ponto de vista, é relevante compará-las, principalmente a métrica *F1-score*.

3.3 LAUDOS MÉDICOS

O principal objeto de estudo deste projeto são os laudos médicos. Tendo em vista que existe uma alta demanda por interoperabilidade entre os sistemas de tecnologia da informação em saúde, é de suma importância compreender os padrões existentes que auxiliam nesta questão.

3.3.1 SNOMED-CT

A SNOMED-CT (*Systematized Nomenclature of Medicine Clinical Terms*), é uma terminologia clínica internacional multilíngue, usada atualmente em mais de 50 países, sendo a língua oficial o inglês. Foi inicialmente desenvolvida pelo *College of American Pathologists* e pelo Serviço Nacional de Saúde do Reino Unido, sendo adquirida em 2007 pela *International Health Terminology Standards Development Organization* (IHTSDO), uma organização dinamarquesa sem fins lucrativos.

Nos sistemas de informação é usado para intercâmbio eletrônico de informação clinica e de saúde, sendo também o padrão desejável nas especificações de interoperabilidade. A linguagem é muito próxima da utilizada naturalmente por cada país, o que garante o uso de um código único apesar das especificidades.

A terminologia está organizada em conceitos, interligados entre si, permitindo refiná-los e detalhá-los em diversos níveis. Esta funcionalidade permite aumentar a riqueza e consequentemente a qualidade dos dados inseridos em um sistema.

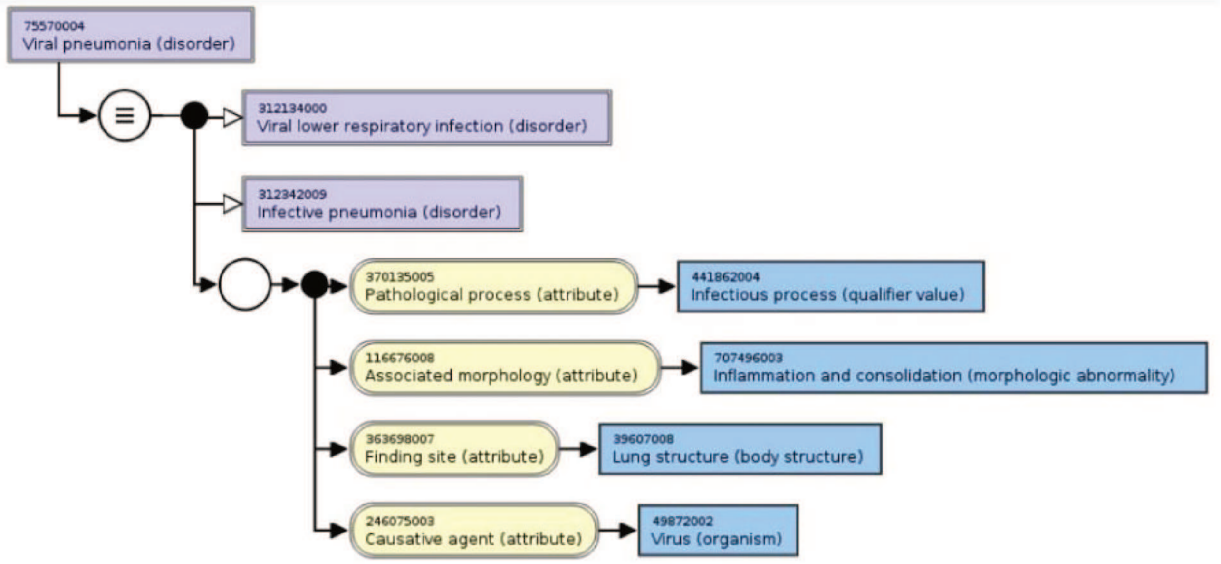
Para mostrar a potencialidade da SNOMED-CT é apresentado um exemplo. Uma **pneumonia viral** pode ser dita apenas como uma doença respiratória pela Classificação Internacional de Doenças (CID). Já na SNOMED-CT existem várias dimensões de significados: pneumonia infecciosa, doença respiratória, encontra-se no pulmão, causada por vírus.

Na Figura 11 é mostrado o grafo que relaciona tais informações, reforçando o que é doença (*disorder*), atributos (*attributes*), qualificadores (*qualifier value*), anormalidade morfológica (*morphologic abnormality*), parte do corpo (*body structure*) e até organismo (*organism*). Esses são só alguns exemplo de conceitos, mas existem muitos outros, como os achados médicos (*findings*) e procedimentos (*procedures*).

Apesar de muito completa a terminologia, infelizmente ainda não existe uma versão em português. A opção utilizada pela empresa Portal Telemedicina é a internacional, que está em inglês, o que faz com que seja sempre necessário aplicar uma etapa de validação com especialistas entre os termos nos diferentes idiomas.

Vale complementar que no sistema central de dados da Portal Telemedicina algumas categorias disponíveis na SNOMED-CT estão modeladas com nomenclaturas diferentes. Doenças são referenciadas como *conditions*, *morphologic abnormalities* são considerados equivalentes aos *findings*, *physical objects* são *products*, entre outros.

Figura 11 – Exemplo de grafo que relaciona conceitos sobre pneumonia viral da SNOMED-CT.

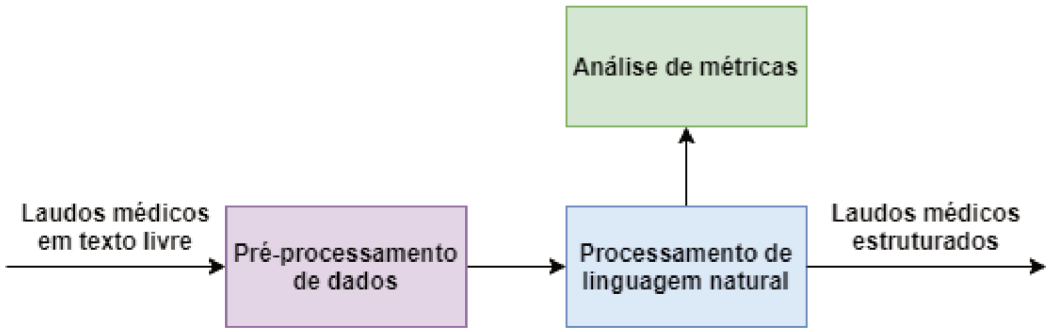


Fonte – (SNOMED-CT, s.d.)

4 MÉTODO

Para a realização deste projeto, que visa à estruturação automática de laudos médicos para extração de informações clínicas, foram necessárias diversas etapas de trabalho, apresentadas nas seções a seguir. De acordo com as experiências com IA dentro da empresa, tinha-se uma ideia de como seriam os principais componentes do estruturador: um bloco com pré-processamento dos dados, outro com o(s) modelo(s) de NLP e uma etapa de análise das métricas dos mesmos. O diagrama esquemático abaixo mostra, de forma simplificada, uma visão macro desse sistema, que recebe como entrada um laudo em texto livre e retorna o mesmo estruturado.

Figura 12 – Diagrama esquemático simplificado do estruturador automático.



Fonte – Elaborada pela autora (2020).

Outro conhecimento que auxiliou bastante na definição dos passos de trabalho, foi com relação à prévia experiência com o microsserviço de estruturação que utiliza expressões regulares. Esse conhecimento tornou a etapa de coleta e análise dos dados mais assertiva, pois já se sabia, de forma geral, que passos seriam necessários. Dessa forma, foi possível partir rapidamente para um estudo da modalidade médica escolhida (raio-X), seguida pela busca dos códigos SNOMED-CT correspondentes e pela validação das informações selecionadas com o médico especialista.

O maior desafio foi na definição dos detalhes de implementação, principalmente do bloco que envolveu o NLP propriamente dito, assim como as métricas disponíveis. Para isso, foi feita uma vasta pesquisa inicial por artigos técnicos que servissem de base ao que se desejava realizar, assim como literaturas mais práticas, que possuíssem códigos e casos de uso.

Uma primeira escolha, para facilitar a busca pelas referências, foi com relação ao tipo de problema a ser resolvido. No caso deste projeto, a modelagem que refletia melhor a aplicação foi a extração ou reconhecimento de entidades(no inglês, *entity extraction* ou *entity recognition*), uma vez que se deseja extrair códigos SNOMED-CT dos textos de laudo.

O termo NLP na literatura é muito geral e engloba muitos tipos diferentes de formatações de problemas e aplicações. Por exemplo, desde a tradução de textos, extração de entidades e de detecção de sentimentos. Devido a isso, definiu-se em conjunto dos orientadores a busca seletiva de artigos baseada nas seguintes premissas:

- artigos publicados no período de 2018 a 2020: A área de NLP é muito nova e muda rapidamente, assim como os algoritmos mais relevantes para tratamento de dados são mais recentes e por isso definiu-se o período dos últimos dois anos;
- busca pelo trabalho que teve mais sucesso na extração de entidades em contexto geral: Considerou-se importante analisar quais técnicas poderiam ser as mais adequadas, independentemente do caso de uso, uma vez que o problema a ser resolvido neste projeto é bastante específico e ainda não havia sido investigado sob outros pontos de vista;
- busca pelo trabalho que teve mais sucesso na extração de entidades no contexto de saúde: A ideia foi pesquisar por trabalhos na área de saúde para verificar pontos relevantes que poderiam surgir sobre a parte médica e que talvez necessitariam de maior atenção, mesmo que os algoritmos aplicados não fossem os mais avançados ou eficazes;

No Quadro 3 são apresentadas as palavras-chave usadas na procura dos artigos através do site *Google Scholar*. Pesquisou-se em inglês, uma vez que estes tipos de trabalho (mesmo de fontes brasileiras) são usualmente publicados no idioma. Já que também era desejado automatizar, em um primeiro momento, a etapa da seleção das expressões mais importantes dos laudos para posterior busca pelos códigos SNOMED-CT, foi incluída na pesquisa as palavras *unsupervised* (para algoritmos não-supervisionados) e *unlabeled* (indicando problemas sem *labels* disponíveis previamente).

Quadro 3 – Resumo das palavras-chave pesquisadas e quantidade de artigos encontrados.

Palavras-chave	**Artigos**
nlp unsupervised entity extraction recognition portuguese	2340
nlp unsupervised unlabeled entity extraction recognition portuguese	740
nlp unsupervised unlabeled clinical entity extraction recognition portuguese	144

Fonte – Elaborado pela autora (2020).

Dos 144 resultados, foram escolhidos 12 para o contexto específico em saúde e 6 para o geral. Fazendo uma leitura do resumo, introdução e conclusão dos mesmos, selecionaram-se os artigos abaixo, que já foram comentados no capítulo 3:

- "*Portuguese Named Entity Recognition using BERT-CRF*"(FÁBIO SOUZA; LOTUFO, 2020), uma vez que apresenta uma aplicação em português, com uma abordagem moderna;
- "*A Bootstrapping Approach With CRF and Deep Learning Models for Improving the Biomedical Named Entity Recognition in Multi-Domains*"(KIM *et al.*, 2019), já que possui uma estrutura reprodutível e utilizável no contexto do projeto.

Com as leituras iniciais e esses dois artigos selecionados, foi possível delimitar melhor o trabalho a ser realizado. Para o anteprojeto, foram definidas etapas mais superficiais, já que ainda não se tinha um conhecimento completo das informações aqui apresentadas. Nas seções a seguir são detalhadas as decisões tomadas frente ao que foi estipulado inicialmente e que guiaram a execução efetiva.

4.1 COLETA, ANÁLISE E PRÉ-PROCESSAMENTO DE DADOS

A etapa de preparação dos dados é uma das mais importantes neste projeto. Obtendo uma melhor compreensão das características dos textos de laudo torna-se possível identificar quais arquiteturas são mais adequadas para resolver o problema. As tarefas associadas a essa fase foram:

- pesquisa por bibliografias, atlas médicos, vídeos para adquirir maior familiaridade com o vocabulário radiológico;
- extração dos laudos médicos do sistema da empresa para análise de alto nível (verificação da frequência das palavras nos laudos, expressões mais relevantes), utilizando o *Google Colab* (serviço de nuvem gratuito voltado a esse tipo de tarefa);
- busca por códigos SNOMED-CT para as expressões mais interessantes extraídas da análise anterior e validação com um médico especialista, ou seja, verificação de quais informações clínicas seriam mais relevantes de serem identificadas pelo estruturador de laudos;
- processamento dos dados textuais, aplicando técnicas como tokenização, limpeza com remoção de pontuação, acentos, etc.;
- criação dos conjuntos de treino, validação e teste com *labels* para treinamento de algoritmos supervisionados.

Os passos citados foram escolhidos com base tanto na experiência com o microsserviço de estruturação já existente na empresa como nas leituras realizadas. Em vista disso, não houve dificuldade em defini-los.

Vale destacar que a criação de conjuntos com *labels* foi necessária, uma vez que verificou-se que não seria possível, dentro do escopo e tempo de projeto, aplicar técnicas de IA não-supervisionadas. A forma com que foram criados, porém, foi modificada durante o trabalho.

Primeiramente os dados seriam anotados por um especialista, já que ainda não havia expressões regulares implementadas para raio-X. Porém, ocorreu que as mesmas precisaram ser feitas para um projeto urgente da empresa e, em paralelo, o especialista não teve mais disponibilidade para fazer as anotações. Logo, a saída foi anotar os dados com as expressões criadas, adaptando-as quando necessário.

Outro ponto de atenção foi relacionado à divisão do conjunto de dados em treino, validação e teste. Visando a evitar o problema de *data leakage*[1], buscou-se ter muita cautela nesse procedimento. Esperava-se que seria possível fazer a divisão pelo médico que laudou o exame, já que podem existir diferenças entre a forma de escrita de cada um, porém, verificou-se que, devido ao grande desbalanceamento entre a quantidade de laudos por médico, optou-se por usar apenas uma amostragem randômica, que é explicada com maiores detalhes no capítulo 5.

4.2 IMPLEMENTAÇÃO DO NLP COM ANÁLISE DE MÉTRICAS DE DESEMPENHO

Com os dados formatados, parte-se para a implementação do processamento de linguagem natural. Tendo em vista as arquiteturas e técnicas apresentadas no capítulo 3, esta etapa consistiu em:

- testar ferramentas de NLP da *Google Cloud Platform*, uma vez que é a tecnologia de nuvem que a empresa utiliza;
- implementar as abordagens escolhidas;
- analisar os resultados obtidos com métricas de desempenho clássicas de inteligência artificial e aprendizado de máquina.

Foi feita uma prova de conceito com as ferramentas da GCP, comentada no primeiro Relatório Executivo da disciplina. Fazendo uma análise crítica, concluiu-se que o custo-benefício das mesmas não era compensado, já que necessitariam etapas adicionais de processamento e armazenamento para a tarefa desejada.

De acordo com todo o estudo realizado, elegeu-se que o artigo de KIM *et al.* (2019) seria o mais adequado para servir de base para o projeto, uma vez que, em um cenário em que um especialista anotaria os dados a quantidade de laudos com *labels* seria pequena, tornando interessante a aplicação da técnica de *bootstrapping*

[1] *Data leakage* é o uso de informações no processo de treinamento do modelo que não se espera que estejam disponíveis, fazendo com que as métricas sejam superestimadas.

proposta pelos autores para a criação de um conjunto de dados anotados maior automaticamente. Com isso, esse passo foi dividido na implementação do *bootstrapping* propriamente dito e de duas arquiteturas diferentes de NLP para a extração de entidades.

Para essas arquiteturas especializadas, foram escolhidos os modelos CRF baseada em *features* e Bi-LSTM-CRF. Ambas foram selecionadas não somente pelos artigos base do projeto, mas também pelas características já comentadas no capítulo 3. A diferença é que no presente trabalho foram testadas várias configurações possíveis, procurando entender a influência dos parâmetros e algoritmos, uma vez que nos artigos o contexto dos dados era bastante diferente.

Como os laudos foram anotados com expressões regulares automaticamente, decidiu-se fazer uma série de diferentes experimentos. Primeiro, selecionou-se uma quantidade pequena de laudos anotados (chamado *seed corpus*) e aplicou-os nas arquiteturas especializadas. De acordo com o desempenho desse teste, aplicou-se o conjunto total de treino anotado. Por fim, os parâmetros dos modelos desse último experimento foram utilizados para a técnica de *bootstrapping*, usando o conjunto de treino total separado em *seed corpus* e *unlabeled corpus*, com e sem *labels*, respectivamente.

As métricas aplicadas foram também definidas pelos artigos, sendo que foi observado que são, de fato, as mais utilizadas para NLP. São elas *precision*, *recall* e *F1-score*, com variações macro e micro. Seguindo as diretrizes da empresa, definiu-se como objetivo alcançar métricas acima de 80%. Algumas especificidades das bibliotecas de *Python* usadas permitiram ainda análises adicionais, não previstas anteriormente, mas que complementaram positivamente a discussão dos resultados, como é o caso da explicabilidade dos modelos.

Destaca-se ainda que, o conjunto de validação só foi utilizado para a Bi-LSTM-CRF para uma primeira avaliação dos parâmetros do modelo, pois não foi encontrada uma forma de incorporá-lo durante o treinamento da CRF baseada em *features*. Além disso, todas as métricas de desempenho apresentadas são referentes unicamente ao conjunto de teste.

Por fim, no anteprojeto foi falado na aplicação de *transfer learning* (em português, transferência de aprendizado), com alguma abordagem mais moderna de NLP. Verificou-se, porém, que a etapa seria mais adequada como sugestão de aperfeiçoamento, de acordo com o tempo disponível para a realização do projeto, o que foi indicado no Relatório Executivo 3.

4.3 ANÁLISE DE MÉTRICAS DE PESQUISA E DESENVOLVIMENTO

Por fim, de acordo com os objetivos estabelecidos na Introdução deste projeto, realizou-se a análise de métricas de pesquisa e desenvolvimento. São dois passos

fundamentais:

- comparar com o microsserviço de estruturação de laudos já existente (que utiliza expressões regulares);
- com base na escolha da abordagem final, dissertar sobre as possíveis métricas.

Já que não existe uma métrica equivalente as de IA para a estruturação feita com expressões regulares, a análise aplicada foi com relação à robustez da solução. Ou seja, é desejável identificar como o modelo baseado em IA performa diante de textos com ruídos (erros de digitação, gramaticais, ordem das palavras) e compará-lo com a atuação do microsserviço atual. Essa foi uma análise apenas qualitativa e que foi verificada através de diferentes exemplos de laudos médicos.

No caso das métricas de pesquisa e desenvolvimento, o estudo é um pouco mais subjetivo. Procurou-se avaliar o contraste do desenvolvimento e manutenção do microsserviço atual com a da solução proposta, refletindo também sobre os passos para a incorporação e escalabilidade da nova abordagem no contexto da empresa.

4.4 FERRAMENTAS

Durante a execução do projeto foram utilizados diversos recursos para o desenvolvimento. Para gerenciamento das atividades foi usado o *Jira* e *Trello*; para a busca dos códigos SNOMED-CT o seu browser público[2]; além das várias bibliotecas da linguagem de programação *Python*, que são citadas ao longo do relatório, destacam-se as ferramentas *Google Colab*, *Bitbucket* (para armazenamento dos códigos) e alguns componentes da GCP (*BigQuery*, *AutoML*, etc.). Também fez-se uso da linguagem de consulta estruturada ou SQL para extrair informações do banco de dados da empresa.

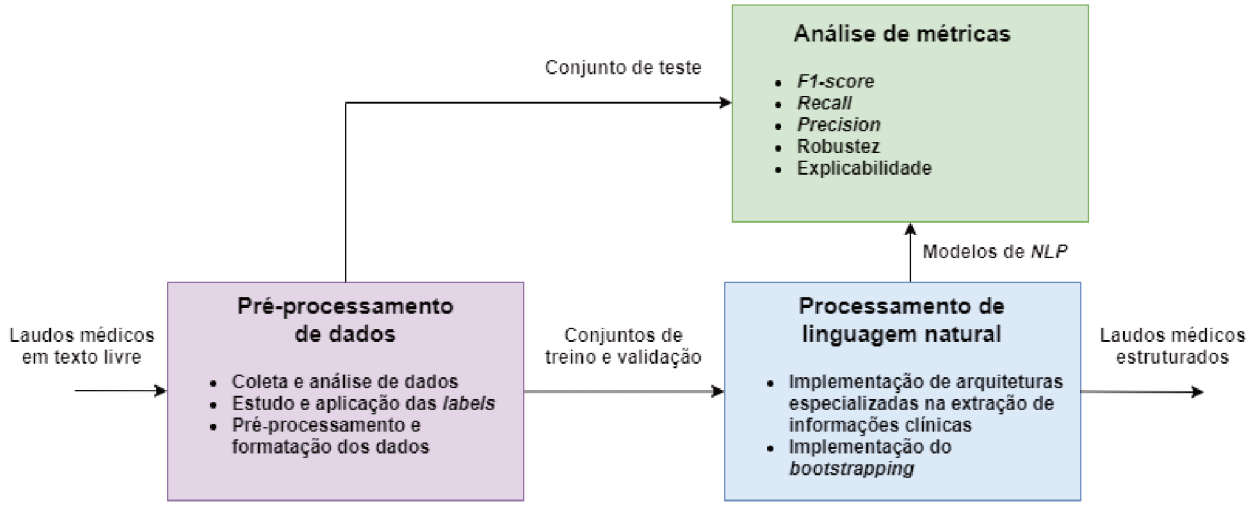
[2] https://browser.ihtsdotools.org/

5 IMPLEMENTAÇÃO

Neste capítulo são apresentados os detalhes de implementação da solução de acordo com o método proposto. A Figura 13 mostra o diagrama exposto no capítulo 4 de forma mais detalhada, identificando os passos intermediários e entradas e saídas para cada módulo. Nas seções seguintes têm-se as particularidades de todas essas etapas.

O repositório que armazena todos os testes realizados teve suas entradas e saídas devidamente versionadas para possíveis rastreamentos futuros. Também foi utilizada a ferramenta *Google Colab* como forma de apoio, facilitando a visualização dos resultados, e para que vários experimentos fossem rodados em paralelo.

Figura 13 – Diagrama esquemático da solução final do estruturador automático.

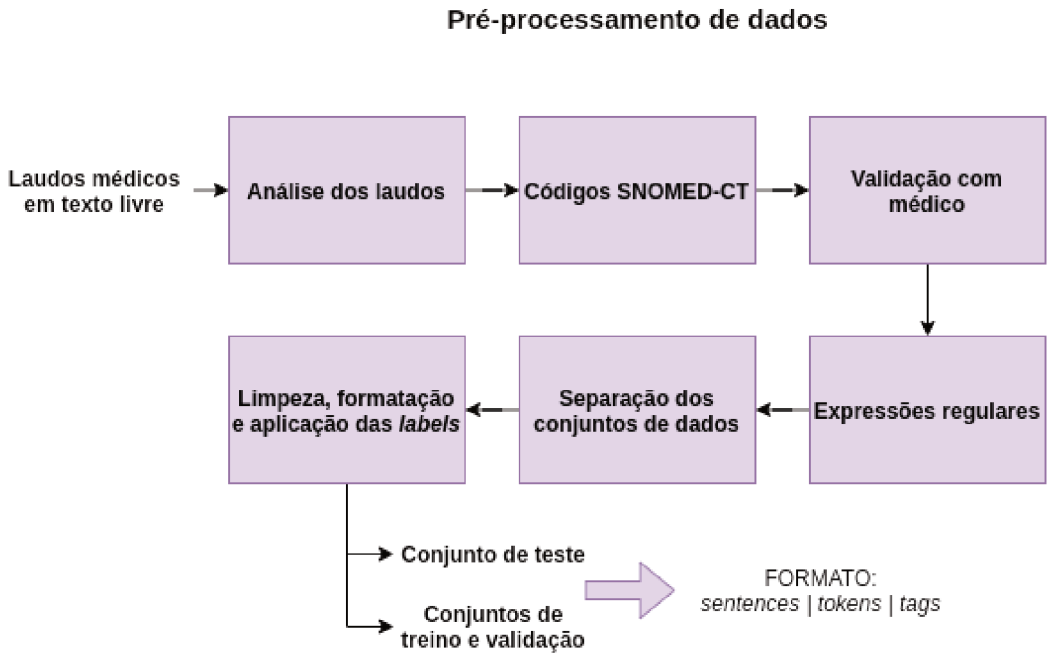


Fonte – Elaborada pela autora (2020).

5.1 PRÉ-PROCESSAMENTO DE DADOS

O diagrama da Figura 14 reúne os passos detalhados que são apresentados de 5.1.1 a 5.1.3 e que compõem o pré-processamento de forma geral. Esta parte do projeto foi bastante crucial, uma vez que pequenas mudanças poderiam afetar significativamente o resultado final.

Figura 14 – Diagrama detalhado das etapas do pré-processamento.



Fonte – Elaborada pela autora (2020).

5.1.1 Coleta e análise de dados

A criação do conjunto de dados mostrou-se parte fundamental do projeto, sendo tão importante quanto a tarefa de extração das informações clínicas dos laudos em si. Foi necessário focar na máxima automação destes processos, uma vez que até o presente momento tudo era realizado de forma bastante manual.

Em um primeiro momento, dedicou-se um período para o estudo do domínio de aplicação, que envolveu desde a leitura de artigos até assistir a alguns vídeos sobre a interpretação de radiografias. Em paralelo foi feita a coleta dos dados do banco da empresa gerando um arquivo em formato CSV para realizar análises mais detalhadas sobre os laudos específicos.

O arquivo, com aproximadamente 129 mil laudos de raio-X de tórax, especialidade que foi focada devido ao contexto da pandemia da COVID-19, foi carregado em um *data frame*[1]. Fez-se a separação com amostragem randômica simples, técnica que evita o problema de viés, em três conjuntos: 80% para treino, 10% validação e 10% teste. Após remoção de duplicados no conjunto de treino, restaram cerca de 30 mil laudos únicos, sendo que esses têm, em média, 44 palavras.

A Tabela 3 apresenta as divisões do conjunto de dados em números. O conjunto de treino para o processo de *bootstrapping* é separado ainda em duas partes: o *seed*

[1] Um *Data Frame* é uma estrutura bidimensional de dados, como uma planilha.

corpus (selecionado também por amostragem randômica simples), o qual recebe as *labels*, e o *unlabeled corpus*.

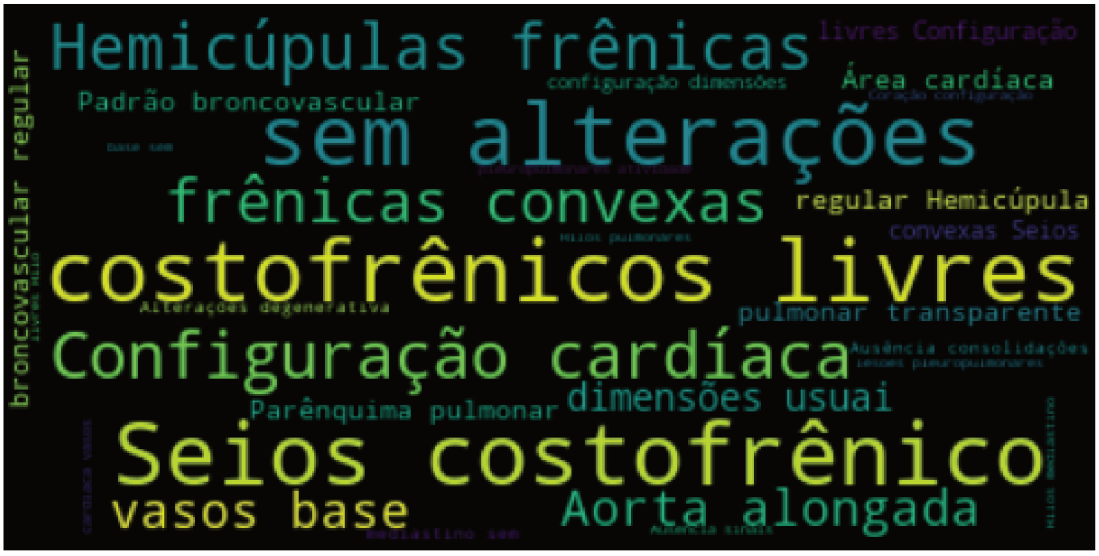
Tabela 3 – Distribuição do conjunto de dados separado em treino, validação e teste.

	Treino	**Validação**	**Teste**
Laudos	27520	12925	12926
Tokens	1226741	505532	502279

Fonte – Elaborada pela autora (2020).

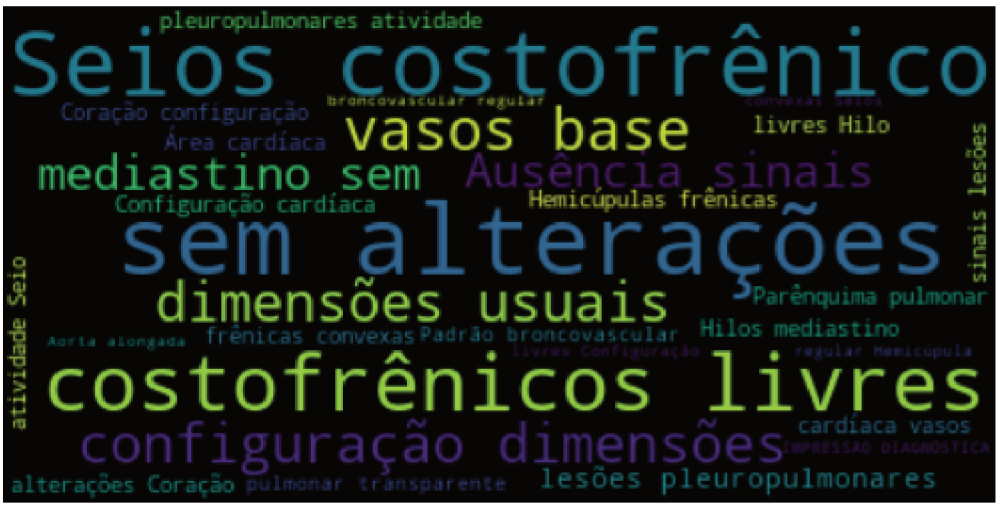
As Figuras 15, 16 e 17 apresentam as expressões mais relevantes relacionadas ao raio-X de tórax presentes nos laudos analisados. É possível perceber que as nuvens ficaram bastante semelhantes, o que aponta que os conjuntos de treino, validação e testes são, neste sentido, semelhantes, mesmo que com *labels* em quantidade desbalanceada. Outro destaque é que tais expressões estão mais relacionadas com partes do corpo e também que os achados clínicos mais comuns são os normais, como é o caso de "sem alterações".

Figura 15 – Nuvem com as expressões mais frequentes do conjunto de **treino** de laudos de raio-X de tórax.



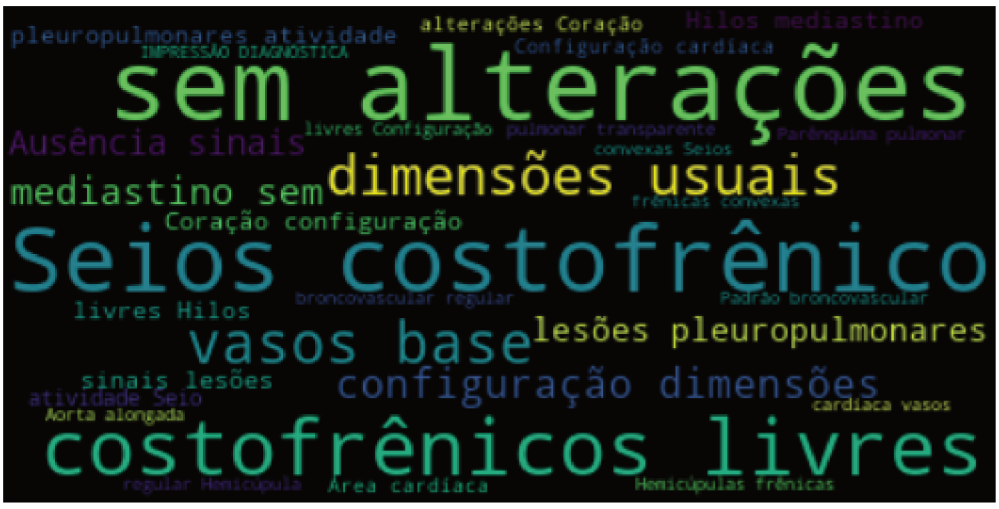
Fonte – Elaborada pela autora com as bibliotecas de *Python*: *Pandas*, *Pillow*, *Wordcloud* e *Matplotlib* (2020).

Figura 16 – Nuvem com as expressões mais frequentes do conjunto de **validação** de laudos de raio-X de tórax.



Fonte – Elaborada pela autora com as bibliotecas de *Python*: *Pandas*, *Pillow*, *Wordcloud* e *Matplotlib* (2020).

Figura 17 – Nuvem com as expressões mais frequentes do conjunto de **teste** de laudos de raio-X de tórax.



Fonte – Elaborada pela autora com as bibliotecas de *Python*: *Pandas*, *Pillow*, *Wordcloud* e *Matplotlib* (2020).

5.1.2 Estudo e obtenção das *labels*

A partir dos conhecimentos adquiridos na pesquisa sobre o vocabulário técnico de raio-X de tórax, foi possível buscar pelos códigos SNOMED-CT correspondentes às entidades-alvo a serem extraídas dos laudos. Este é um trabalho bastante manual, que deve ser feito a cada nova modalidade e/ou especialidade médica e que pode ser

resumido pelos passos a seguir:

- seleção das expressões relevantes através de uma leitura atenta dos laudos únicos;
- agrupamento das expressões semelhantes e possíveis variações na escrita;
- busca pela tradução para o inglês;
- busca pelos códigos SNOMED-CT correspondentes;
- organização de questionamentos a serem feitos a um médico especialista.

Com todas as dúvidas estruturadas, é realizada uma reunião com um médico especialista em formato de entrevista. De maneira geral, as perguntas feitas referem-se, principalmente, aos possíveis sinônimos para uma dada expressão, para que facilite a busca por uma tradução ou até a confirme. Também são questionados itens relativos à utilidade dos modelos de inteligência artificial que poderão ser produzidos. Por exemplo, compreender o que os médicos consideram quando classificam um exame como normal ou que está com má qualidade técnica.

A partir disso, segue-se para a etapa de anotação das *labels* em todos os conjuntos de dados (com exceção do *unlabeled corpus*, no caso do *bootstrapping*). Para isso, foi utilizado o microsserviço de estruturação de laudos já existente. Através de expressões regulares foi possível criar um conjunto para treinamento inicial que, apesar de não ser robusto, ou seja, não detectar expressões com erros gramaticais não previstos, consegue captar boa parte das informações ou pelo menos as mais relevantes de acordo com o especialista.

5.1.3 Pré-processamento e formatação dos dados

Esta etapa consistiu em preparar os dados para a aplicação nas arquiteturas escolhidas, já que foi verificado que cada uma exige um formato específico. Outra observação importante foi com relação à influência das características da língua portuguesa nos modelos de NLP, como a acentuação. A opção escolhida foi utilizar os laudos de forma mais crua o possível e aplicar mudanças apenas para tentar melhorar os resultados obtidos ou possíveis problemas.

A formatação comum para todos os algoritmos ficou definida como um *dataframe* com três colunas: frases, *tokens* e *tags*. A de frases serviu para identificar a qual laudo pertencem os *tokens* e suas *tags*. A de *tokens* apresenta o resultado da tokenização (separando palavras, pontuação e números), utilizando a biblioteca *NLTK* (*Natural Language Toolkit*) para *Python*.

Por fim, na coluna *tags* tem-se as *labels* aplicadas de acordo com o padrão IBO2 (*Inside, Outside, Beginning*). Tal formatação é muito utilizada em reconhecimento de

entidades e indica se um dado *token* é o início (indicado pela letra **B**), ou continuação de uma entidade, (letra **I**) ou se não possui uma correspondência (letra **O**). Na Figura 18, tem-se um exemplo para um laudo fictício de eletrocardiograma, o qual apresenta duas entidades: "ritmo sinusal"e "ECG dentro da normalidade".

Figura 18 – Exemplo de formatação das entidades seguindo o padrão IOB2 para a frase "laudo de ecg: ritmo sinusal parece mostrar ecg dentro da normalidade.". Duas entidades devem ser detectadas: a primeira composta por dois *tokens* e a segunda por quatro.

Tokens	*Tags*
laudo	O
de	O
ecg	O
ritmo	B-426783006
sinusal	I-426783006
parece	O
mostrar	O
ecg	B-16485400
dentro	I-16485400
da	I-16485400
normalidade	I-16485400

Fonte – Elaborada pela autora (2020).

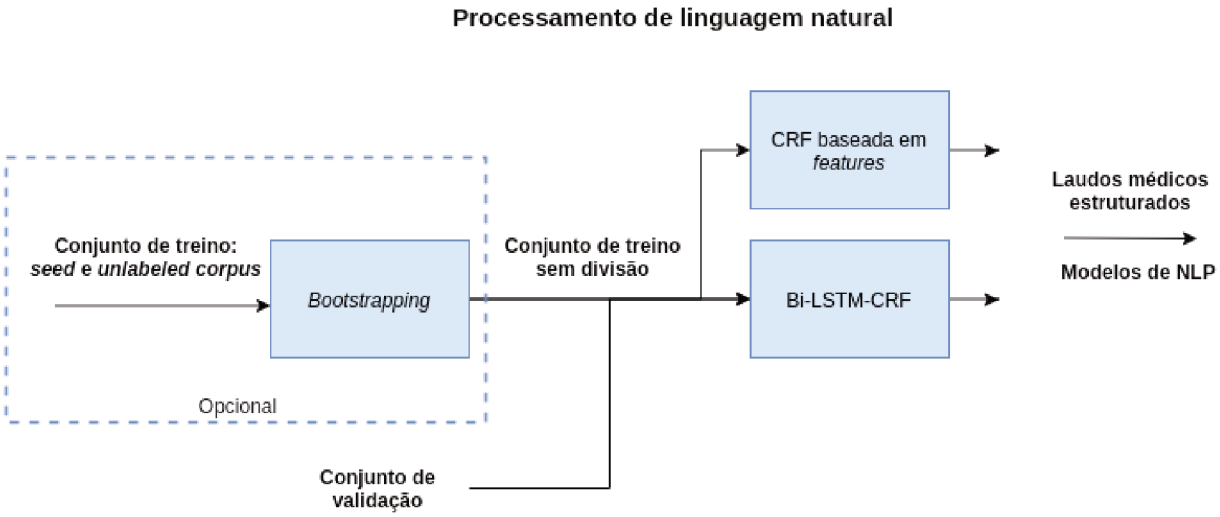
Após aplicar a tokenização identificou-se que alguns *tokens* estavam aglutinados erroneamente. Para amenizar este problema, uma vez que a maioria era por falta de espaço entre pontuação e palavras, criou-se uma função de limpeza dos dados. Essa substitui ponto, dois pontos e ponto e vírgula por espaço, remove parênteses e exclui espaços extras.

5.2 PROCESSAMENTO DE LINGUAGEM NATURAL

No diagrama da Figura 19 tem-se os passos determinados para o processamento de linguagem natural propriamente dito. A aplicação do *bootstrapping* é considerada opcional no caso em que a quantidade de dados anotados disponíveis é suficiente para treinamento, validação e teste.

Além disso, quando se fala em "conjunto de treino sem divisão" (Figura 19) refere-se a um conjunto inteiro que possua *labels*, que pode ser originado pelo *bootstrapping*, no qual se chama *machine-labeled corpus*, ou por outra seleção externa.

Figura 19 – Diagrama detalhado da etapa de processamento de linguagem natural.



Fonte – Elaborada pela autora (2020).

5.2.1 Implementação do *bootstrapping*

Um dos maiores desafios na área de IA refere-se à quantidade de dados disponíveis e representativos para realizar o treinamento de novos modelos. Como já apresentado no capítulo 3, o trabalho de KIM *et al.* (2019) traz uma possível abordagem para auxiliar neste problema.

No caso deste projeto, não foi possível criar um conjunto de dados anotados manualmente, uma vez que não havia um especialista disponível. Porém, através do microsserviço de estruturação de laudos atual, que utiliza expressões regulares, tornou-se viável adquirir *labels* criadas por máquina como conjunto inicial.

O Algoritmo 1 mostra, em forma de pseudocódigo, o funcionamento do *bootstrapping* de acordo com o artigo-base (KIM *et al.*, 2019). Este método consiste em utilizar um conjunto inicial de dados anotados pequeno (*seed corpus*), para treinar um classificador do tipo CRF e aplicá-lo em um conjunto maior de dados sem *labels* (*unlabeled corpus*).

O *unlabeled corpus*, após anotado pelo próprio classificador treinado na iteração atual, é denominado *machine-labeled corpus*. Esse, juntamente com o *seed corpus*, é usado para treinar um novo modelo de CRF após cada iteração do *bootstrapping*.

Os *machine-labeled corpus* da iteração atual e anterior são comparados durante o procedimento, a fim de que seja assegurado que uma *label* que o modelo aprendeu a identificar pelo *seed corpus*, que é considerado o conjunto padrão (e no caso de ter sido anotado manualmente por um especialista, seria o "gabarito") não seja modificada de forma incorreta. Essa lógica é específica do artigo-base, não exclusiva da técnica

Algoritmo 1: Técnica de *bootstrapping* baseada em KIM *et al.* (2019).

S: *Seed Corpus* (com *labels*),
U: *Unlabeled Corpus* (sem *labels*),
$Iter_{max}$: Número de iterações
Saída:
M: Lista com *Machine-labeled corpus* de cada iteração (M_i)
início
  classificador inicial $C_0 \leftarrow$ treinamento CRF(S)
  $M_0 \leftarrow$U
  $i \leftarrow 1$
  enquanto $i < Iter_{max}$ **faça**
    $M_i \leftarrow$ aplica CRF(*unlabeled*, C_{i-1})
    para $label_{i-1}$ *em* M_{i-1} *e* $label_i$ *em* M_i **faça**
      se $label_{i-1} \neq label_i$ **então**
        se $label_{i-1} \neq$ *'O'* **então**
          $label_i \leftarrow$label$_{i-1}$;
        fim
      fim
    fim
    se $i \neq Iter_{max}$ **então**
      $C_i \leftarrow$ treinamento CRF(S, M_i)
      $M_{i-1} \leftarrow$M$_i$
    fim
  fim
fim
retorna $M_1, ..., M_{Itermax}$

de *bootstrapping* em sua generalidade. Ao final, são retornados todos os *machine-labeled*, para que seja escolhido o mais representativo para o treino das arquiteturas especializadas.

Na Figura 20 tem-se um exemplo do funcionamento da técnica apresentada. Considerando o seguinte laudo médico ("RADIOGRAFIA: Paciente possui nódulo pulmonar grande e cardiomegalia presente."), que faz parte do *unlabeled corpus* e já foi separado em *tokens*, pode-se aplicá-lo ao modelo treinado apenas com o *seed corpus* como o primeiro passo do método e seu resultado é mostrado na segunda coluna.

Na terceira coluna apresenta-se o mesmo laudo aplicado ao modelo treinado com o *seed corpus* e o *machine-labeled corpus* resultante da primeira iteração. Ao passar pelo *bootstrapping*, verifica-se que na iteração anterior, "nódulo pulmonar" havia sido identificado como uma entidade diferente da atual, portanto deve-se mantê-la como antes. Já "cardiomegalia" não teve alteração.

Figura 20 – Exemplo simplificado de funcionamento da técnica de *bootstrapping*. Verifica-se que na iteração anterior (primeira coluna), "nódulo pulmonar" havia sido identificado como uma entidade diferente da atual, portanto deve-se mantê-la como antes. Já "cardiomegalia" não teve alteração.

Tokens	*Unlabeled* aplicado ao modelo treinado com *seed*	*Unlabeled* aplicado ao modelo treinado com *seed* e *machine-labeled 1*	*Unlabeled* aplicado ao modelo treinado com *seed* e *machine-labeled 1* após *bootstrapping*
RADIOGRAFIA	O	O	O
Paciente	O	O	O
possui	O	O	O
nódulo	B-86838002	B-255003	B-86838002
pulmonar	I-86838002	I-255003	I-86838002
grande	O	O	O
e	O	O	O
cardiomegalia	B-8186001	B-8186001	B-8186001
presente	O	O	O

Fonte – Elaborada pela autora (2020).

5.2.2 Implementação das arquiteturas especializadas na extração de informações clínicas

A partir dos conjuntos de dados pré-processados, partiu-se para a implementação das arquiteturas específicas para a tarefa a ser resolvida. São elas: CRF baseada em *features* e Bi-LSTM-CRF.

5.2.2.1 CRF baseada em *features*

A CRF implementada foi escrita com o auxílio da biblioteca *python-crfsuite*. O desempenho da arquitetura dependeu basicamente do pré-processamento específico, uma vez que o pacote provê funções simples para o treinamento do modelo. O código feito foi usado tanto para o ciclo do *bootstrapping* quanto dos testes isolados, modificando-se apenas os seus parâmetros.

Como foram mantidos os laudos com acentuação, maiúsculas, minúsculas e alguns tipos de pontuação, algumas *features* relacionadas foram escolhidas. No Quadro 4 abaixo, tem-se a lista completa. Acredita-se que é possível agregar mais funções futuramente, uma vez que a língua portuguesa possui várias características diferentes.

Na Figura 21 é apresentado um exemplo de dicionário de *features* para um *token* de uma frase ("Enfisema de partes moles difuso."). Cada chave do dicionário representa o nome da *feature* e de acordo com a análise aplicada, recebe um valor. Nesse caso, o *token* "Enfisema", por ser o primeiro recebe *BOS* (*Beginning of sentence*) como *True* (verdadeiro). Além disso, é carregada também a informação referente ao *token* vizinho "de", indicado pelo "+1". Assim sendo, cada exemplo para a CRF, ou seja, cada laudo médico, constitui-se de uma lista ordenada de dicionários de *features* e cada dicionário de *features* está associado a uma *label*.

Quadro 4 – Quadro de *features* para CRF.

Feature	**Significado**
bias	Indica a proporção de uma dada label no conjunto de treino.
word.lower()	Palavra em letras minúsculas.
word[-2:]	Dois últimos caracteres da palavra.
word.remove_accent()	Palavra sem acentos.
word.remove_accent_lower()	Palavra sem acentos e com letras minúsculas
word.isupper()	Palavra com todos os caracteres em maiúsculo.
word.istitle()	Palavra que começa com maiúsculo, iniciando a frase.
word.isdigit()	Token é um número ao invés de palavra.
word.ispunct()	Token é uma pontuação ao invés de palavra.
word.consonants()	Palavra com mais de duas consoantes consecutivas.
word.len()	Tamanho da palavra.
BOS	Indica que o *token* é o primeiro do laudo.
EOS	Indica que o *token* é o último do laudo.

Fonte – Elaborado pela autora (2020).

Figura 21 – Exemplo de dicionário de *features* do primeiro *token* da frase "Enfisema de partes moles difuso.".

```
{'+1:word.isdigit': False,
 '+1:word.ispunct': False,
 '+1:word.istitle()': False,
 '+1:word.isupper()': False,
 '+1:word.lower()': 'de',
 '+1:word.remove_accent': 'de',
 '+1:word.remove_accent_lower': 'de',
 'BOS': True,
 'bias': True,
 'word.consonants': False,
 'word.isdigit': False,
 'word.ispunct': False,
 'word.istitle()': True,
 'word.isupper()': False,
 'word.len()': 8,
 'word.lower()': 'enfisema',
 'word.remove_accent()': 'Enfisema',
 'word.remove_accent_lower()': 'enfisema',
 'word[-2:]': 'ma'}
```

Fonte – Elaborada pela autora (2020).

5.2.2.2 Bi-LSTM-CRF

A implementação da Bi-LSTM-CRF foi feita utilizando a biblioteca de código aberto *Keras*, escrita em *Python*. Além de uma linguagem de mais alto nível, é modular e bastante expansível, facilitando o trabalho de desenvolvimento.

A preparação dos dados para treino envolveu algumas análises adicionais em relação às seções 5.1.1 até 5.1.3. Um parâmetro muito importante é o máximo tamanho de um laudo (*max_length*), equivalendo à maior contagem de *tokens*. Para isso, utilizou-

se o conjunto de dados completo, mas pré-processado, resultando em um tamanho de 146 como máximo de *tokens* em um laudo.

Uma segunda verificação foi com relação ao número de *tokens* e entidades distintas, na qual também foi usado o conjunto total. Esse dado serve para criar um vocabulário de equivalências para o algoritmo, que é utilizado posteriormente para codificar cada *token* e entidade em números inteiros.

Para padronizar os vetores, que representam as frases, com o mesmo formato e tamanho, é aplicado o que se denomina *pad sequences*. Nesse caso, o preenchimento dos vetores foi feito à direita com zeros no caso das sequências de palavras e à direita com o número que representa a *tag* "O"para as *labels* até alcançar o *max_length* definido.

A rede neural implementada constitui-se de cinco camadas na seguinte ordem: entrada, *embedding*, LSTM bidirecional, densa e CRF. A primeira transforma as entradas da rede em tensores, que são entidades geométricas que generalizam a noção de escalares, vetores e matrizes. A camada de *embedding* modifica os números inteiros que representam os *tokens* para vetores densos (mais compactos para facilitar a computação) de tamanho fixo, que está relacionado ao número de *tokens* do vocabulário e o *max_length*, além de uma dimensão arbitrária.

A camada de LSTM bidirecional implementa a estrutura *encoder-decoder* falada no capítulo 3. No caso da Bi-LSTM optou-se por usar a *variational dropout*. Já a camada densa faz a ponte para a CRF do tipo *linear chain*. O código implementado é apresentado em 5.1.

Listing 5.1 – Modelo Bi-LSTM-CRF em linguagem *Python*.

```
def bilstm_crf_model(max_length: int, n_words: int, n_tags: int):

    input = Input(shape=(max_length,))
    model = Embedding(input_dim=n_words + 1,
                      output_dim=20,
                      input_length=max_length)(input)
    model = Bidirectional(LSTM(units=50,
                               return_sequences=True,
                               recurrent_dropout=0.1))(model)
    model = TimeDistributed(Dense(50, activation="relu"))(model)
    crf = CRF(n_tags)
    out = crf(model)

    model = Model(input, out)
    model.compile(optimizer="rmsprop",
                  loss=crf.loss_function,
                  metrics=[crf.accuracy])

    return model
```

5.3 HIPERPARÂMETROS DOS MODELOS

Para a arquitetura Bi-LSTM-CRF foi implementado o código já apresentado em 5.1. Os hiperparâmetros (parâmetros de definição global) do modelo são o tamanho máximo de um laudo e a quantidade de palavras e entidades. Vale destacar nesta etapa que a dimensão de saída (*output_dim*) da camada *Embedding*, assim como o número de neurônios (*units*) e a taxa de *dropout* (*recurrent_dropout*) foram definidos empiricamente.

Como função de ativação escolheu-se a ReLU e como otimizador o RMSprop. Acrescenta-se ainda que, como a camada final é a CRF, foram utilizadas as funções de *loss* e métricas específicas da mesma, que avaliam o desempenho da rede durante o treinamento.

No caso da CRF optou-se por testar os cinco algoritmos disponíveis, variando alguns parâmetros. No Quadro 5 os mesmos são apresentados. Vale destacar que existem outros que podem ser modificados, mas foram escolhidos os mais relevantes. Além disso, o *possible_transitions* foi setado como verdadeiro em todos os testes.

Quadro 5 – Quadro de hiperparâmetros de treino para CRF.

Parâmetro	**Significado**
max_iter	Número máximo de iterações.
possible_transitions	Inclui transições (sequências) possíveis entre os *tokens*, mesmo que não observadas.
c1	Coeficiente para regularização L1.
c2	Coeficiente para regularização L2.

Fonte – Elaborado pela autora (2020).

Com relação ao *bootstrapping*, foram realizadas três iterações. A CRF interna usada na técnica utiliza o algoritmo L-BFGS, com $c1 = 1$, $c2 = 1e^{-3}$, que são o padrão da biblioteca, *possible_transitions* = *True* (escolhido pela sua funcionalidade) e *max_iterations* sendo 100 no classificador inicial e 75 nas iterações seguintes. Esses dois últimos foram definidos através de experimentos iniciais, observando-se a curva de perda (*loss*).

5.4 ANÁLISE DE MÉTRICAS DE DESEMPENHO

Para as métricas, foram usadas algumas bibliotecas de *Python*. Para a Bi-LSTM-CRF o próprio módulo de treinamento da *Keras* já provê algumas informações durante o processo, sendo possível utilizá-las para visualização em gráficos. O módulo da CRF também tem essa possibilidade.

No caso da CRF a biblioteca usada ainda possuía funções para analisar a explicabilidade dos modelos a partir de dois pontos de vista. Um foi com relação ao estado

das *features* e outro para as transições entre elas. Assim, foi possível compreender melhor o que o algoritmo havia aprendido.

Outro pacote importante é o *Seqeval*, que possui funções para criar relatórios com as principais métricas de *machine learning*. Alguns cálculos adicionais foram implementados à parte, com funções simples da própria linguagem *Python* com o intuito de verificar o desempenho e apresentar em forma de gráficos.

Vale ressaltar ainda que considerou-se que um acerto corresponde ao modelo predizer a exata entidade para um dado *token* ou conjunto de *tokens* que a formem. Além disso, optou-se por utilizar tanto as métricas micro quanto macro, para que fosse possível compreender a performance a nível de entidades e e a nível do modelo de forma geral, já que existe um desbalanceamento na quantidade das mesmas.

6 RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados os testes realizados para a solução e a discussão dos mesmos. As subseções para os experimentos estão divididas de acordo com as seguintes nomenclaturas e seus objetivos:

- **aplicação do *seed corpus***: aplicar 600 laudos anotados pelo microsserviço com expressões regulares para testar diferentes configurações possíveis para cada arquitetura para se obter uma melhor compreensão do funcionamento das mesmas;
- **aplicação do conjunto de treino completo**: verificar se os resultados obtidos a partir das configurações testadas no *seed corpus* são reproduzidos em um conjunto maior, adaptando os parâmetros quando necessário. O conjunto completo consiste nos 27520 laudos anotados pelo microsserviço com expressões regulares referidos na Tabela 3 do capítulo 5;
- **aplicação do *bootstrapping***: utilizar os *machine-labeled corpus* resultantes de cada iteração da técnica para treinamento das melhores configurações de CRF e Bi-LSTM-CRF (de acordo com os experimentos das seções anteriores).

Além disso, destaca-se que os hiperparâmetros, como o número máximo de iterações, foram definidos também de maneira experimental. Em alguns casos foram observadas as curvas de *loss* para se ter uma ideia do ponto em que as métricas iriam saturar e a partir desse valor aproximado, escolher as variações apresentadas nas tabelas de cada teste. Ressalta-se que, futuramente, é possível automatizar essa etapa de definição dos hiperparâmetros.

6.1 EXPERIMENTOS

6.1.1 Aplicação do *seed corpus*

Este experimento consistiu em aplicar os 600 laudos (com *labels*) selecionados para o classificador inicial do *bootstrapping* diretamente nas arquiteturas propostas. Como a quantidade de dados é menor, utilizou-se esta etapa para testar as diferentes configurações possíveis para cada modelo, fornecendo insights para experimentos seguintes.

O conjunto total (treino, validação e teste), nesse caso, possui um vocabulário com 4081 *tokens* e 121 entidades. Vale destacar que essas informações (número de *tokens* e de entidades) são utilizadas apenas para parametrizar o modelo, ou seja, os conjuntos de validação e teste são isolados do conjunto de treinamento. Além disso, existem 23585 *tokens* com *labels*, sendo que o restante é a *tag* "O"(*Outside*), que não entra no cálculo das métricas.

Bi-LSTM-CRF

Com esta arquitetura, a proposta foi variar o número de épocas e o tamanho do *batch*. A Tabela 4 resume os testes realizados.

Tabela 4 – Testes e métricas no conjunto de teste para a Bi-LSTM-CRF treinada com o *seed corpus*.

Épocas	**Tamanho *batch***	**Micro *F1-score***	**Macro *F1-score***	***Recall***	**Micro *Precision***	**Macro *Precision***
5	**32**	**66%**	**59%**	**65%**	**68%**	**55%**
5	64	57%	49%	58%	56%	45%
5	100	53%	44%	53%	53%	40%
15	32	51%	4%	50%	51%	38%
15	64	11%	6%	8%	20%	5%
50	100	39%	28%	39%	39%	27%

Fonte – Elaborada pela autora (2020).

Foi feita ainda uma tentativa com *batch* de 16, porém como as métricas obtidas foram muito inferiores, ocultou-se esse teste. Como se pode notar, o melhor resultado foi com cinco épocas e tamanho de *batch* 32. Nesse caso, todas as métricas tiveram o melhor desempenho. Com tal experimento, demonstrou-se que um número menor (apesar de não ser o mínimo) para ambos os parâmetros traz uma performance, a princípio, superior. Já que aqui o tamanho do conjunto de treino é reduzido, será verificada essa hipótese nos experimentos seguintes.

Outro destaque é que, como o conjunto de treino não possuía tantos exemplos, as métricas mantiveram-se muito baixas. É desejável pela empresa que, sempre que possível, esses números sejam acima de 80%, sendo que, dependendo da tarefa, pode-se abrir mão de uma métrica em detrimento da melhoria de outra (escolher hiperparâmetros que resultem em maior *recall*, mesmo que diminua o *precision*, por exemplo), o que também não foi o caso. Apesar disso, pode-se afirmar que, apesar dos poucos exemplos, o modelo teve um desempenho superior ao que se esperaria e poderia, talvez, ser aperfeiçoado.

CRF

Para os experimentos do *seed corpus* na CRF, optou-se por testar os cinco algoritmos disponíveis na biblioteca *python-crfsuite*. Como cada tipo tem uma diversidade de parâmetros, escolheu-se variar o número de iterações em todos e em dois deles os coeficientes de regularização. Decidiu-se apresentar apenas as métricas *F1-score*, a fim de simplificar a análise, já que levam em conta *precision* e *recall* no seu cálculo.

Na Tabela 5 são apresentadas as variações para o algoritmo L-BFGS e L2-SGD. A melhor opção entre os dois está destacada na primeira coluna e foi com um número máximo de iterações de 60, coeficiente c1 menor e c2 *default*. É notável que com c1

maior, as métricas caem de forma drástica, o que é compreensível, uma vez que o número de exemplos é pequeno e a regularização L1 reduz os pesos tendendo a zero para evitar o *overfitting*. No caso do algoritmo L2-SGD, acredita-se que, como o conjunto de dados é pequeno, a natureza intrínseca do algoritmo não foi bem aproveitada, já que sua vantagem é lidar justamente com grandes processos de treino.

Tabela 5 – Testes e métricas no conjunto de teste para CRF com o algoritmo L-BFGS e L2-SGD treinadas no *seed corpus*.

CRF com algoritmo L-BFGS				**CRF com algoritmo L2-SGD**			
max_iter	**Parâmetros**	**Micro *F1-score***	**Macro *F1-score***	***max_iter***	**Parâmetros**	**Micro *F1-score***	**Macro *F1-score***
60	**c1=0.01 c2= 1e^{-3}**	**98.5%**	**98%**	60	c2=1e^{-5}	95.9%	94%
60	c1=1 c2=1e^{-5}	97.5%	96%	60	c2=1e^{-3}	95.5%	94%
60	c1=1 c2=1	90.7%	86%	60	c2=1	88%	83%
60	c1=5 c2=1e^{-3}	83.6%	78%	100	c2=1e^{-5}	96.2%	95%
100	c1=1 c2=1e^{-3}	94.2%	90%	100	c2=1e^{-3}	96.3%	95%

Fonte – Elaborada pela autora (2020).

Para os outros três algoritmos disponíveis são mostrados apenas os resultados das melhores configurações testadas (Tabela 6). As métricas para os métodos *Averaged Perceptron* (AP) e *Passive Aggressive* (PA) ficaram bem próximas, como pode ser visto na Tabela 6. O *recall* e *precision* associados, não apresentados na tabela, mas estudados, ficaram bastante semelhantes. Apesar dos algoritmos PA e AROW computarem a função de *loss* de forma bastante semelhante, suas métricas foram bem diferentes, mostrando que a parcela a mais que o PA traz nesse cálculo do *loss* provavelmente auxilia no desempenho do modelo.

Tabela 6 – Testes e métricas no conjunto de teste das CRF com os algoritmos PA, AP e AROW treinadas com o *seed corpus*.

	Micro *F1-score*	**Macro *F1-score***
AP (*max_iter*=500)	97.4%	97%
PA (*max_iter*=500)	98.2%	98%
AROW (*max_iter*=10)	84.7%	87%
AROW *(max_iter*=60)	83.2%	89%

Fonte – Elaborada pela autora (2020).

Por fim, observou-se que a configuração que alcançou os melhores resultados foi com o algoritmo L-BFGS, apesar de empatar com outros em métricas específicas. Além disso, AP e PA ficaram bem próximos em termos de *recall* e *precision*, com bom desempenho para *F1-score*, ficando também como outros dois testes a serem aplicados no conjunto de treino maior.

6.1.2 Aplicação do conjunto de treino completo

A partir dos insights obtidos na primeira bateria de testes, partiu-se para a aplicação do conjunto de treino completo com 27520 laudos com *labels*. O conjunto total aqui possui 5996 palavras e 125 entidades. Os experimentos consistiram basicamente em utilizar as melhores configurações da seção anterior.

Bi-LSTM-CRF

A Tabela 7 mostra as variações aplicadas à arquitetura. Nos testes com menos exemplos verificou-se que um número menor de épocas e de tamanho de *batch* performou melhor, então procurou-se analisar se o mesmo ocorreria aqui. Como o tamanho de *batch* não afetou as métricas no melhor caso, partiu-se para um estudo de desempenho por entidades.

Tabela 7 – Testes e métricas no conjunto de teste para Bi-LSTM-CRF treinada no conjunto de treino completo com *labels*.

Épocas	**Tamanho *batch***	***F1-score***
3	32	99.8%
3	64	99.8%
3	100	99.8%
5	32	99.7%
5	64	99.5%
5	100	98.5%

Fonte – Elaborada pela autora (2020).

Entre 32 e 64 de *batch* e três épocas, apenas uma entidade ("84089009", que representa "hérnia hiatal"e possuía 14 exemplos no conjunto de teste) obteve métricas diferentes, sendo que no segundo caso foi o melhor resultado. Apesar disso, algumas entidades obtiveram métricas individuais iguais a zero ou muito baixas, pois provavelmente possuíam poucos ou nenhum exemplo no conjunto de treino.

CRF

Para a CRF optou-se por testar três algoritmos com a configuração que teve o melhor desempenho no *seed corpus*. Todos tiveram métricas muito semelhantes e bastante elevadas, como pode ser visto na Tabela 8. O *F1-score* micro e macro foram

os mesmos. Vale destacar também o tempo de treinamento, já que neste teste chamou atenção: com menos horas foi possível treinar um modelo com métricas muito boas.

Tabela 8 – Comparação das métricas no conjunto de teste entre algoritmos da CRF treinadas no conjunto de treino completo.

Algoritmo	**Parâmetros**	***F1-score***	**Tempo treinamento (horas)**
L-BFGS	c1=0.01 c2=1e-3 max_iter=60	99.8%	1.34
AP	max_iter=500	99.9%	5.85
PA	max_iter=500	99.9%	5.24

Fonte – Elaborada pela autora (2020).

6.1.3 Aplicação do *bootstrapping*

Compreendidos os modelos e seus parâmetros em diferentes cenários, encaminhou-se para a aplicação da técnica de *bootstrapping*. Optou-se por fazer três iterações, uma vez que no artigo-base (KIM *et al.*, 2019) foi o suficiente.

Os experimentos finais consistiram em aplicar os *machine-labeled corpus* resultantes de cada iteração em três configurações de modelos. O conjunto total de dados era composto por um vocabulário com 6112 *tokens* e 121 *tags*.

Tabela 9 – Métricas do conjunto de teste para cada modelo treinado com o *machine-labeled corpus* resultante de cada iteração do *bootstrapping*.

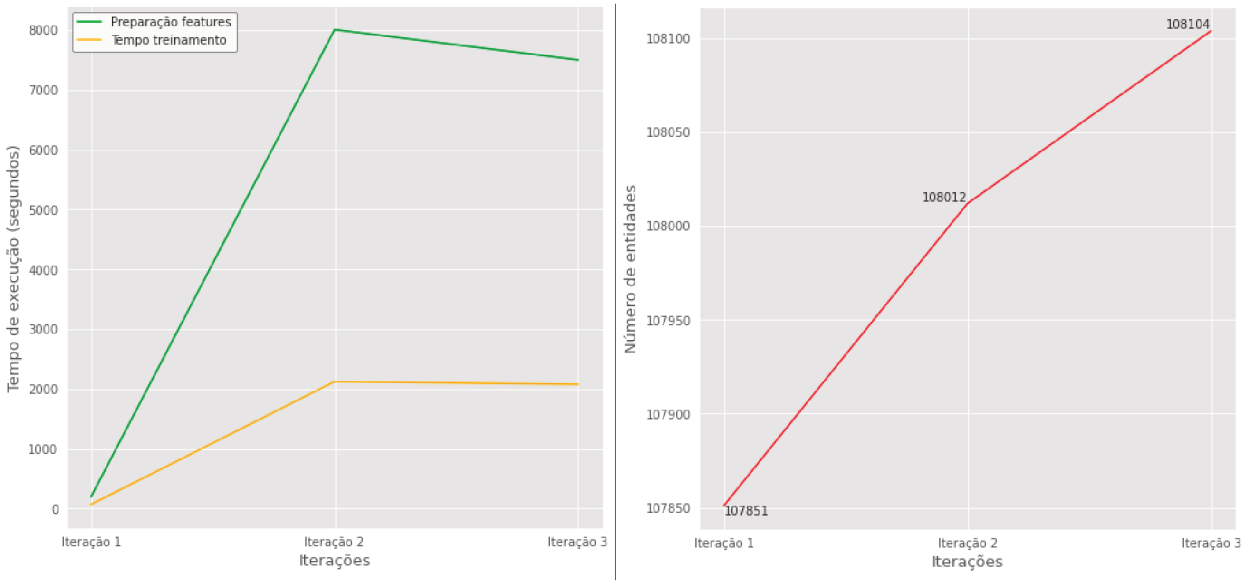
	Iteração 1			**Iteração 2**			**Iteração 3**		
	Bi-LSTM -CRF	**L-BFGS**	**PA**	**Bi-LSTM -CRF**	**L-BFGS**	**PA**	**Bi-LSTM -CRF**	**L-BFGS**	**PA**
Micro *F1-score*	89%	94%	91.1%	89.9%	94%	91.2%	88.8%	94%	91.1%
Macro *F1-score*	84%	90%	87%	85%	90%	87%	83%	90%	87%
Recall	84%	89%	86%	84%	89%	86%	83%	89%	86%
Micro *Precision*	95%	100%	97%	97%	100%	97%	95%	100%	96%
Macro *Precision*	91%	98%	96%	91%	98%	96%	92%	98%	96%

Fonte – Elaborada pela autora (2020).

A Bi-LSTM-CRF aplicada foi com três épocas e tamanho de *batch* 64. Na Tabela 9 pode-se verificar que o melhor desempenho com ela foi alcançado na segunda iteração. Para a CRF com o algoritmo *Passive Aggressive* a melhor performance geral também ocorreu na segunda iteração (pela métrica Micro *F1-score*), enquanto na CRF treinada com L-BFGS, o comportamento das métricas manteve-se igual entre as iterações.

Na Figura 22, à esquerda, tem-se o tempo de preparação das *features* para a CRF interna do *bootstrapping* e o tempo de treinamento em si. Na primeira iteração, como essas operações são aplicadas no *seed corpus*, que é muito menor, é natural que seja mais rápido.

Figura 22 – À esquerda, o tempo de execução da preparação das *features* e do treino da CRF. Já à direita, quantidade de entidades identificadas ao longo das iterações do *bootstrapping*.



Fonte – Elaborada pela autora (2020).

Acredita-se que esse tempo poderia ser otimizado, pois para todas essas etapas foi utilizada a biblioteca *Pandas* para salvar os dados gerados para consulta durante as análises do projeto. Atualmente existem outros pacotes disponíveis para *Python* que trabalham esse tipo de dado de maneira mais eficiente, mas por questão de praticidade optou-se por usar uma ferramenta que já se tinha domínio.

Ainda na Figura 22, à direita, é apresentada a evolução na detecção das entidades durante o processo de *bootstrapping* para o *machine-labeled corpus*, sem a *tag* "O". É notável a quantidade de entidades identificadas já na primeira iteração, na qual o *bootstrapping* em si não tem efeito, uma vez que é comparado o conjunto sem *labels* (*unlabeled*) com o que foi aplicado no modelo treinado apenas no *seed corpus*.

A partir da Figura 22, verificou-se que com apenas 600 exemplos já foi possível extrair uma grande quantidade de informações, o que demonstra o poder da técnica e da arquitetura interna escolhida (CRF) para esse processo. Ao longo das iterações subsequentes existe um ótimo crescimento nas detecções, o que era esperado, já que após a primeira iteração o classificador é treinado com o *seed corpus* e *machine-labeled corpus* juntos.

6.2 DISCUSSÃO

Na Figura 23 são apresentadas as entidades que mais foram modificadas durante o *bootstrapping*. A partir dela, pode-se concluir que as métricas gerais não foram tão modificadas nas arquiteturas finais, porque algumas das entidades que mais tiveram alterações são as que já possuíam muitos exemplos. Logo, quando o *machine-labeled corpus* era aplicado para treinamento nas arquiteturas finais, praticamente não alterava a aprendizagem dos modelos, principalmente no caso da CRF, que leva em conta as *features*.

Figura 23 – Entidades que mais sofreram mudanças durante as iterações do *bootstrapping*.

	iter1	iter2	iter3
B-36118008	115	116	121
B-443092002	143	147	150
I-443092002	143	147	150
I-128601007	370	372	373
B-128601007	370	372	373
I-419991009	707	707	708
B-419991009	707	707	708
B-8186001	846	847	848
B-249674001	964	968	970

	iter1	iter2	iter3
B-27925004	975	980	984
B-298382003	1015	1017	1020
B-162904008	1713	1716	1718
I-249674001	1709	1719	1724
I-162904008	3426	3432	3436
B-128308005	3711	3717	3722
B-409609008	5480	5484	5485
B-827032003	9331	9346	9354
I-827032003	21492	21542	21567

Fonte – Elaborada pela autora (2020).

Apesar disso, um resultado muito interessante foi extraído da Figura 23. A entidade "36118008", que representa "Pneumotórax", no *seed corpus* possuía apenas três exemplos, porém todos eram referentes ao *token* "PNM", que é a forma abreviada que os médicos utilizam. Mesmo com esses três exemplos possuindo *tokens* vizinhos diferentes, o modelo (CRF) conseguiu detectá-la facilmente no *unlabeled corpus*.

Um comportamento semelhante foi observado para as entidades "443092002" (que representa "Calo ósseo") e "128601007" ("Infecção pulmonar"). Ambas possuíam apenas quatro exemplos no *seed corpus*, porém no *unlabeled* foram detectadas. Isso indica que a CRF é uma arquitetura muito interessante para ser utilizada pela técnica de *bootstrapping* e que possivelmente com poucos exemplos, mas que sejam representativos, é viável obter um modelo resultante desse processo com boas métricas.

Verificando as métricas *precision*, *recall* e *F1-score* por entidades, aplicando-se o conjunto de teste no modelo CRF treinado com o algoritmo PA para o *machine-*

labeled corpus de cada iteração, foi possível perceber que, individualmente, a detecção de algumas delas melhorou. A entidade já referida para "Calo ósseo", por exemplo, subiu 0.03% entre a primeira e segunda iterações (e manteve-se na terceira). Porém, observou-se que muitas dessas métricas já estavam altas devido ao grande número de exemplos treinados, mesmo que tenham sofrido mudanças entre as iterações, como na Figura 23.

6.2.1 Análise de robustez dos modelos

Uma verificação importante, é em relação à robustez dos modelos. Os desempenhos podem ser contrastados também com o microsserviço de estruturação que usa as expressões regulares.

A robustez da Bi-LSTM-CRF foi verificada primeiramente no experimento com o conjunto de treino completo (de acordo com a seção 6.1.2) para o modelo treinado com três épocas e tamanho de *batch* 64. Na Figura 24 foi analisado um laudo do conjunto de teste. Na esquerda tem-se a frase original e na direita a mesma com pequenas mudanças (troca de letras e sem acentuação nos *tokens* sublinhados com um traço vermelho).

Figura 24 – Predições do modelo Bi-LSTM-CRF treinado com o conjunto completo para um laudo com e sem ruído.

Tokens	Predições	*Tokens*	Predições
Enfisema	B-87433001	Enfizema	O
de	O	de	O
partes	O	partes	O
moles	O	moles	O
difuso	O	difuso	O
.	O	.	O
Não	O	Não	O
há	O	há	O
sinais	O	sinais	O
de	O	de	O
pneumotórax	B-36118008	pneumotorax	B-36118008
.	O	.	O
Opacidade	B-128308005	Opacidade	B-128308005
lateral	O	lateral	O
do	O	do	O
hemitórax	O	hemitórax	O
esquerdo	O	esquerdo	O
.	O	.	O
Dreno	B-32712000	Dreno	B-32712000
na	O	na	O
base	O	base	O
do	O	do	O
hemitórax	O	hemitórax	O
direito	O	direito	O
.	O	.	O
Calcificações	B-129748009	Calcificaçoes	O
ateromatosas	B-48434008	ateromatosas	B-48434008
aórticas	O	aórticas	O
.	O	.	O

Fonte – Elaborada pela autora (2020).

Como foi utilizada uma vetorização das frases muito simples, ou seja, que não carrega nenhuma informação contextualizada, para a Bi-LSTM-CRF, apesar das métricas altas (todas acima de 99%), a robustez ficou prejudicada (qualitativamente observável na Figura 24). Isso significa que o modelo teve um comportamento semelhante ao das expressões regulares, o qual erra a predição diante de sutis mudanças (ruídos) não previstas. Para isso, a solução é fazer uso de vetores que representem melhor as relações entre os *tokens*. Como as métricas para essa arquitetura foram inferiores no *bootstrapping*, optou-se por não apresentar resultados de robustez para ela.

Para as CRF, com erros gramaticais ou de digitação, o comportamento é semelhante ao da Bi-LSTM-CRF no conjunto de treino completo. Porém, como a CRF é baseada em *features* que verificam a palavra sem acento e em letras minúsculas, o desempenho melhorou diante desse tipo de ruído.

Na Figura 25 são mostradas as predições resultantes dos modelos de CRF treinados com o *machine-labeled corpus* de cada iteração do *bootstrapping*. O laudo utilizado é um exemplo que o microsserviço de estruturação com expressões regulares acertaria 100%. O único que acertou todas as entidades, nesse caso, foi o PA treinado com os *machine-labeled corpus* da segunda e terceira iterações (duas últimas colunas). Além disso, para esses dois, modificando-se o laudo para remover os acentos e/ou inverter a ordem das entidades não altera o resultado.

Figura 25 – Classificação das CRF L-BFGS e PA treinadas com o *machine-labeled corpus* de cada iteração para um laudo. A saída correta está representada nas duas últimas colunas do algoritmo PA.

	L-BFGS			PA		
Tokens	Predições (M1)	Predições (M2)	Predições (M3)	Predições (M1)	Predições (M2)	Predições (M3)
Aorta	O	O	O	O	O	O
com	O	O	O	O	O	O
calcificações	B-129748009	B-129748009	B-129748009	B-129748009	B-129748009	B-129748009
ateromatosas	B-48434008	B-48434008	B-48434008	B-48434008	B-48434008	B-48434008
.	O	O	O	O	O	O
No	O	O	O	O	O	O
geral	O	O	O	O	O	O
,	O	O	O	O	O	O
transparência	O	B-827032003	O	O	B-827032003	B-827032003
pulmonar	O	I-827032003	O	O	I-827032003	I-827032003
normal	O	I-827032003	O	O	I-827032003	I-827032003
,	O	I-827032003	O	O	O	O
apesar	O	I-827032003	O	O	O	O
da	O	I-827032003	O	O	O	O
opacidade	B-128308005	B-128308005	B-128308005	B-128308005	B-128308005	B-128308005
lateral	O	O	O	O	O	O
na	O	O	O	O	O	O
base	O	O	O	O	O	O
do	O	O	O	O	O	O
tórax	O	O	O	O	O	O
.	O	O	O	O	O	O

Fonte – Elaborada pela autora (2020).

Para ultrapassar o desempenho das expressões regulares, além da vetorização mais complexa (pré-treinar em um vocabulário para absorver as características do idioma português, por exemplo) para representar melhor as relações entre os *tokens* no caso da Bi-LSTM-CRF, seria necessário fazer ainda uma augmentação dos textos incluindo ruídos aleatórios (erros de digitação ou gramaticais, por exemplo). Outra possibilidade seria se um especialista anotasse os dados com padrões não previstos pelo microsserviço atual.

Como os modelos foram treinados com dados anotados pelas expressões regulares, que já haviam sido implementadas identificando algumas dessas questões, e nas condições apresentadas, é difícil que seja obtido um resultado superior. Apesar disso, as técnicas investigadas mostram que, caso fosse decidido usar o microsserviço atual para as anotações para novas modalidades médicas, as expressões poderiam ser muito mais simplificadas, ou seja, não precisariam prever tantas possibilidade de padrões de escrita.

6.2.2 Explicabilidade dos modelos

Para compreender melhor o que os modelos de CRF aprenderam, pode-se estudar as transições entre *tokens* e os estados das *features* mais fortes e fracas, algo que é proporcionado pela própria biblioteca *python-crfsuite*. As duas figuras a seguir mostram os estados para o *machine-labeled corpus* resultante da primeira iteração do *bootstrapping*, uma vez que para as seguintes o comportamento foi análogo.

A Figura 26 apresenta o estado das *features* que obtiveram uma pontuação mais positiva no aprendizado para as CRF. Pode-se observar que os modelos fixaram-se principalmente nos nomes das entidades e alguns aparecem mais de uma vez, o que demonstra que as *features lower*, *remove_accent* e ambas combinadas talvez tenham ficado redundantes, justamente porque as expressões regulares já cobriam a questão da acentuação.

No caso da CRF com PA, o modelo ainda verificou outras características. É interessante que a *feature bias* para a *tag* "O" também apareceu, o que é natural, uma vez que existem muitos exemplos da mesma. Isso mostra que, para esse algoritmo, essa *feature* não é muito adequada, uma vez que não é aplicada uma regularização como no L-BFGS. Felizmente, essa observação referente ao estado da *tag* "O"não afeta as métricas, já que ela não é contabilizada.

Curiosamente, a *feature* que detecta se a palavra tem mais de duas consoantes seguidas (*word.consonants*), identificou a entidade relacionada à infiltração, provavelmente pelo número de exemplos. A partir desses estados mais positivos, melhorias poderão ser implementadas nas funções de extração de *features* e também esclarece melhor o funcionamento de algumas delas.

Figura 26 – Estado das *features* mais positivas para as CRF com PA e L-BFGS treinadas com o *machine-labeled corpus* da iteração 1 do *bootstrapping*.

PA

```
B-128308005 word.lower():opacidade
B-128308005 word.remove_accent_lower():opacidade
B-129748009 word.lower():calcificações
B-129748009 word.remove_accent_lower():calcificacoes
B-129748009 word.lower():calcificação
B-129748009 word.remove_accent_lower():calcificacao
B-409609008 word.lower():infiltrados
B-409609008 word.remove_accent_lower():infiltrados
O      bias
B-409609008 word.consonants
```

L-BFGS

```
B-19923001 word.lower():cateter
B-699330001 word.lower():osteófitos
B-705655007 word.lower():esternorrafia
B-705655007 word.remove_accent_lower():esternorrafia
B-129748009 word.lower():calcificação
B-129748009 word.remove_accent_lower():calcificacao
B-15524008 word.lower():velamento
B-15524008 word.remove_accent_lower():velamento
B-107669003 word.lower():degenerativas
B-107669003 word.remove_accent():degenerativas
```

Fonte – Elaborada pela autora (2020).

Observando-se os estados mais negativos na Figura 27, é possível identificar em que formato as entidades são menos vistas. Por exemplo, na segunda coluna, para a iteração 1 do L-BFGS, a entidade "I-827032003", que representa o código SNOMED-CT para "transparência pulmonar", possui peso negativo para a *feature* EOS que indica que o *token* é o último da frase. Analisando o conjunto de dados, verifica-se que isso de fato é verdade, pois a entidade aparece muito mais no início dos laudos. Logo, o modelo aprendeu corretamente.

Figura 27 – Estado das *features* mais negativas para as CRF com PA e L-BFGS treinadas com o *machine-labeled corpus* da iteração 1 do *bootstrapping*.

PA	L-BFGS
B-827032003 bias	I-38341003 -1:word.istitle()
O word[-2:]:ia	I-302108003 word.istitle()
O -1:word.lower():sonda	I-363646005 +1:word.ispunct
O -1:word.remove_accent_lower():sonda	B-36118008 word.istitle()
O word[-2:]:vo	I-464205000 word.istitle()
O word.remove_accent():Osteofitos	B-827032003 word[-2:]:ão
O word.lower():atelectasia	B-38341003 +1:word.istitle()
O word.remove_accent_lower():atelectasia	B-45647009 +1:word.ispunct
O -1:word.ispunct	I-302108003 -1:word.istitle()
O -1:word.remove_accent():hilos	I-46621007 word.istitle()
O word.lower():fratura	B-8847002 -1:word.ispunct
O word.remove_accent lower():fratura	I-827032003 EOS

Fonte – Elaborada pela autora (2020).

Já o algoritmo PA, penalizou mais características observadas da *tag* "O". As informações da Figura 27 mostram que o modelo aprendeu que as *features* indicadas geralmente são entidades, ao invés de nenhuma. Também é possível pensar que o algoritmo aprendeu as entidades que aparecem o nome, porque penaliza a *tag* "O"para tais *tokens*.

As transições positivas e negativas também trazem outra visão sobre o aprendizado dos modelos. Para exemplificar mais objetivamente, na Figura 28 têm-se as informações referentes às CRF treinadas após a segunda iteração do *bootstrapping*.

Figura 28 – Transições entre *tokens* para as CRF treinadas com o *machine-labeled corpus* resultante da segunda iteração do *bootstrapping*.

PA

Transições Positivas	Transições negativas
I-827032003 -> I-827032003 1.663118	B-298493002 -> O -0.334885
B-827032003 -> I-827032003 0.968210	O -> I-162904008 -0.364948
O -> O 0.916393	I-419991009 -> O -0.379037
I-168710006 -> I-168710006 0.612741	I-363646005 -> O -0.394496
B-419991009 -> I-419991009 0.593019	O -> I-302108003 -0.416191
I-363646005 -> I-363646005 0.566744	O -> I-249674001 -0.421725
B-363646005 -> I-363646005 0.527973	B-363646005 -> O -0.432670
B-249674001 -> I-249674001 0.452184	I-302108003 -> O -0.440249
B-168733007 -> I-168733007 0.443552	O -> I-168710006 -0.465631
B-128601007 -> I-128601007 0.442325	O -> B-363646005 -0.514139
I-298493002 -> I-298493002 0.437149	I-827032003 -> O -0.534787
B-128308005 -> I-128308005 0.431307	O -> I-298493002 -0.536994
B-298493002 -> I-298493002 0.425597	B-827032003 -> O -0.589441
B-409609008 -> O 0.417352	O -> I-363646005 -0.652488
I-249674001 -> I-249674001 0.406368	O -> I-827032003 -1.465558

L-BFGS

Transições positivas	Transições negativas
I-168710006 -> I-168710006 7.121478	O -> I-38341003 -6.870613
I-249674001 -> I-249674001 7.048880	B-73725006 -> O -6.887734
B-60046008 -> I-60046008 6.949445	O -> I-46621007 -6.907615
I-298493002 -> I-298493002 6.214967	O -> I-302108003 -7.241248
B-128601007 -> I-128601007 5.442048	O -> I-162904008 -7.285249
I-827032003 -> I-827032003 5.142868	B-363646005 -> O -7.318498
B-827032003 -> I-827032003 4.979739	B-419991009 -> O -7.636442
B-464205000 -> I-464205000 4.675491	O -> I-168733007 -7.729662
B-142111000119108 -> B-419991009 4.381553	O -> I-128308005 -7.913727
B-302108003 -> I-302108003 4.226022	O -> I-168710006 -8.404425
O -> O 3.668523	B-409609008 -> I-827032003 -8.822429
B-168733007 -> I-168733007 3.642529	O -> I-249674001 -9.126418
B-249674001 -> I-249674001 3.510127	O -> I-298493002 -9.217397
B-419991009 -> I-419991009 3.423366	O -> I-363646005 -10.542582
B-38341003 -> I-38341003 3.033730	O -> I-827032003 -13.222705

Fonte – Elaborada pela autora (2020).

No caso da L-BFGS, é possível perceber que é muito comum que existam sequências de *tokens* que possuem a *tag* "O", até porque há muitos *tokens* sem entidades definidas. O modelo também parece ter compreendido corretamente que *tokens* que possuem essa *tag* não devem ser seguidos por entidades que começam com "I", como indicado nas transições negativas.

Para o algoritmo PA o mesmo pode ser observado, acrescentando ainda que

penaliza no caso de uma entidade com *tag* "B" seguida por "O" quando na verdade a mesma é composta por mais de um *token*, como é o caso da "363646005" (mediastino alargado). Também é verificado que entidades diferentes dificilmente estão em sequência, ou seja, geralmente há *tokens* sem uma entidade entre elas, ou seja, as transições mais fortes são de *tags* em sequência ou para "O".

6.2.3 Métricas de pesquisa e desenvolvimento

De acordo com o que foi estabelecido nos objetivos que constam na Introdução, devem ser definidas algumas métricas referentes à pesquisa e desenvolvimento relacionadas à solução proposta. Os três principais pontos envolvem recursos tecnológicos, humanos e manutenção.

A estrutura do atual microsserviço utiliza basicamente dois componentes da GCP: *Pub/Sub*, que serve para troca de mensagens entre sistemas, e *Kubernetes*, que roda efetivamente a aplicação dentro de um contêiner. Para a nova solução, propõe-se duas alternativas.

A primeira, que praticamente não alteraria em termos de infraestrutura, seria apenas salvar os modelos de NLP no *Storage* da GCP e carregá-los ao disponibilizar uma nova versão do microsserviço. Isso facilitaria o versionamento dos modelos também e praticamente a influência nos gastos seria pequena. A segunda possibilidade, que em termos de custo seria interessante, é trocar o *Kubernetes* por uma *Cloud Function*, que seria disparada ao receber uma notificação do Pub/Sub.

Já com relação aos recursos humanos, para escalar a solução para novas modalidades, o ideal seria ter um especialista para anotar as *labels* no conjunto de dados. Caso isso não seja possível, existe a vantagem de reduzir o tempo que hoje é gasto na implementação das expressões regulares, uma vez que foi verificado que elas podem ser simplificadas e também não necessitariam de testes automatizados (atualmente, para cada entidade existe uma expressão e para cada uma delas é escrito um teste).

O processo inicial de entendimento de dados, busca pelos códigos SNOMED-CT e validação com o médico a princípio não teria mudanças. Futuramente, pode-se aplicar automações para diminuir o tempo de alocação para facilitar esse desenvolvimento. Isso também auxiliaria no treinamento de novas pessoas da equipe, mas ainda assim, continua sendo viável apenas um integrante tratar dessas etapas.

Em termos de manutenção, podem ser levantados os pontos seguintes. Em alguns momentos poderá ser necessária a criação de novas *features*, no caso da CRF. Também há a inclusão de novos códigos SNOMED-CT e modalidades, que resulta na disponibilização de mais modelos (algo que já é automatizado). Além disso, pode ser preciso retreinar algum modelo para um novo cliente caso as métricas estejam abaixo de 80%.

7 CONCLUSÃO

Neste projeto de fim de curso foi proposta uma abordagem para estruturação automática de laudos médicos para extração de informações clínicas através de processamento de linguagem natural. Foram estudadas e testadas duas arquiteturas distintas, procurando compreender seu funcionamento e parâmetros. Também foi aplicada a técnica de *bootstrapping* para a criação de conjuntos de dados anotados a partir de um conjunto pequeno.

A decisão final de qual modelo utilizar em produção será tomada ao aplicar as alternativas para um conjunto de dados inédito, de um cliente novo, até porque já se sabe que existem melhorias a serem aplicadas. Ainda assim, todas as três configurações testadas cumpriram com as métricas *precision*, *recall*, *F1-score*, tanto micro quanto macro, acima de 80%.

A técnica de *bootstrapping* mostrou potencial para resolver o problema dos dados anotados disponíveis e acredita-se que poderá ser utilizada, inclusive, em outras tarefas que envolvem NLP na empresa. Porém, ressalta-se que o correto é validar essa hipótese através de outros testes, como diminuir a quantidade de exemplos do *seed corpus* para avaliar o número mínimo necessário para obtenção de boas métricas.

Mesmo que não se tenha disponibilidade de um especialista para a anotação dos dados (para a validação das *labels* será sempre imprescindível tê-lo), as expressões regulares podem ser usadas para isso. Inserindo métodos de augmentação (inserção de ruídos ou variações dos *tokens*) e vetorização (vocabulários pré-treinados, por exemplo) mais avançados para enriquecer o conjunto de treinamento, acredita-se que melhorias poderão ser obtidas. Outro ponto de atenção é o desbalanceamento das entidades, algo que é natural devido ao contexto de saúde (alguns achados médicos não são tão comuns), mas que pode ser contornado.

Em relação à parte de negócio, verificou-se que otimizações de operação, que impactam em provável diminuição de custos, são viáveis. Mesmo com apenas um profissional sendo responsável pelos trabalhos com NLP, algumas etapas poderiam ser facilitadas com a nova solução, como foi detalhado na seção 6.2.3.

Como sugestões de aperfeiçoamento, para garantir o melhor uso em produção, propõe-se alguns passos, listados a seguir:

- para melhorar a robustez dos modelos, aplicar técnicas de vetorização das frases mais complexas, augmentar dados e tratar o desbalanceamento;
- para tentar obter um melhor desempenho no caso da Bi-LSTM-CRF, sugere-se, além do item anterior, implementar o chamado *early stopping*, que serve para monitorar o erro de generalização durante o treinamento;

- tornar os códigos implementados mais eficientes, utilizando outras bibliotecas de *Python*, como *Dask* ou *Vaex*;
- incluir funções de similaridade para extrair a informação de presença das entidades (por exemplo, extrair "Ausência de consolidação" e indicar que "consolidação" possui a presença *NEGATED*);
- automatizar algum passo da análise dos dados para facilitar a buscar pelos códigos SNOMED-CT.

Vale destacar ainda que seria interessante testar a solução com dados anotados por um especialista, para uma confiabilidade ainda maior. Além disso, existe também a questão de entidades que se sobrepõem, algo que não foi abordado neste projeto.

A partir dos resultados obtidos pode-se afirmar que os objetivos estabelecidos foram cumpridos. Será possível substituir, gradativamente, o microsserviço de estruturação de laudos atual, que utiliza expressões regulares, por modelos de NLP, uma vez que se provaram ser mais adaptáveis e escaláveis (já que bastará aplicar os dados a um fluxo de pré-processamento, treinamento e avaliação bem definidos). Em breve, espera-se que a solução seja aplicada para raio-X em um novo cliente e também para laudos de tomografia, que possuem achados médicos semelhantes.

REFERÊNCIAS

ACADEMY, Data Science. **Deep Learning Book**. 2019. Disponível em: http://www.deeplearningbook.com.br/.

ALGORITHMIA. **Introduction to Optimizers**. Mai. 2018. Disponível em: https://algorithmia.com/blog/introduction-to-optimizers.

ARBEL, Nir. **How LSTM networks solve the problem of vanishing gradients**. Dez. 2018. Disponível em: https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577.

CHOWDHURY, Gobinda G. Natural Language Processing. **Annual Review of Information Science and Technology**, 2005.

DINIZ, Pedro. **CNN**. Disponível em: https://colab.research.google.com/drive/1pbbIqPmZiLoFKDYvmZiF4tHNspTmLi42#scrollTo=sj9irTjAmOVO.

FÁBIO SOUZA, Rodrigo Nogueira; LOTUFO, Roberto. Portuguese Named Entity Recognition using BERT-CRF, fev. 2020.

GIORGI, John M.; BADER, Gary D. Towards reliable named entity recognition in the biomedical domain. **Bioinformatics Oxford**, 2019.

HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. **Neural Computation**, 1997.

HUILGOL, Purva. **Accuracy vs. F1-Score**. Ago. 2019. Disponível em: https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2#:~:text=Accuracy%5C%20is%5C%20used%5C%20when%5C%20the,as%5C%20in%5C%20the%5C%20above%5C%20case..

JI, X.; CHUN, S. A.; GELLER, J. Monitoring Public Health Concerns Using Twitter Sentiment Classifications. *In*: 2013 IEEE International Conference on Healthcare Informatics. [*S.l.*: *s.n.*], 2013. P. 335–344.

JOHN LAFFERTY, Andrew McCallum; PEREIRA, Fernando C.N. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, jun. 2001.

KIM, J.; KO, Y.; SEO, J. A Bootstrapping Approach With CRF and Deep Learning Models for Improving the Biomedical Named Entity Recognition in Multi-Domains. **IEEE Access**, 2019.

LIU, Zengjian; YANG, Ming; WANG, Xiaolong; CHEN, Qingcai; TANG, Buzhou; WANG, Zhe; XU, Hua. Entity recognition from clinical texts via recurrent neural network. **BMC Med Inform Decis Mak**, 2017.

LUOEN, Ma. Reconhecimento de fala e processamento de linguagem natural: Módulo que permita assistir os médicos no bloco de operatório a partir de comandos de voz. **Repositório Científico do Instituto Politécnico do Porto**, 2016.

NACIONAL, Jornal. **Entenda como é feita a coleta de dados da Covid-19 em todo o país**. Jun. 2020. Disponível em: https://g1.globo.com/jornal-nacional/noticia/2020/06/26/entenda-como-e-feita-a-coleta-de-dados-da-covid-19-em-todo-o-pais.ghtml.

NET, Oficina da. **O que são as redes neurais artificiais?** Fev. 2019. Disponível em: https://www.oficinadanet.com.br/tecnologia/25007-o-que-sao-as-redes-neurais-artificiais.

OKAZAKI, Naoaki. **CRFsuite: a fast implementation of Conditional Random Fields (CRFs)**. [*S.l.*: *s.n.*], 2007. Disponível em: http://www.chokkan.org/software/crfsuite/.

ORACLE. **O que é um Chatbot?** Disponível em: https://www.oracle.com/br/solutions/chatbots/what-is-a-chatbot/.

SHMUELI, Boaz. **Multi-Class Metrics Made Simple, Part II: the F1-score**. Jul. 2019. Disponível em: https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1.

SNOMED-CT. **SNOMED-CT Browser**. Disponível em: https://browser.ihtsdotools.org/?.

WIKIPEDIA. **Gradiente descent**. Disponível em: https://en.wikipedia.org/wiki/Gradient_descent.

WIKIPEDIA. **Sentiment analysis**. Disponível em: https://en.wikipedia.org/wiki/Sentiment_analysis.

YSE, Diego Lopez. **Your Guide to Natural Language Processing (NLP)**. Jan. 2019. Disponível em: https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1.

ZAMBIASI, Saulo Popov. **O Neurônio Artificial**. 2011. Disponível em: https://www.gsigma.ufsc.br/~popov/aulas/rna/neuronio_implementacao/.

ZHIHENG HUANG, Wei Xu; YU, Kai. Bidirectional LSTM-CRF Models for Sequence Tagging, jan. 2015.

APÊNDICE A – CÓDIGOS SNOMED-CT E EQUIVALÊNCIAS

Todas as 83 equivalências entre os códigos e a tradução para português apresentadas foram validadas por um médico especialista. Vale destacar que existem os sinônimos das expressões que, por uma questão de simplicidade, não foram incluídos na lista.

SNOMED-CT	**Equivalência**	**Tipo**
399269003	Artrose	*Condition*
46621007	Atelectasia	*Condition*
142111000119108	Ectásica	*Condition*
27779004	Esternotomia	*Procedure*
48434008	Ateromatosa	*Finding*
8186001	Cardiomegalia	*Condition*
129748009	Calcificação	*Finding*
298493002	Hipercifose	*Condition*
13645005	DPOC	*Condition*
107669003	Degenerativa	*Finding*
15524008	Obliteração	*Finding*
162904008	Pulmão hipoexpandido	*Finding*
249674001	Hiperinsuflação pulmonar	*Finding*
409609008	Infiltrado	*Condition*
827032003	Pulmão normal	*Finding*
95436008	Consolidação	*Condition*
309529002	Massa pulmonar	*Finding*
67599009	Pulmão congesto	*Condition*
60046008	Derrame pleural	*Condition*
85090007	Dextrocardia	*Finding*
87433001	Enfisema	*Condition*
73725006	Espessamento pleural	*Condition*
8847002	Espondilose	*Condition*
298382003	Escoliose	*Condition*
705655007	Esternorrafia	*Product*
51615001	Fibrose	*Condition*
125605004	Fratura	*Condition*
45647009	Granuloma	*Finding*
363646005	Mediastino alargado	*Finding*
27925004	Nódulo	*Finding*
86838002	Nódulo pulmonar	*Finding*

255003	Nódulo calcificado	*Finding*
45181002	Hérnia de Schmorl	*Finding*
128308005	Opacidade	*Finding*
312894000	Osteopenia	*Condition*
699330001	Osteófitos	*Finding*
36118008	Pneumotórax	*Condition*
233604007	Pneumonia	*Condition*
168710006	Tecidos moles normais	*Finding*
168733007	Raio-X de tórax normal	*Finding*
449842009	Sonda nasogástrica	*Finding*
464205000	Sonda nasoentérica	*Product*
32712000	Dreno	*Product*
19923001	Cateter	*Product*
441509002	Marcapasso	*Finding*
2282003	Prótese mamária	*Product*
737277001	Prótese válvula cardíaca	*Finding*
419991009	Tubo endotraqueal	*Finding*
302108003	Tubo traqueostomia	*Finding*
386011001	Clipes vasculares	*Product*
40617009	Assistência ventilatória	*Procedure*
705749009	Parafuso	*Product*
234083009	Punção venosa	*Condition*
32398004	Bronquite	*Condition*
442095009	Artroplastia	*Procedure*
23680005	Entesopatia	*Condition*
405773007	Cifoscoliose	*Condition*
128601007	Infecção pulmonar	*Condition*
64859006	Osteoporose	*Condition*
443092002	Calo ósseo	*Finding*
463037002	Clipes metálicos	*Product*
191394000	Linfonodos calcificados	*Condition*
84089009	Hérnia hiatal	*Condition*
125571002	Lobectomia	*Procedure*
69031006	Mastectomia	*Procedure*
65818007	Stent	*Product*
19578002	Artrodese	*Procedure*
16470007	Eletrodos	*Product*

261531000	Costelectomia	*Procedure*
82542004	Pneumopericárdio	*Condition*
52299001	Osteocondroma	*Finding*
391987005	Peito escavado	*Condition*
38774000	*Pectus carinatum*	*Condition*
396285007	Broncopneumonia	*Condition*
54150009	IVAS	*Condition*
38341003	Hipertensão arterial	*Condition*
32048006	Adenoma	*Finding*
17204006	Pneumoperitônio	*Condition*
396275006	Osteoartrite	*Condition*
17808001	Veia ázigos	*Condition*
70995007	Hipertensão pulmonar	*Condition*
3716002	Hipertrofia tireoidiana	*Condition*
267038008	Edema	*Finding*