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

Maria Fernanda Dias Portella

Geração de Laudos Médicos por Reconhecimento de Fala

Florianópolis
2020

Maria Fernanda Dias Portella

Geração de Laudos Médicos por Reconhecimento de Fala

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 Aislan Antonelo, Dr.
Supervisor: Rodolfo Rodrigues Moreira

Florianópolis
2020

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

Portella, Maria Fernanda Dias
Geração de laudos médicos por reconhecimento de fala / Maria Fernanda Dias Portella ; orientador, Eric Aislan Antonelo, 2020.
46 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. Reconhecimento de fala. 4. Rede neural convolucional. I. Antonelo, Eric Aislan. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Maria Fernanda Dias Portella

Geração de Laudos Médicos por Reconhecimento de Fala

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, 15 de Dezembro de 2020.

Prof. Hector Bessa Silveira, Dr.
Coordenador do Curso

Banca Examinadora:

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

Rodolfo Rodrigues Moreira
Supervisor
Empresa

Jhonatan Alves
Avaliador
Instituição

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

AGRADECIMENTOS

Agradeço, primeiramente, aos meus pais - Paulo e Vera - que me possibilitaram essa jornada acadêmica. E ao meu namorado, Daniel Franzoni, que me acompanhou e me apoiou durante a fase de desenvolvimento deste projeto.

Agradeço também ao meu orientador, professor Eric Antonelo, que me guiou nesse período, procurando sempre me estimular e me direcionar para concluir o trabalho com êxito.

E por último, um agradecimento especial ao meu mentor na empresa, Rodolfo Moreira, pelas supervisões nos projetos e pelos ensinamentos que me permitiram melhorar o desempenho no processo de formação profissional.

RESUMO

O objetivo principal desse projeto é a criação de um modelo de aprendizado de máquina para a geração de laudos médicos utilizando reconhecimento de fala. A fim de concluir este projeto com sucesso, foi realizado um aprofundamento nos estudos de aprendizado de máquina e a simulação de um modelo simples de rede neural convolucional para reconhecimento de áudio. Por fim, o trabalho apresenta a elaboração do modelo de reconhecimento de fala focado em geração de laudos médicos que permite a transcrição do laudo em tempo real. Além de elaborar um dicionário médico para este fim, a aplicação desenvolvida também possui modelos pré formatados de laudos, ferramentas para formatação, atalhos de voz para a realização de alguns tipos de ações e mecanismo para a correção de palavras não reconhecidas, permitindo uma constante atualização do conjunto de dados.

Palavras-chave: Aprendizado de máquina. Reconhecimento de fala. Rede neural convolucional.

ABSTRACT

The main objective of this project is the creation of a machine learning model for the generation of medical reports using speech recognition. In order to successfully complete this project, some machine learning related themes were studied and the simulation of a model of convolution neural network was trained. Finally, this document presents the elaboration of the speech recognition model focused on generation medical reports that allows the transcription in real time. Besides preparing a medical dictionary for this purpose, the developed application also has preformatted report templates, formatting tools, voice shortcuts and a mechanism for correcting unrecognized words, allowing a constant update of the data set.

Keywords: Machine learning. Speech recognition. Convolutional neural network.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A implementação de soluções utilizando técnicas de inteligência artificial e aprendizado de máquina estão cada vez mais presentes no cotidiano. Com isto, surgem novas necessidades de aprimorar e agilizar processos que outrora eram realizados manualmente. Neste projeto, o objetivo é a otimização de geração de laudos médicos utilizando reconhecimento de fala. O processo, em um contexto geral, funciona através de uma ferramenta que permita a transcrição do laudo em tempo real, bem como outros mecanismos que tornem este processo mais fluido e que permita sua melhoria constante.

A fala é um instrumento extremamente utilizado nas relações humanas e muito importante para a comunicação entre profissionais da saúde e seus pacientes. Com a agilidade que a atual sociedade exige em seus atendimentos, os médicos têm a necessidade de acelerar os processos de atividades, como a geração de laudos. O reconhecimento de fala para elaboração de laudos é visto como uma ferramenta facilitadora no trabalho cotidiano do médico e com o desenvolvimento desta ferramenta espera-se otimizar o trabalho do médico agregando agilidade no atendimento.

Apesar de já existirem diversas ferramentas de reconhecimento de fala no mercado atual, há uma dificuldade de utilização destas, principalmente quando se trata de um cenário específico, como a área médica. Desta forma, além de realizar a construção inicial de um vocabulário contendo termos médicos, o projeto também possibilita uma fácil utilização da ferramenta, permitindo também o gerenciamento descomplicado da mesma.

Para atingir os resultados esperados em um espaço curto de tempo, o projeto foi realizado tendo como base a metodologia ágil, que será abordada posteriormente. A gestão do projeto e as reuniões com o cliente ficaram a cargo do líder técnico, que definiu a sequência de atividades a serem realizadas e supervisionou os resultados. Todos os passos e decisões foram sempre compartilhados.

Neste documento, será realizada uma breve apresentação da empresa no qual foi realizado o projeto de fim de curso e serão apresentadas as ferramentas e metodologias utilizadas para o desenvolvimento deste projeto. Para a melhor compreensão do trabalho como um todo, será realizada uma contextualização teórica sobre os assuntos a serem abordados. Após estas apresentações, o documento conta com um capítulo explicativo da simulação de um modelo simples de rede neural convolucional para reconhecimento de áudio. Com isto, será apresentado por fim o projeto realizado para a geração de laudos médicos através do reconhecimento de fala.

1.1 OBJETIVOS

O objetivo geral deste projeto é apresentar uma solução para a geração de laudos médicos através do reconhecimento de fala.

Os objetivos específicos são:

- Realizar o aprofundamento do estudo de aprendizado de máquina;
- Simular um modelo simples de rede neural convolucional para reconhecimento de áudio;
- Elaborar um modelo de reconhecimento de fala e apresentar um case realizado nesta área.

2 A EMPRESA

A Radix (do latim "raiz", "origem") é uma empresa multinacional de tecnologia e engenharia e oferece serviços e soluções digitais altamente qualificados e com independência tecnológica para atender as principais indústrias de processo do Brasil e do mundo.

A empresa chegou ao mercado em abril de 2010 e hoje possui sedes no Rio de Janeiro, São Paulo, Belo Horizonte e em Houston - além de atuação em todos os continentes do planeta. Para isso, conta com um time de gerentes e consultores multidisciplinares, nas áreas de Engenharia, Automação & TI Industrial e Desenvolvimento de Software.

Dentre seus valores, estão ética, inovação, comprometimento e foco no ser humano. Este último é um pilar decisivo para o sucesso do negócio, levando a Radix há dez anos dentre as Melhores Empresas para se Trabalhar nestes anos de vida da empresa. Atualmente a Radix foi eleita, na pesquisa promovida pelo Great Place To Work (GPTW), a Melhor Empresa para se Trabalhar no Rio de Janeiro e também no Brasil na categoria de média empresa, além de ter conquistado o segundo lugar das Melhores Empresas para se Trabalhar em Tecnologia da Informação.

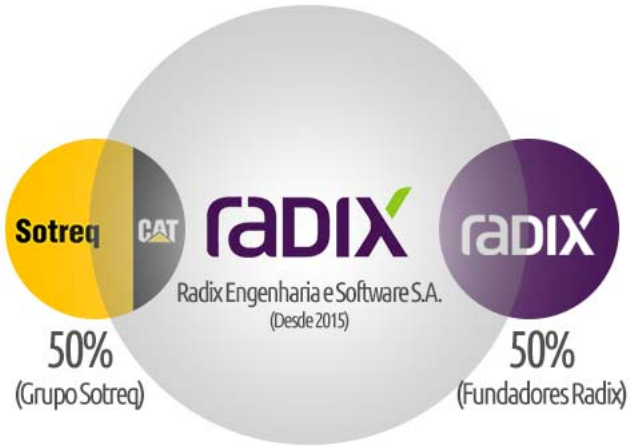
A empresa também é uma das sete empresas atuantes no Brasil que possui certificado de maturidade e desenvolvimento CMMI Nível 5, que fornece às organizações elementos essenciais de processos eficazes. Este é outro motivo que faz com que a empresa possua clientes em todos os segmentos de negócios, como óleo e gás, energia, metais e mineração, papel e celulose, transportes, finanças, varejo, entre outros.

No início de 2015, a empresa apostou na macro diversificação e conquistou um novo sócio para capitalizar seus negócios: a Sotreq, maior revendedora de produtos Caterpillar e também com capital 100% nacional. A atual divisão societária da Radix está representada na Figura 1.

2.1 SOLUÇÕES TECNOLÓGICAS

A Radix investe não apenas em soluções tecnológicas, mas também no suporte estratégico à tomada de decisão do cliente. As tecnologias desenvolvidas pela empresa aumentam a eficiência e agregam valor ao negócio do cliente. Algumas destas tecnologias surgiram de soluções, frameworks e metodologias elaboradas de forma customizada para empresas, sempre considerando o que já existe no mercado, atentando à relação custo e benefício e seguindo os melhores conceitos de engenharia de projetos.

Figura 1 – Divisão societária.



Fonte – Radix.

3 METODOLOGIA E FERRAMENTAS

3.1 METODOLOGIA

A metodologia que orquestrou o projeto foi o "Scrum". Scrum é uma abordagem para gestão e planejamento de projetos, que visa diminuir a quantidade de pessoas envolvidas no projeto, assim como o tempo de desenvolvimento. Tem como intuito prover mais resultados de melhor qualidade e com um menor custo.

O termo Scrum vem do rúgbi e se refere à maneira como um time se une para avançar com a bola pelo campo. Tudo se alinha: posicionamento cuidadoso, unidade de propósito e clareza de objetivo. Trata-se de uma analogia perfeita para o que se espera das equipes (SUTHERLAND; SUTHERLAND, 2014).

Nesta metodologia, os projetos são divididos em ciclos chamados de "sprints". No início de cada ciclo, deve-se planejar o sprint que pode ser de uma semana a um mês. Um sprint representa um espaço de tempo dentro do qual um conjunto de atividades deve ser realizado. As tarefas a serem realizadas são definidas pela equipe a partir da lista de prioridades do projeto.

A cada dia de um sprint, a equipe faz uma reunião diária rápida, com todos de pé. O objetivo é manter todos informados sobre o que já foi feito, identificar desafios e priorizar as próximas atividades.

Ao fim do sprint, a equipe mostra o que conseguiu realizar naquele tempo e parte para o planejamento do próximo sprint. Assim reinicia-se o ciclo.

No caso do projeto atual, a equipe era pequena. Desta forma, o conhecimento era continuamente compartilhado entre os membros e os desafios eram igualmente divididos e resolvidos de maneira bastante dinâmica.

A cada novo passo do projeto, eram realizados estudos e testes para posteriormente serem implantados no sistema desenvolvido. Isto tornou o projeto ainda mais ágil e eficiente.

3.2 FERRAMENTAS

Dentre os estudos e soluções, foram considerados diversos aspectos para a escolha das ferramentas utilizadas. As ferramentas escolhidas e algumas das que foram consideradas nos estudos serão descritas a seguir.

3.2.1 Python

O Python é uma linguagem de programação criada no início dos anos 1990 por Guido van Rossum na Holanda como sucessor de uma linguagem chamada ABC. Apesar de Guido ser o principal autor do Python, seu desenvolvimento e manutenção contam com contribuição de muitos outros autores.

Todas as versões do Python, desde a sua criação, são de código aberto. Em 2001, a *Python Software Foundation* foi fundada, uma organização sem fins lucrativos criada especificamente para possuir propriedade intelectual relacionada ao Python.

Python é uma linguagem de programação de alto nível e permite que a escrita seja compacta e legível. Oferece estrutura e suporte para diversos programas. E é uma linguagem livre e multiplataforma. Dessa maneira, os programas escritos em Python são executados na maioria das plataformas existentes sem problemas.

Também possui uma biblioteca padrão bastante extensa, que contém classes, métodos e funções para realizar praticamente qualquer tarefa. Além de possibilitar o uso de diversas outras bibliotecas, para realizar os mais diversos tipo de função.

3.2.2 Flask

Flask é uma estrutura base minimalista com ferramentas e componentes para agilizar o processo de desenvolvimento de uma aplicação. Esta estrutura é desenvolvida em Python e possui como pressuposto não atrapalhar o desenvolvimento, sendo simples, rápido, apresentando boas soluções para projetos e aplicações robustas.

3.2.3 CRUD

CRUD é um acrônimo do idioma inglês das quatro operações básicas utilizadas em um sistema que trabalha com banco de dados. São elas: criação (*create*), consulta (*read*), atualização (*update*) e exclusão (*delete*) dos dados.

Do ponto de vista do desenvolvedor do sistema, essas ações serão necessárias para criar as tabelas do banco de dados e as funções que atualizarão o banco, assim como as interfaces como página na internet ou aplicativo para celular, nos quais o usuário interage com esses dados.

3.2.4 Serviços Cognitivos

As principais ferramentas que possuem serviços cognitivos disponíveis no mercado são oferecidas por empresas como AWS, Google, IBM e Microsoft. A fim de escolher a melhor plataforma *Speech to Text* para o projeto, realizou-se pesquisa destas principais ferramentas, além de ter sido também realizada a busca por ferramentas *open source*. As pesquisas incluíram principalmente pontos como:

- A forma como é feita a integração da ferramenta;
- Se a ferramenta possui integração offline;
- Se a ferramenta possui integração com mobile. Caso positivo, com quais plataformas;

- Nível do reconhecimento de voz;
- Se a solução da ferramenta já apresenta possibilidades de identificar partes do texto em categorias e salvá-las para pesquisas futuras.

3.2.4.1 AWS

Amazon Transcribe é a ferramenta disponibilizada pela AWS que permite a adição de recursos de conversão de fala para texto à aplicações. Esta ferramenta utiliza um processo de aprendizagem profunda chamada reconhecimento automático de fala para converter fala em texto.

A AWS possui também sua própria ferramenta de conversão de fala médica para texto, a Amazon Transcribe Medical. Este serviço utiliza aprendizado de máquina para transcrever com precisão terminologias médicas, como nomes de medicamentos, procedimentos e até mesmo condições ou doenças.

A integração da ferramenta é realizada através da configuração da interface de linha de comandos para desenvolver uma chamada ao SDK (*Software Development Kit*). A chamada, por sua vez, pode ser enviada pela interface de linha de comandos ou atráves do console da Amazon.

A integração offline é possível se os arquivos estiverem no armazenamento S3 da AWS. Neste formato, a conversão está disponível apenas em espanhol e inglês.

A integração da ferramenta com mobile é com SDK para JavaScript. Porém não possui API REST.

Não há nada especificado na solução a respeito da identificação de partes do texto em categorias. Mas há a possibilidade de enviar palavras e frases como dicas para que a solução melhore a transcrição. Este mecanismo é conhecido como Vocabulário Personalizado.

3.2.4.2 Google

A ferramenta de *Speech to Text* do Google permite a conversão de voz em texto utilizando uma API desenvolvida com as tecnologias de inteligência artificial do próprio Google.

Sua integração é feita através da biblioteca de cliente. Deve-se criar o aplicativo e configurar o projeto no console do Google Cloud Platform (GCP), instalar SDK e biblioteca de cliente, desenvolver a chamada da API em várias linguagens de programação utilizando as credenciais do GCP.

O *Cloud Speech to Text* permite fácil integração das tecnologias de reconhecimento de fala do Google nos aplicativos. O reconhecimento dos arquivos pode ser síncrono (com áudios de curta duração - até 1 minuto), assíncrono (com áudio de longa duração - até 480 minutos) - ambos REST e gRPC - ou de streaming (tempo real,

captando áudio de um microfone). O conteúdo de áudio pode ser enviado diretamente para o *Cloud Speech to Text* ou pode ser processado conteúdo de áudio que já reside no Google Cloud Storage.

Com mobile, a ferramenta do Google possui integração com Android e IOS por RPC API.

Assim como a ferramenta da AWS, a *Speech to Text* do Google também tem a possibilidade de enviar palavras e frases como dicas para que a solução melhore a transcrição, chamado de *Speech Adaptation*. Também há a possibilidade de adicionar palavras ou pequenas frases através do *Speech Contexts*.

3.2.4.3 IBM

O *Watson Speech to Text* é a ferramenta da IBM para transcrever áudio para texto em diferentes idiomas, incluindo português brasileiro. O serviço *Speech to Text* oferece uma API para incluir recursos de transcrição de fala em aplicativos. Ele combina informações sobre a estrutura de linguagem com a composição do sinal de áudio.

Sua integração é feita usando SDK e API para autenticação. E a integração com mobile utiliza SDK Swift.

Assim como as ferramentas anteriores, esta tem a possibilidade de enviar palavras e frases como dicas para que a solução melhore a transcrição, chamada de *Custom Words*.

3.2.4.4 Microsoft

O recurso de conversão de fala em texto oferecido pela Microsoft Azure é disponibilizado em mais de 40 idiomas e permite a personalização de modelos para aumentar a precisão da terminologia desejada.

Sua integração é feita através da criação de um *speech service*, após isto é criado um aplicativo e obtido um *endpoint* que será utilizado ao desenvolver a chamada ao speechsdk.

É possível capturar áudio de um microfone, ler de um fluxo ou acessar arquivos de áudio do armazenamento com o SDK de fala e as APIs REST. O SDK de fala é compatível com WAV/PCM 16 bits, 16 kHz/8 kHz, áudio de canal único para o reconhecimento de fala. Outros formatos de áudio são compatíveis com o uso do ponto de extremidade REST de conversão de fala em texto ou o serviço de transcrição em lote.

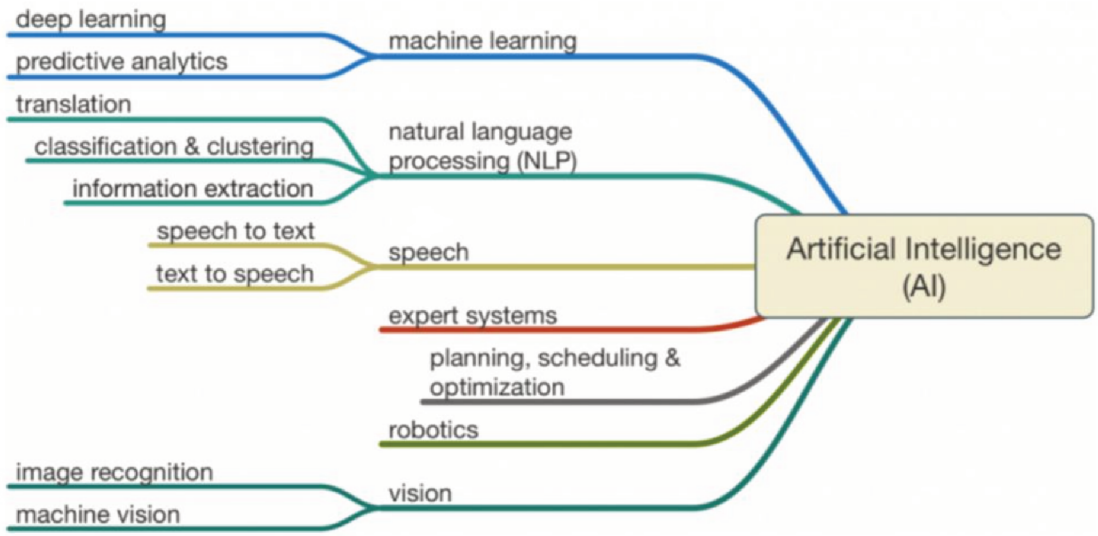
Para integração offline, deve ser criado um *WebHook*. E sua integração com mobile através de SDK de fala está disponível em muitas linguagens de programação e em todas as plataformas. A integração da API da Microsoft com o Android é feita através da linguagem de programação Java. Já para iOS, pode ser feita através das linguagens Objective C ou Swift.

Por padrão, o serviço de *speech to text* usa o modelo de linguagem universal. Esse modelo é treinado usando dados de propriedade da Microsoft e é implantado na nuvem. É ideal para cenários de conversação e de ditado. Se for utilizada a conversão de fala em texto para funcionalidades de reconhecimento e transcrição em um ambiente exclusivo, é possível criar e treinar modelos acústicos, de idioma e de pronúncia personalizados para lidar com o ruído ambiente ou vocabulário específico do setor. Também possui a possibilidade de enviar palavras e frases como dicas para que a solução melhore a transcrição, chamado de *Phrase Lists*.

4 FUNDAMENTAÇÃO TEÓRICA

A inteligência artificial é um ramo da ciência da computação que possui foco na construção de computadores capazes de simular o comportamento inteligente similar ao humano. Existem diversos conceitos associados a este avanço tecnológico, como pode-se visualizar na Figura 2, dentre eles está o conceito de aprendizado de máquina.

Figura 2 – Áreas de estudo da inteligência artificial.



Fonte – Arquivo da Universidade Radix.

4.1 APRENDIZADO DE MÁQUINA

A definição de aprendizado de máquina varia muito entre autores. Segundo Samuel (1959) o aprendizado de máquina é um campo de estudo que fornece aos computadores a habilidade de aprender sem que sejam explicitamente programados. Já para Mitchell (1998) é um programa de computador que aprende através da experiência E com respeito a um tipo de tarefa T e performance P, se sua performance na tarefa T como medido por P, melhoram com a experiência E.

Em geral, o aprendizado de máquina é uma solução ótima para:

- Problemas para os quais as soluções existentes necessitam de muito ajuste manual;
- Problemas complexos para os quais não existe uma boa solução;
- Ambientes com dados flutuantes;
- Obter ideias sobre problemas complexos;

- Lidar com grandes quantidades de dados.

Existem diferentes tipos de aprendizado de máquina e eles podem ser classificados com base em diferentes características, como:

- Se eles são ou não treinados com supervisão humana: aprendizado supervisionado, aprendizado não supervisionado, aprendizado semissupervisionado e aprendizado por reforço;
- Se eles podem ou não aprender de forma incremental (aprendizado online ou aprendizado em lote);
- Se eles funcionam comparando novos dados com dados conhecidos ou se detectam padrões nos dados de treinamento para construir um modelo preditivo (aprendizado baseado em instância ou aprendizado baseado em modelo);

O aprendizado de máquina conquistou a indústria e vem avançando rapidamente nos últimos anos. A técnica tem atraído produtos e serviços de alta tecnologia e se estendendo para diversas aplicações. Uma destas aplicações é o reconhecimento de fala que será abordado posteriormente.

4.1.1 Aprendizado Supervisionado

Como citado anteriormente, os sistemas de aprendizado de máquina podem ser classificados de acordo com a supervisão humana que recebem. No caso deste projeto, o tipo de aprendizado utilizado foi o supervisionado.

O aprendizado supervisionado é o tipo de aprendizado de máquina mais comum. Em poucas e simples palavras, este tipo de aprendizado ensina o computador a fazer algo, enquanto que o aprendizado não supervisionado deixa o computador aprender por ele próprio.

Neste tipo de aprendizado, os dados de treinamento fornecidos ao algoritmo incluem as soluções desejadas, chamadas de rótulos. O aprendizado supervisionado ocorre quando já existe um conjunto de dados rotulado e se sabe qual a saída correta esperada, que deve ser semelhante ao conjunto de dados e que a entrada e a saída possuem relação entre si.

Existem dois tipos de problemas dentro do aprendizado supervisionado: classificação e regressão. Para problemas de classificação existe uma variável categórica. O filtro de spam no e-mail é um bom exemplo. Ele é treinado com muitos e-mail de exemplo junto com cada classe (spam ou não spam) e deve aprender a classificar novos e-mails. Enquanto para problemas de regressão existe uma variável numérica alvo, como o preço de um carro, dado um conjunto de informações sobre o veículo

(quilometragem, marca, idade, etc.). Para treinar o sistema, deve-se dar a ele muitos exemplos de carros, juntamente com suas informações e seus preços.

Alguns dos principais algoritmos de aprendizado supervisionado são:

- Regressão Linear;
- Regressão Logística;
- k-Nearest Neighbors (kNN);
- Máquinas de Vetores de Suporte (SVMs);
- Árvores de Decisão e Florestas Aleatórias;
- Redes Neurais.

Neste projeto, o algoritmo de maior interesse é o de redes neurais que será abordado mais profundamente a seguir.

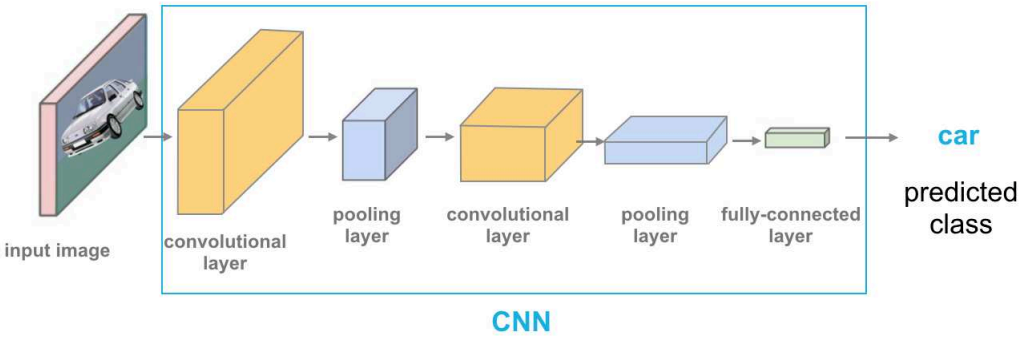
4.1.2 Redes Neurais Convolucionais

As redes neurais são modelos computacionais inspirados no sistema nervoso central de um animal. Por este motivo, são chamadas de redes neurais artificiais. Surpreendentemente, elas já existem há um bom tempo. Foram introduzidas pela primeira vez em 1943, pelo neurofisiologista Warren McCulloch e pelo matemático Walter Pitts. Eles apresentaram um modelo computacional simplificado de como os neurônios biológicos no cérebro de animais podem trabalhar juntos para realizar cálculos complexos usando lógica proposicional. Esta foi a primeira arquitetura de rede neural artificial. Após isto, diversas outras arquiteturas foram criadas, como a rede neural convolucional, que é bastante aplicada em processamentos e análises de imagens e sons.

As redes neurais convolucionais (CNNs), também conhecidas como ConvNets, surgiram do estudo da visão do córtex cerebral e são utilizadas para o reconhecimento de imagens desde os anos 1980. Recentemente, devido ao aumento do poder computacional, a quantidade de aplicações nas quais este algoritmo é utilizado aumentou consideravelmente. Eles podem ser aplicados em serviços de pesquisa de imagens, carros autônomos, sistemas automáticos de classificação de vídeo. Mas não se limitam aos aspectos visuais, também são utilizados em tarefas como reconhecimento de fala ou processamento de linguagem natural.

A primeira aplicação bem sucedida utilizando rede neural convolucional foi desenvolvida em 1998 por Yann LeCun. Esta aplicação possuía sete camadas e sua arquitetura se concentrava no reconhecimento de caracteres de caligrafia. A arquitetura ficou conhecida como LeNet-5 e uma representação didática da mesma pode ser visualizada na Figura 3.

Figura 3 – Arquitetura LeNet-5.



Fonte – Analytics India Magazine.

Na Figura 3, vê-se como entrada uma imagem que passa alternadamente pelas camadas convolucionais e de *pooling*, por fim passa por duas camadas *fully connected* seguidas. Com isto, tem-se a saída ou o resultado que, neste caso, é a predição do conteúdo da imagem. Os tipos de camadas abordados aqui, serão explicados a frente.

As ConvNets são projetadas para processar dados em forma de matrizes múltiplas. Diversas modalidades de dados se encontram na forma dessas matrizes. Podem ser dados em uma dimensão (1D) como sinais e sequências, duas dimensões (2D) como imagens e espectrogramas de áudio e três dimensões (3D) como vídeos e imagens volumétricas.

A ideia principal de uma rede convolucional é filtrar linhas, curvas e bordas e, em cada camada acrescida, transformar essa filtragem em uma imagem mais complexa. Para isto, a arquitetura de uma ConvNet é estruturada como uma série de etapas e conta com quatro conceitos baseados nas propriedades de sinais naturais: conexões locais, pesos compartilhados, *pooling* e o uso de muitas camadas.

Os primeiros estágios da arquitetura são compostos por dois tipos de camadas: camadas convolucionais e camadas de pooling. As convoluções funcionam como filtros que enxergam pequenos quadrados e passam por toda a imagem captando os traços mais marcantes. A profundidade da saída de uma convolução é igual a quantidade de filtros aplicados. Quanto mais profundas as camadas das convoluções, mais detalhados os traços identificados. Já a camada de *pooling* serve para simplificar a informação da camada anterior. Assim como a convolução, a camada de *pooling* transita por toda a saída da camada anterior e resume a informação desta.

O resultado destas camadas deve passar por uma função de ativação a fim de trazer a não linearidade ao sistema, ou seja, para que a rede consiga aprender qualquer tipo de funcionalidade. Existem diversas funções com este fim, porém a mais indicada para utilizar com redes convolucionais é a ReLU, por ser mais eficiente computacionalmente. Esta é definida como a parte positiva de seu argumento, isto é,

a função zera todos os valores negativos de saída da camada anterior.

Ao final da rede é colocada uma camada totalmente conectada (*fully connected*), na qual a entrada é a saída da camada anterior e todas as unidades de entrada possuem um peso separável para cada unidade de saída. Essa camada é responsável por classificar a imagem da entrada em uma classe previamente fornecida. A saída da camada totalmente conectada são N neurônios, com N sendo a quantidade de classes do modelo.

Há inúmeras aplicações com redes convolucionais desde o início dos anos 1990, quando começaram a ser desenvolvidas redes neurais de atraso de tempo para reconhecimento de voz e leitura de documentos. Neste projeto, o foco do problema está no reconhecimento de fala que será abordado na sequência.

4.2 RECONHECIMENTO DE FALA

O reconhecimento de fala é uma das aplicações mais complexas existentes de inteligência artificial e ainda possui muito campo para avançar. Esta tecnologia é basicamente composta por três aplicações bases que são modelo acústico, modelo fonético e modelo de linguagem.

O modelo acústico é uma junção de todas as palavras e todos os áudios que estão disponíveis para a construção de um modelo. Com transcrição manual e uma base de áudios de diversas pessoas e sotaques é criado um modelo inicial acústico. Para isto, são necessárias aproximadamente cem mil horas de gravações (incluindo as traduções da linguagem) a fim de obter os primeiros resultados.

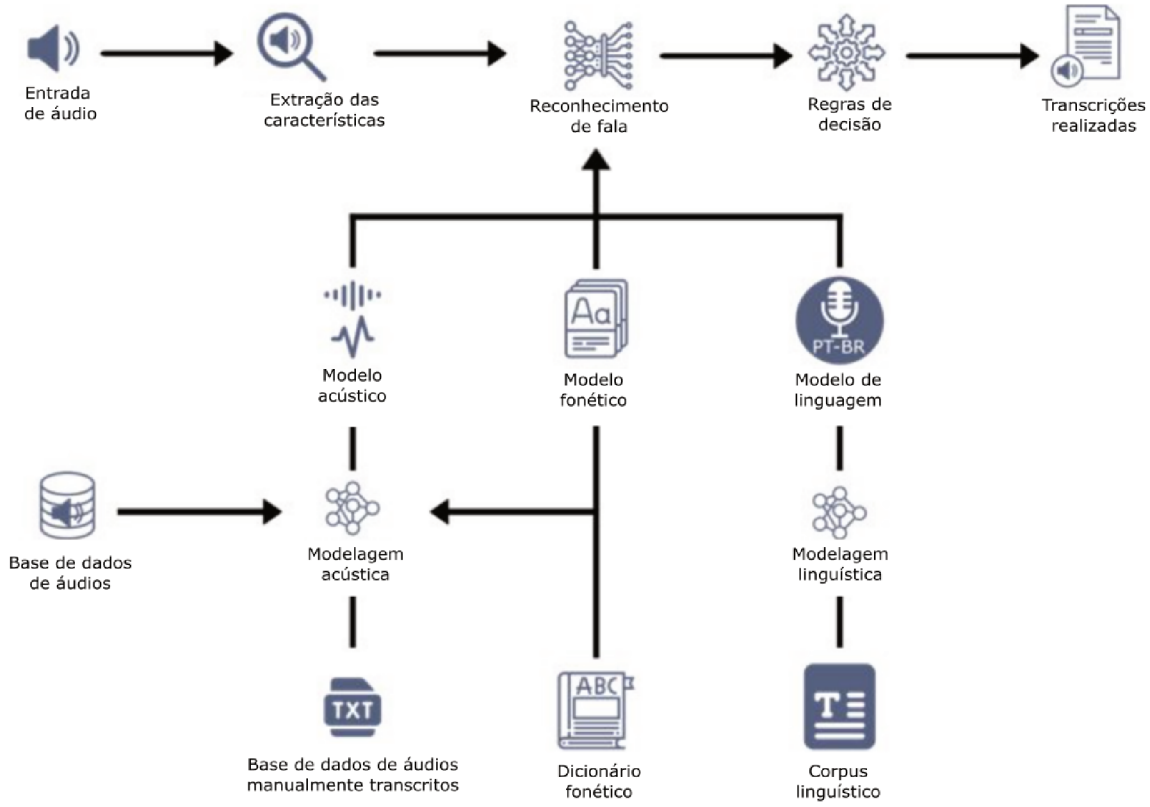
O modelo fonético é basicamente a forma como o ser humano escuta, é o fonema da palavra. Este modelo funciona como uma referência para o computador, permitindo que ele possa entender as palavras. Resumindo, ao falar do modelo fonético são as palavras como elas de fato soam.

Além disso, o reconhecimento de fala conta também com o corpus linguístico. O corpus linguístico é o conjunto de regras da língua portuguesa, no caso do Brasil, incluindo as composições de frases.

Ao falar de um projeto de reconhecimento de fala, uma das primeiras coisas a se pensar é o tratamento do som. Existe uma etapa chamada de análise exploratória de um arquivo de áudio. A forma como o áudio é captado influencia em como esse áudio será enviado para a aplicação de inteligência artificial. Dentre algumas informações mais relevantes do tratamento do áudio, o primeiro passo é fazer a análise deste áudio, identificar suas características e possíveis tratamentos.

Na análise do áudio, é preciso ter atenção ao tipo de formato do arquivo (WAV, MP3, etc.). Existem alguns formatos que são melhores do que os outros. Ao trabalhar com reconhecimento de fala, o formato MP3 é julgado ruim, pois ele pode destruir algumas características do áudio. O formato FLAQ é um dos melhores formatos para

Figura 4 – Passos para o reconhecimento de fala.



Fonte – Arquivo da Universidade Radix.

trabalhar, porém é difícil de encontrá-lo. Um formato razoavelmente bom para trabalhar e fácil de encontrar é o WAV. Ele tem uma qualidade um pouco melhor para usar em trabalhos de reconhecimento de fala. O codec do áudio que é a maneira como é feita a compressão do arquivo de áudio e pode influenciar na performance da aplicação de inteligência artificial.

Outras situações passíveis de atenção são verificar se o áudio é mono ou estéreo e a quantidade de canais que esse áudio possui (ou seja, a quantidade de locutores que falam no áudio). É necessário que seja possível identificar quais são os canais existentes e o que cada um está falando.

Além desses passos, é importante verificar e procurar corrigir as imperfeições no áudio, como redução de ruídos externos. A voz humana tem uma frequência específica. Existem ruídos que podem ser baixos ou ruídos mais altos. É possível passar filtros altos e baixos pra fazer a redução destes ruídos.

Após a transcrição do áudio ser realizada, é preciso avaliar sua performance. Em um projeto de reconhecimento de fala normalmente são utilizadas ferramentas cognitivas ou empresas que trabalham com essa parte de cognição. Isto por que são necessárias aproximadamente cem mil horas faladas que acarretariam em um trabalho

muito grande. As plataformas cognitivas (AWS, Google, Microsoft, IBM, entre outras como Nuance e Varrent) já possuem esta base. Desta forma, a boa prática nestes projetos é utilizar uma ferramenta de mercado para fazer essa transcrição. No entanto, para casos específicos - como um dicionário médico com informações técnicas, jargões, termos e nomes - o vocabulário personalizado precisa ser adicionado à plataforma de alguma maneira.

Então, para realizar a melhora da qualidade da transcrição, é possível melhorar o áudio. Porém, uma boa recomendação é a criação de um vocabulário personalizado. Um ponto que é totalmente diferencial é a utilização do dicionário fonético. Existem algumas plataformas cognitivas que permitem a inserção da pronúncia correta da palavra. Em resumo, quanto melhor a qualidade do áudio, melhor a performance do reconhecimento de fala. Mas também existem outras possibilidades que fazem com que aumente a performance da aplicação.

Após melhorar a qualidade do áudio, criar o vocabulário personalizado e dicionário fonético, é preciso realizar a avaliação da transcrição. Para isto, existe uma métrica chamada WER (*Word Error Rate* ou Taxa de Erros da Palavra). WER é uma métrica comumente usada para avaliar o desempenho de sistemas de reconhecimento de fala, ela mede a distância do que foi transcrito com o que de fato era para ser realizado. Em outras palavras, é a distância de edição entre a sequência de palavras de referência e sua transcrição automática pelo sistema.

A fórmula para calcular a taxa WER pode ser averiguada na equação

$$WER = 100\% \times \frac{S + D + I}{\left|W\right|}, \tag{1}$$

onde W é o total de palavras dentro da frase. A sequência das palavras reconhecidas pode conter três tipos de erros. O primeiro tipo é conhecido como substituição (S) e acontece quando uma palavra incorreta é reconhecida no lugar de uma palavra falada corretamente. O segundo erro é a exclusão (D), que ocorre quando uma palavra falada não é reconhecida. Por fim, o terceiro erro acontece quando palavras extras são estimadas pelo reconhecedor e chama-se inserção (I).

Dessa maneira, obtém-se uma espécie de acuracidade das palavras. Para sistemas de reconhecimento simples, como sistemas com palavras isoladas, o desempenho é simplesmente a porcentagem de palavras não reconhecidas.

Para realizar a comparação descrita anteriormente, é necessário ter tanto o áudio quanto a tradução verdadeira, ou seja, é preciso obter a verdade fundamental (em inglês, *ground truth*). Ao realizar essa comparação é possível ter um entendimento de quais palavras foram excluídas, substituídas e inseridas.

Ao finalizar a avaliação da transcrição do áudio, encaminha-se para a etapa de colocação de um pipeline de qualidade. Pipeline é uma sequência de passos, algumas melhorias que podem ser colocadas em toda a operação a fim de facilitar um trabalho

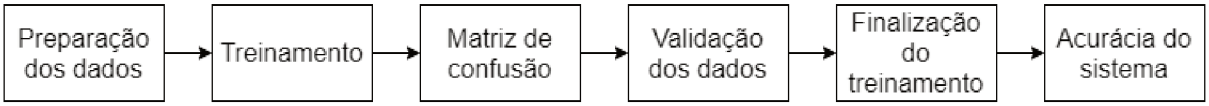
que anteriormente era muito complexo de ser realizado. Existem pipelines técnicos e pipelines de modelos de negócio. O pipeline técnico seria a possibilidade de melhoria da transcrição ou a identificação de ruídos, por exemplo. Falando do pipeline de modelo de negócio, poderia ser a identificação de melhorias no processo como um todo.

Assim, olhar a qualidade do que está sendo feito após a realização e avaliação das métricas é uma operação muito importante que pode agregar um valor muito grande para o negócio.

5 RECONHECIMENTO DE FALA

Esta seção mostrará a construção de um exemplo básico de reconhecimento de fala que identifica dez palavras diferentes. É importante ressaltar que os sistemas reais de reconhecimento de voz e áudio são muito mais complexos, porém este exemplo fornece uma compreensão básica das técnicas envolvidas. O modelo obtido tenta classificar um áudio de um segundo como silêncio, uma palavra desconhecida ou as palavras em inglês "yes", "no", "up", "down", "left", "right", "on", "off", "stop"e "go". Este reconhecimento de fala seguirá a sequência de passos demonstrada na Figura 5.

Figura 5 – Sequência do exemplo de reconhecimento de fala.



5.1 TREINAMENTO DOS DADOS

A fim de realizar o exemplo, é necessário ter instalado na máquina uma plataforma de aprendizado de máquina com ferramentas, bibliotecas e recursos - o TensorFlow. O treinamento do modelo pode levar várias horas e utiliza bastante memória da máquina.

Para iniciar o processo, deve-se baixar o conjunto de dados chamado *Speech Commands*, que consiste em aproximadamente cento e cinco mil arquivos de áudio no formato WAV. Esses áudios são de pessoas dizendo trinta palavras diferentes e foram coletados pelo Google.

Este conjunto de dados de áudio de palavras faladas foi projetado para ajudar a treinar e avaliar sistemas de reconhecimento de palavras-chave. Essa tarefa é um desafio interessante por que requer um conjunto de dados especializado, diferente dos convencionais que são utilizados para reconhecimento automático de fala de frases completas. É possível também incluir arquivos gravados na própria voz para agregar ao treinamento.

Ao finalizar a transferência do arquivo, inicia-se o laço de treinamento. A cada etapa são mostradas as respectivas informações: número da etapa, taxa de aprendizagem, acurácia e entropia cruzada. Essas informações podem ser visualizadas na Figura 6.

O número da etapa mostra em qual etapa do ciclo o treinamento se encontra. São dezoito mil etapas no total. A taxa de aprendizagem controla a velocidade das atualizações de peso da rede.

Figura 6 – Informações da etapa de treinamento.

```
INFO:tensorflow:Saving to "/tmp/speech_commands_train/conv.ckpt-17900"
I1122 08:41:34.191745 139749286704960 train.py:296] Saving to "/tmp/speech_commands_train/conv.ckpt-17900"
INFO:tensorflow:Step #18000: rate 0.000100, accuracy 83.0%, cross entropy 0.504188
I1122 08:44:28.302851 139749286704960 train.py:260] Step #18000: rate 0.000100, accuracy 83.0%, cross entropy 0.504188
```

O valor de acurácia representa quantas classes foram previstas corretamente na respectiva etapa de treinamento. Esse valor costuma flutuar muito, mas tende a aumentar em média conforme o treinamento avança. O modelo produz uma matriz de números, um para cada rótulo, e cada número é a probabilidade prevista de a entrada ser essa classe. O rótulo previsto é adotado através da escolha da entrada com a pontuação mais alta. As pontuações estão sempre entre zero e um, com valores mais altos representando mais confiança no resultado.

A entropia cruzada é o resultado da função de perda que está sendo usado para guiar o processo de treinamento. Esta é uma pontuação obtida comparando-se o vetor de pontuações do treinamento atual com os rótulos corretos, e isso tende para baixo ao longo do treinamento.

A cada quatrocentas etapas é registrada uma matriz de confusão, que é uma tabela de visualização do desempenho do algoritmo. A matriz de confusão final do treinamento completo pode ser verificada na Figura 7.

Figura 7 – Matriz de confusão final do treinamento.

```
W1122 08:45:33.523953 139749286704960 train.py:320] Confusion Matrix:
 [[404   0   0   0   2   0   1   0   0   0   1   0]
 [  4 278  10   9  15  18  17  15  19   4   8  11]
 [  0   3 395   5   0   2  12   0   0   0   2   0]
 [  1  10   5 348   0  17  10   2   0   0   0  12]
 [  1   4   1   0 388   2   2   0   9   5  10   3]
 [  1   9   5  29   2 348   4   0   2   0   0   6]
 [  1   2  12   1   3   1 386   4   0   0   1   1]
 [  2  13   1   1   0   0  10 365   2   2   0   0]
 [  1  10   1   0   7   5   1   0 354  15   1   1]
 [  1   3   0   0  35   1   5   1  14 331   6   5]
 [  2   0   0   0  10   3   0   0   0   2 391   3]
 [  3  17   3  67   5  13   7   3   0   1   1 282]]
WARNING:tensorflow:Final test accuracy = 87.3% (N=4890)
W1122 08:45:33.524117 139749286704960 train.py:321] Final test accuracy = 87.3% (N=4890)
```

Cada coluna da matriz representa um conjunto de amostras que foram previstas para cada um dos rótulos ("silence", "unknown", "yes", "no", "up", "down", "left", "right", "on", "off", "stop"e "go"). Então a primeira coluna representa todos os áudios que foram previstos como silêncio ("silence"), a segunda coluna é de todos que foram previstos como desconhecido ("unknown") e assim por diante.

Cada linha da matriz representa os áudios por seus rótulos corretos e verdadeiros. A primeira linha contém todos os áudios que eram silêncio, a segunda é de todos que eram desconhecidos, etc.

Essa matriz pode ter mais utilidade do que apenas uma pontuação de precisão, por que ela fornece um bom resumo de quais erros a rede está cometendo. Um modelo perfeito produziria uma matriz de confusão em que todas as entradas fossem zero, exceto pela diagonal principal. Identificar desvios desse padrão pode auxiliar a descobrir como o modelo é facilmente confundido e, dessa forma, é possível resolver os problemas encontrados adicionando mais dados ou limpando categorias.

5.2 VALIDAÇÃO DOS DADOS

Após a matriz de confusão, o sistema mostra uma linha contendo informações do número da etapa e da acurácia de validação.

Uma boa prática é separar o conjunto de dados em três categorias. O maior conjunto de ter cerca de 80% dos dados e é utilizado para treinar a rede. Um conjunto menor, que pode obter cerca de 10% dos dados, é conhecido como conjunto de validação. Ele é reservado para avaliação da acurácia durante o treinamento. Por último, outro conjunto com cerca de 10% dos dados é o conjunto de teste. Este é usado para avaliar a acurácia após a conclusão do treinamento.

Essa divisão é realizada para evitar que a máquina memorize as entradas durante o treinamento. Ao manter o conjunto de validação separado, é possível garantir que o modelo funcione com dados não visto anteriormente. O conjunto de teste é uma proteção adicional para garantir que o modelo não esteja se ajustando para funcionar apenas com os conjuntos de treinamento e validação.

O roteiro de treinamento separa automaticamente o conjunto de dados nestas três categorias e o valor da acurácia mostrado pelo sistema é referente ao conjunto de validação.

5.3 RESULTADOS

Após algumas horas de treinamento, o roteiro deve ter concluído todas as dezoito mil etapas. O sistema imprime uma matriz de confusão final, junto com uma pontuação de acurácia, ambos executados no conjunto de teste explicado anteriormente.

Também fica acessível uma plataforma, chamada Tensorboard, na qual pode-se visualizar o processo de treinamento através de gráficos das grandezas exploradas. A plataforma permite ainda a utilização de filtros, alteração da grandeza do eixo horizontal, entre outras configurações.

Nas Figuras 8 e 9, são mostrados os gráficos da evolução da acurácia e da entropia cruzada pelo número de etapas do treinamento, respectivamente. Nestes gráficos, a linha azul representa os dados de treino enquanto a linha vermelha se refere aos dados de validação.

Figura 8 – Gráfico da acurácia do treinamento: Iterações X Acurácia. A linha azul representa dados de treino e a linha vermelha representa dados de validação.

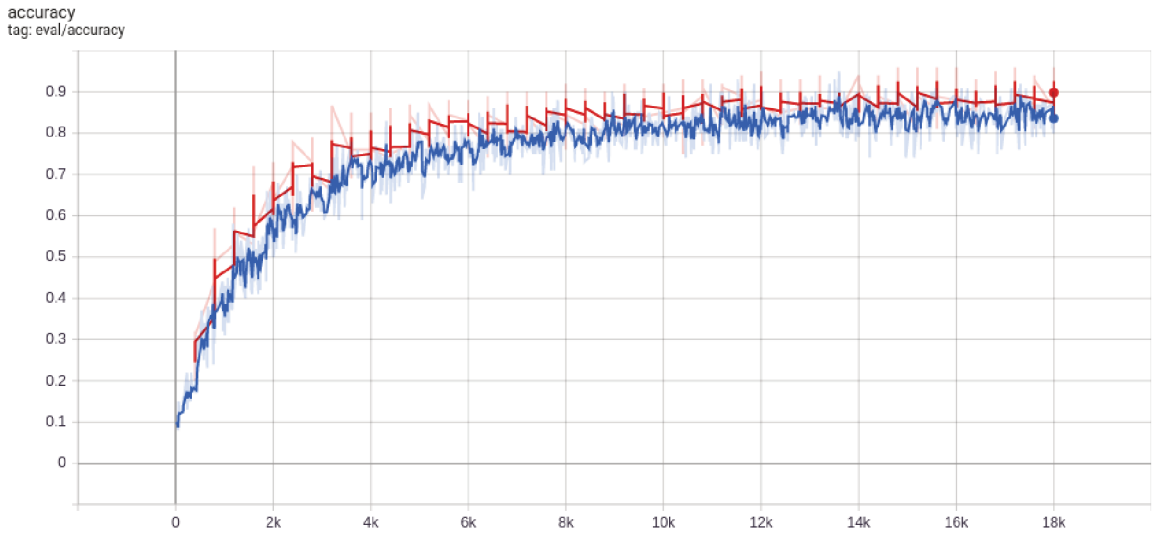
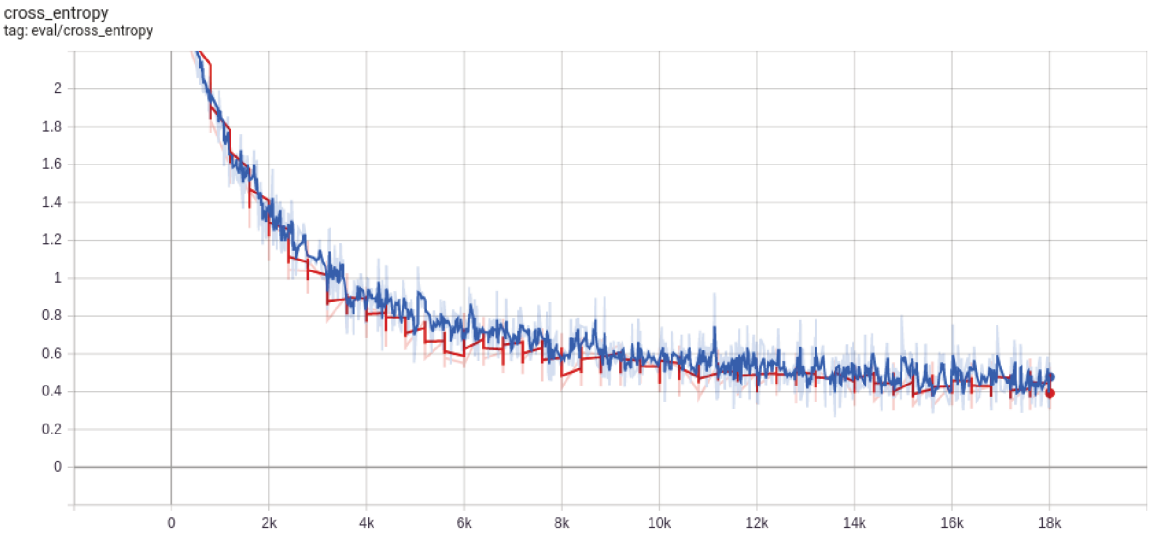


Figura 9 – Gráfico da entropia cruzada do treinamento: Iterações X Entropia. A linha azul representa dados de treino e a linha vermelha representa dados de validação.



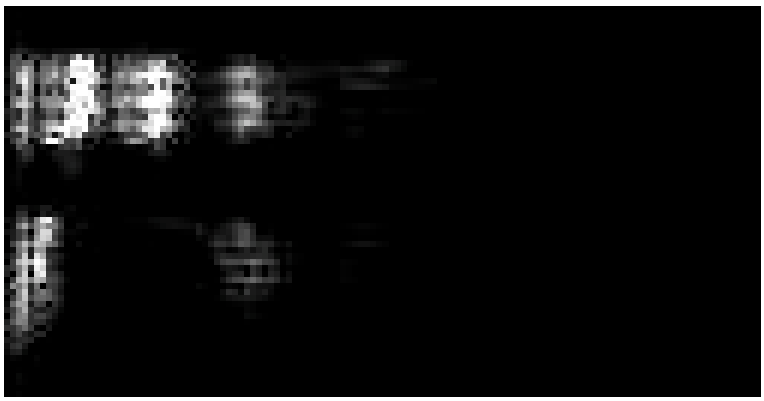
5.4 ARQUITETURA DO SISTEMA

A arquitetura utilizada neste treinamento é relativamente simples, rápida de treinar e fácil de entender. Existem muitas abordagens diferentes para construir modelos de redes neurais para trabalhar com áudio, o modelo utilizado aqui foi baseado no tipo de rede convolucional.

Para construir o modelo, foi definida uma janela de tempo em que acredita-se que as palavras faladas devem se encaixar. E o sinal de áudio é convertido nessa janela de tempo em uma imagem. Isso é feito agrupando as amostras de áudio em segmentos curtos, com apenas alguns milissegundos de duração. Assim, calcula-se a intensidade das frequências em um conjunto de bandas. Cada conjunto de intensidades de frequência é tratado como um vetor de números e esses vetores são organizados em ordem de tempo para formar uma matriz bidimensional.

Enfim, essa matriz de valores pode ser tratada como uma imagem de canal único e é conhecida como espectrograma. Espectrograma é uma representação visual do espectro de frequências de um sinal que varia com o tempo. Quando aplicado a um sinal de áudio, o espectrograma é chamado de sonógrafo, impressão de voz ou simplesmente voz. O tipo de imagem que uma amostra de áudio produz pode ser visualizado na Figura 10. Devido à ordem de memória do TensorFlow, o tempo na figura está aumentando de cima para baixo, com frequências indo da esquerda para a direita.

Figura 10 – Espectrograma de um sinal de áudio.



Fonte – Repositório do TensorFlow.

Como o ouvido humano é mais sensível a algumas frequências do que outras, é comum no reconhecimento de fala fazer um processamento adicional a essa representação para transformá-la em um conjunto de Coeficientes Cepstrais de Mel-Frequência (MFCCs). Esta também é uma representação bidimensional de um canal, portanto, pode ser tratada como uma imagem.

A imagem produzida por essas etapas de processamento é então alimentada em uma rede neural convolucional multicamadas, com uma camada totalmente conectada seguida por uma função softmax no final. A função softmax ou função exponencial normalizada, é uma generalização da função logística para várias dimensões.

A maioria das aplicações de reconhecimento de fala precisa ser executada em um fluxo contínuo de áudio, ao invés de arquivos individuais. Uma maneira típica de

usar um modelo neste ambiente é aplicá-lo repetidamente em diferentes deslocamentos no tempo e calcular a média dos resultados em uma janela curta para produzir uma previsão suavizada. Comparando a entrada com uma imagem, ela estaria rolando continuamente ao longo do eixo do tempo. As palavras que devem ser reconhecidas podem começar a qualquer momento, portanto, é preciso pegar uma série de intervalos para ter a chance de ter um alinhamento que capture a maior parte do enunciado na janela de tempo do modelo. Se a amostragem for feita em uma taxa alta o suficiente, a chance de capturar a palavra em várias janelas será muito boa, portanto, a média dos resultados melhora a confiança geral da previsão.

6 GERAÇÃO DE LAUDOS MÉDICOS POR RECONHECIMENTO DE FALA

O projeto da geração de laudos médicos por reconhecimento de fala foi realizado a fim de atender a demanda de um grupo de medicina e saúde, que identificou a necessidade de criar uma inteligência de negócio capaz de identificar o dicionário médico na hora da criação de laudos médicos por seus profissionais.

Este projeto visou obter como produto uma aplicação de inteligência artificial com foco em reconhecimento de fala particular, para que a própria equipe do grupo fosse capaz de fazer o treinamento adequado de modelo de acordo com o dicionário médico, com saídas via API para integração com aplicações para internet e para computador além de aplicativo para celular. A possibilidade de fazer o reconhecimento de fala com os próprios celulares dos médicos e criar uma arquitetura separada que possa se interconectar com todas as outras aplicações (via API, por exemplo) foi vista como maior ganho de valor pela empresa.

O foco do desenvolvimento foi uma aplicação inicial de inteligência artificial com um modelo pequeno de treinamento e pronto para integração via API mostrando caminhos de como a empresa poderia fazer o treinamento personalizado que desejava.

As tarefas realizadas seguiram aproximadamente o roteiro descrito a seguir. Inicialmente, para a criação da arquitetura de inteligência artificial em nuvem foi necessário realizar uma pesquisa dos serviços cognitivos disponibilizados pelo mercado para escolher qual seria o mais adequado para hospedar o projeto. Após a decisão do serviço, prosseguiu-se buscando um melhor entendimento da arquitetura de nuvem para IA na plataforma escolhida.

O passo seguinte foi realizar a criação da arquitetura do modelo de treinamento. Dessa forma, foi preciso criar um ambiente de crescimento de palavras médicas, onde seria possível adicionar novas palavras para entendimento, selecionar palavras que não foram entendidas, poder identificar o áudio ou ainda a possibilidade do usuário identificar que não foi corretamente enviado.

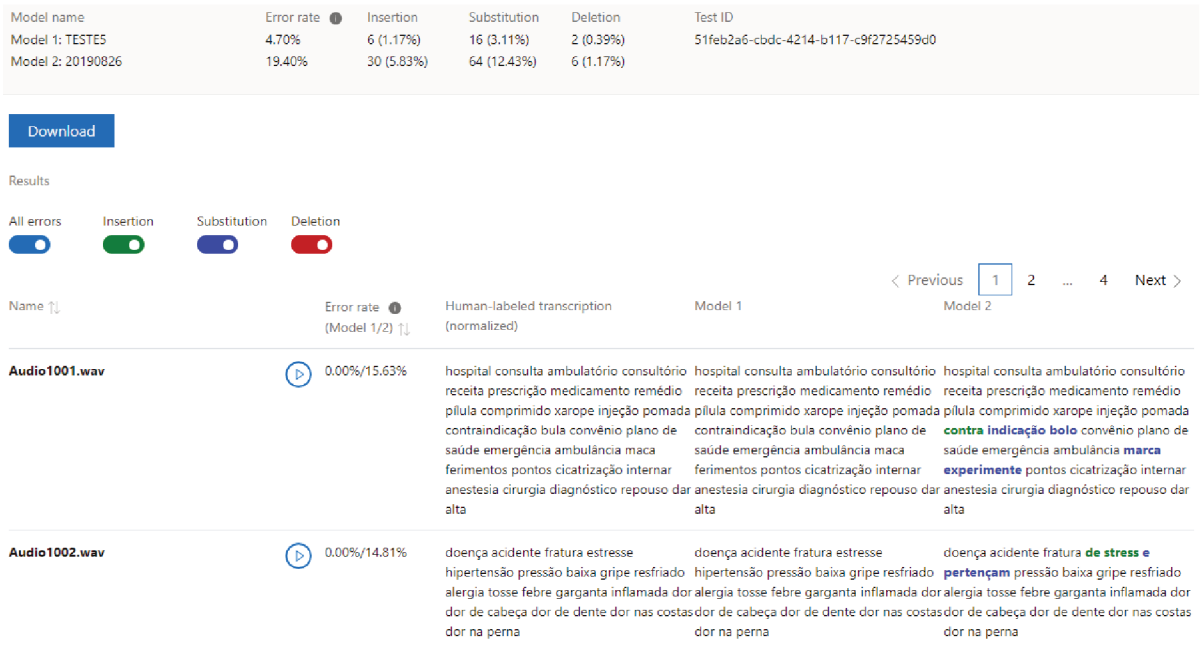
Com estas etapas concluídas, deu-se início ao treinamento do modelo com palavras do cenário médico. A partir de então, o prosseguimento do projeto incluiu as seguintes atividades: treinar o modelo, recuperar textos e palavras dentro da arquitetura de IA, criar início de mineração de texto, apresentar a conversão de fala para texto sendo captado via celular e via página na internet.

Esta última etapa de apresentação da ferramenta de fala para texto também abrangeu a criação de uma arquitetura para aplicativo de celular e para aplicação na internet e habilitação de um campo para converter fala para texto, além da conexão com a API da solução.

6.1 ARQUITETURA DO PROJETO

Após a definição de utilizar os serviços cognitivos da Microsoft, foi criada a primeira arquitetura do projeto. Inicialmente, foi feito o passo a passo para personalizar a base de treinamento para o dicionário médico e, com isto, foi possível realizar o treinamento do primeiro modelo de teste e subir para produção para utilização deste via API. Um exemplo da página de teste pode ser visualizado na Figura 11.

Figura 11 – Página de treinamento utilizando os serviços cognitivos da Microsoft Azure.



Fonte – Radix.

Estes primeiros testes foram realizados de maneira bastante manual pela estudante, através da leitura de textos característicos do meio médico encontrados em sites na internet, gravando o som pelo microfone do próprio celular.

Paralelamente, iniciou-se o desenvolvimento do aplicativo para possibilitar a demonstração da funcionalidade do laudo médico. Assim como a automatização do conjunto de dados e do treinamento.

O passo a passo da arquitetura do projeto foi idealizado e consolidado pelo gestor do projeto. No entanto, todas as etapas - antes e depois da idealização - foram discutidas com a equipe. Da mesma maneira, as atividades que foram realizadas tiveram participação total ou parcial da estudante.

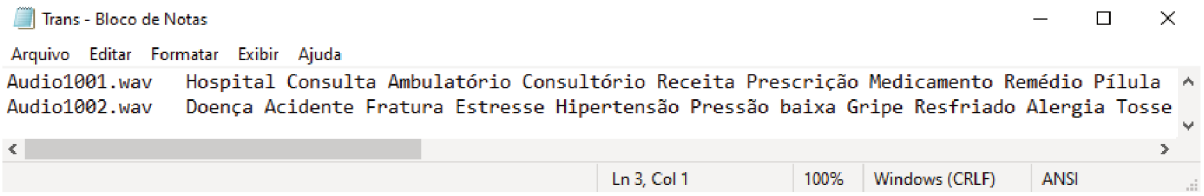
6.2 DESENVOLVIMENTO DO PROJETO

Este sistema foi desenvolvido visando a possibilidade de a empresa ter uma equipe que possa criar e crescer o conjunto de dados de treinamento do dicionário médico, totalmente integrado com os serviços cognitivos da Microsoft. Desta forma, a fim de integrar o sistema com a Microsoft, foi criada uma estrutura em Python, utilizando o framework Flask. Nesta estrutura existe a funcionalidade de autenticação de usuários, além das funcionalidades de gestão do conjunto de dados com a possibilidade de realizar o envio dos áudios e das suas respectivas transcrições.

Dentro da automatização de todo o processo, um dos passos necessários foi a criação do conjunto de dados. Esta foi realizada através de um código em Python para realizar as seguintes ações: selecionar os áudios de determinada pasta do computador, associar os áudios com suas respectivas transcrições, converter as entradas de áudio para o formato necessário, criar arquivo de texto com áudios e transcrições, criar arquivo do tipo ZIP dos áudios e do arquivo de texto, conectar com a API do AzureDevOps e realizar o envio do arquivo ZIP utilizando o método POST.

Conforme mencionado anteriormente, os áudios deveriam ser descritos em um arquivo a ser criado no código Python com o nome "Trans.txt", no qual estaria a cada linha o nome do áudio seguido da sua transcrição, conforme mostra a Figura 12. Após este passo, todos os arquivos de áudio juntamente com o arquivo de texto deveriam ser enviados para um arquivo do tipo ZIP para dar prosseguimento ao processo.

Figura 12 – Exemplo de arquivo de texto criado para utilização da ferramenta cognitiva.



A conversão do áudio também mencionada anteriormente precisava estar de acordo com as configurações solicitadas pela ferramenta cognitiva da Microsoft. A entrada do áudio poderia admitir diferentes formatos (MP4, MP3, WAV, entre outros) e a saída deveria ser apenas WAV, além de atender as configurações mostradas na Figura 13.

Os trechos do código responsável por realizar essas ações pode ser visualizados a seguir. Na Figura 14 pode ser conferida a função de conversão do áudio. Já a Figura 15 possui o trecho do código que de fato chama a função de conversão do áudio de todos os arquivos contidos em uma pasta específica, juntamente com suas transcrições, dentro da função de inclusão de novos áudios na aplicação.

Figura 13 – Configurações de áudio para utilização da ferramenta cognitiva.

Propriedade	**Value**
Formato de arquivo	RIFF (WAV)
Taxa de amostragem	8.000 Hz ou 16.000 Hz
Canais	1 (mono)
Comprimento máximo por áudio	2 horas
Formato de exemplo	PCM, 16 bits
Formato de arquivo	.zip
Tamanho máximo de arquivo	2 GB

Fonte – Radix.

Figura 14 – Trecho do código da função de conversão do áudio.

```
def converter_audio(audio_path):
    aux = audio_path.split('.')
    format = aux[(len(aux)-1)]
    audio = AudioSegment.from_file(audio_path, format=format)

    sound_mono = audio.set_channels(1)
    sound_mono_8000 = sound_mono.set_frame_rate(8000)

    wav_file = str(int(time.time())) + ".wav"

    return wav_file
```

Fonte – Radix.

Também foi projetado o caso para quando já houver um conjunto de dados e for necessário inserir outro conjunto sem perder o existente. O código para este caso pode ser visualizado na Figura 16.

Com a parte do gerenciamento das atividades encaminhada, iniciou-se a criação da estrutura da página em Flask, a qual contou com etapas básicas de autenticação de usuário e página para a gestão dos dados. Para isso, foi realizada uma pesquisa sobre as boas práticas de como utilizar as estruturas de *back-end* e de *front-end* dentro do Flask.

Com a estrutura básica da página criada, os passos seguintes foram criar etapa de conexão do usuário contendo um formulário com as entradas de informações de nome e senha do usuário, criar um menu lateral para navegação entre as páginas, criar

Figura 15 – Trecho do código de preparação para a utilização da ferramenta cognitiva.

```
@app.route('/audio', methods=["GET", "POST"])
def create():
    if request.method == 'POST':
        data = request.values.to_dict()

        if data.get('file', None) is not '' and data.get('transcription', None) is not '':
            filepath = os.path.join(
                os.path.dirname(os.path.abspath(__file__))+"\\static\\audiosaconverter",
                request.files['file'].filename)
            request.files['file'].save(filepath)
            audio_novo = converter_audio(filepath)
            filepath2 = os.path.join(
                os.path.dirname(os.path.abspath(__file__))+"\\static\\audios",
                audio_novo)
            request.files['file'].save(filepath2)
            new = {'file': filepath2, 'transcription': data.get('transcription')}
            db.audios.insert_one(new)
            flash("Transcrição criada com sucesso!", "success")

            return redirect('/audios')

        else:
            flash("Parâmetros incorretos!", "error")
            return redirect('/audio')

    if request.method == "GET":
        return render_template('create.html'), 200
```

Fonte – Radix.

página e CRUD (*Create, Read, Update, Delete*) de gestão do conjunto de áudios.

A Figura 17 mostra o trecho do código que realiza a autenticação com o serviço cognitivo da Microsoft Azure, a fim de que permita à aplicação enviar as solicitações e receber as respostas do servidor.

O trecho de código que contém as informações para criação da página de entrada na aplicação pode ser conferido na Figura 18. Este tipo de autenticação é bastante simples e foi implementado apenas para fins de demonstração para o cliente.

Já o código responsável pela etapa de gestão do conjunto de áudios, como atualizar ou excluir arquivos, está apresentado na Figura 19.

Após a estruturação do sistema encaminhada, foram acrescentadas algumas funcionalidades como ferramentas de edição do texto com a criação de atalhos de fala para iniciar e encerrar a função de negrito e itálico. Ou atalhos para auto textos, ou seja, textos previamente formatados que serão incluídos no corpo do texto ao chamar o atalho. Também foi adicionado um botão para exportar o texto traduzido da fala para um arquivo do tipo RTF padronizado em formato de laudo médico.

Figura 16 – Trecho do código de inserção de um novo conjunto de dados.

```
@app.route('/datasets', methods=['POST'])
def datasets():
    if request.method == 'POST':
        with zipfile.ZipFile(request.files['zip'], 'r') as zip_file:
            zip_file.extractall(os.path.join(os.path.dirname(os.path.abspath(__file__))+"\\static\\zip"))

        audios = os.listdir(os.path.join(os.path.dirname(os.path.abspath(__file__))+"\\static\\zip\\audios"))

        import codecs
        with open(os.path.join(os.path.dirname(
            os.path.abspath(__file__))+"\\static\\zip\\Trans.txt"), 'r+', encoding='utf-8') as f:
            arquivo = f.read()
            print(arquivo)
            for audio in audios:
                conv_audio = salvar_audio(os.path.join(os.path.dirname(
                    os.path.abspath(__file__))+"\\static\\zip\\WAV\\audios", audio), infos)
                print(conv_audio)
                arquivo = re.sub(audio, conv_audio, arquivo)
            print(arquivo)
            f.seek(0)
            f.write(arquivo)
        print('hi')
        teste_fleury = zipfile.ZipFile(os.path.join(os.path.dirname(
            os.path.abspath(__file__))+"\\static\\datasets", 'dataset.zip'), 'w')

        for folder, subfolders, files in os.walk(os.path.dirname(os.path.abspath(__file__))):

            for file in files:
                if file.endswith('.wav') or file.endswith('.txt'):
                    teste_fleury.write(os.path.join(folder, file), os.path.relpath(
                        os.path.join(folder,file), os.path.dirname(os.path.abspath(__file__))),
                        compress_type = zipfile.ZIP_DEFLATED)

        teste_fleury.close()
```

Fonte – Radix.

Figura 17 – Trecho do código para autenticação com o serviço cognitivo da Microsoft Azure.

```
@app.route('/azureModel')
def azure():
    url = server+'.cris.ai/api/speechtotext/v2.0/datasets'
    auth = {'Ocp-Apim-Subscription-Key': key1}
    r = requests.get(url, headers=auth)
    resposta = r.json()

    for data in resposta:
        print(data)

    return "teste"
```

Fonte – Radix.

Figura 18 – Trecho do código da página de autenticação de usuário.

```
@app.route('/contatos')
def contat():
    return render_template('contatos.html')

@app.route('/sobre')
def sobre():
    return render_template('sobre.html')

@app.route('/login')
def login():
    proxima = request.args.get('proxima')
    return render_template('index.html', proxima=proxima)

@app.route('/autenticar', methods=['POST',])
def autenticar():
    if request.form['usuario'] in usuarios:
        usuario = usuarios[request.form['usuario']]
        if usuario.senha == request.form['senha']:
            session['usuario_logado'] = usuario.id
            proxima_pagina = request.form['proxima']
            return redirect(proxima_pagina)
    else :
        return redirect(url_for('login'))

@app.route('/logout')
def logout ():
    session['usuario_logado'] = None
    return redirect(url_for('login'))
```

Fonte – Radix.

Figura 19 – Trecho do código de gestão do conjunto de áudios.

```
@app.route('/audio/<id>', methods=["GET", "POST"])
def audio(id):
    if request.method == 'GET':
        data = db.audios.find_one({'_id': ObjectId(id)})
        return render_template('update.html', data = data)

    data = request.values.to_dict()
    if request.method == 'POST':
        if id != None:
            db.audios.update_one(
                {'_id': ObjectId(id)}, {'$set': data})
            flash("Transcrição atualizada com sucesso!", "success")
            return redirect('/audios')
        else:
            flash("Algo deu errado!", "error")
            return redirect('/audio/'+id)

@app.route('/audio/excluir/<id>', methods=["GET"])
def excluir(id):
    if id != None:
        db_response = db.audios.delete_one({'_id': ObjectId(id)})
        if db_response.deleted_count == 1:
            flash("Deletado com sucesso!", "success")
            return redirect("/audios")
        else:
            flash("Não foi possível concluir a deleção", "error")
            return redirect('/audios')
    else:
        return redirect('/audios')
```

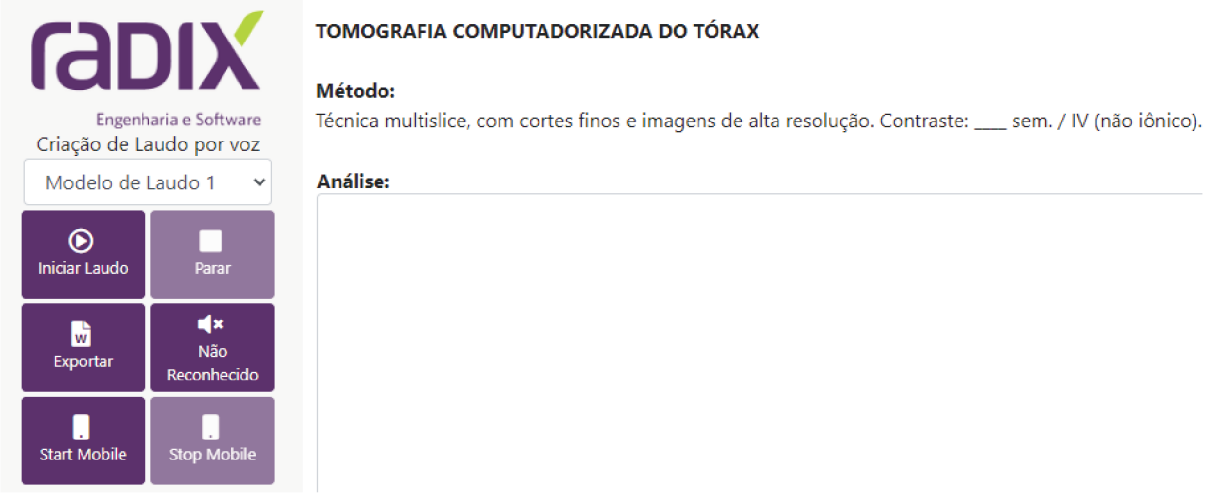
Fonte – Radix.

7 RESULTADOS

O projeto realizado conta com um treinamento que possui vocabulário personalizado de termos e jargões médicos e está hospedado na Microsoft. O modelo da Microsoft possui a possibilidade de utilizar tanto dicionário fonético quanto vocabulário personalizado. Todo o processo feito na plataforma pode ser feito com Python para automatizar toda a transcrição.

Quando utilizada em um ambiente controlado, sem ruídos ou ecos, a ferramenta consegue captar quase tudo que é falado de forma correta. Sendo um produto que faz o reconhecimento de fala de maneira eficiente, a ideia do projeto é dar produtividade aos médicos que fazem os laudos. O sistema final pode ser visualizado na Figura 20.

Figura 20 – Página do sistema desenvolvido para geração de laudo médico com reconhecimento de fala.



Fonte – Radix.

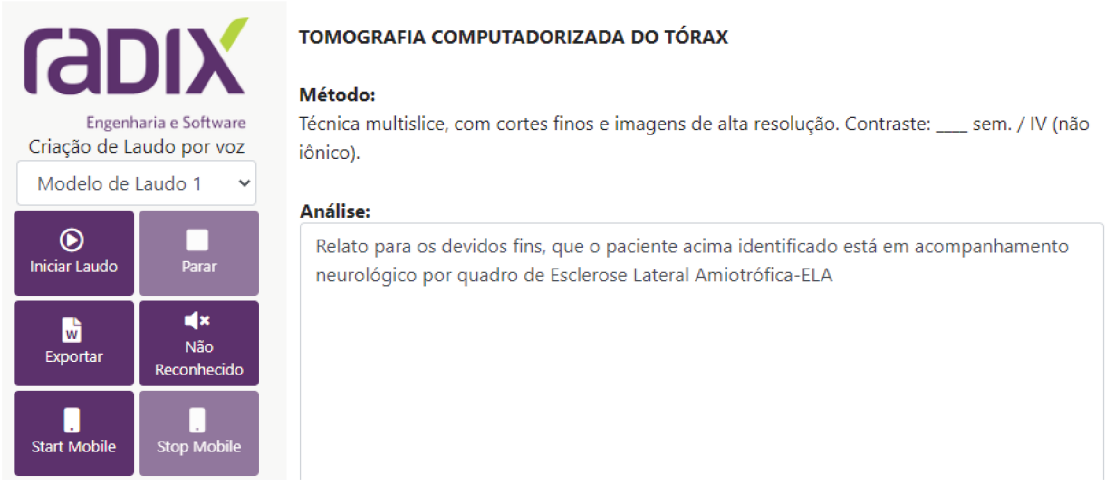
Além de realizar a transcrição e de conseguir fazer algumas ações de formatação no texto, a ferramenta é capaz de adicionar outros textos e de reconhecer palavras-chave. Como exemplo, podem ser citadas as ações:

- Iniciar negrito;
- Finalizar negrito;
- Adicionar texto 1;

A resposta do sistema ao reconhecer a frase "adicionar texto 1"é imprimir na tela o texto referente a este atalho, como pode ser visto na Figura 21

Também há a possibilidade de aumentar o crescimento orgânico dessa ferramenta, então seria possível clicar na palavra que não foi reconhecida, ir na opção não

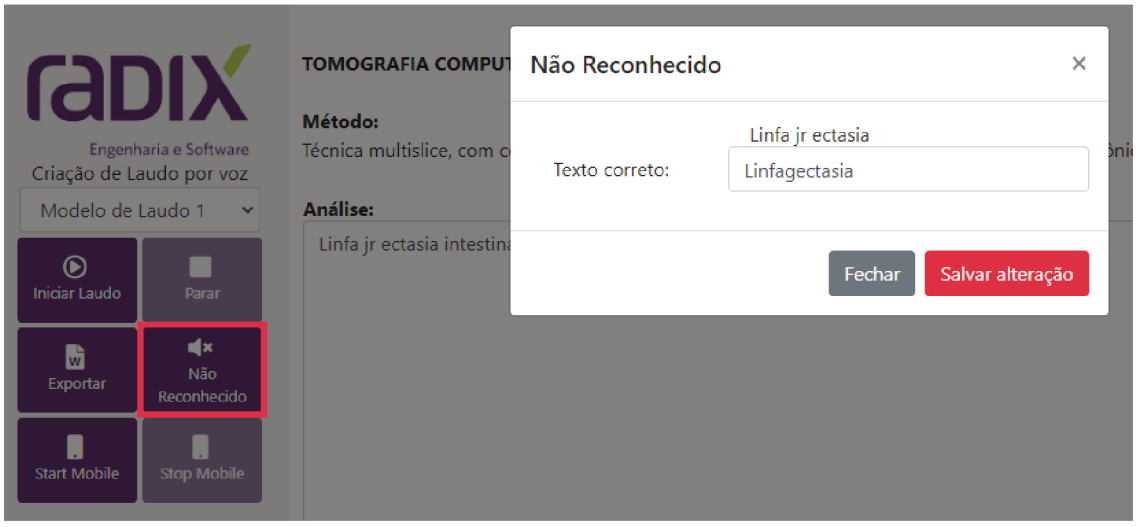
Figura 21 – Utilização do atalho para adicionar auto textos.



Fonte – Radix.

reconhecido e escrever o texto correto, como é mostrado na Figura 22. Assim, cada vez que uma alteração é realizada ela colabora para o crescimento global dos dados.

Figura 22 – Exemplo da utilização da ferramenta para palavras não reconhecidas pelo sistema.



Fonte – Radix.

Para o treinamento foram utilizados apenas alguns termos e jargões médicos, não todos. E com a capacidade de fazer alterações, é possível aumentar a produtividade dos usuários através da melhora de performance do sistema apenas pelo fato de substituir algumas palavras que aparecem na operação.

Além disso, a aplicação também exporta texto para documento e tem comunica-

ção com o celular. Dessa forma, é possível acessar o aplicativo pelo celular e conforme o usuário estiver falando através do microfone do celular o texto é mostrado na tela do computador. Para a empresa essa possibilidade de utilizar o microfone do celular é algo muito interessante, pois elimina a necessidade de comprar microfones dedicados para captação do áudio para ter uma melhor performance do sistema.

Na Figura 23 pode-se ver um esquema sobre como funciona o sistema, como seria o treinamento da aplicação para as solicitações de palavras pelo médico. Basicamente, o sistema para crescimento do conjunto de dados, a solicitação de "smart actions- que foi o nome dado para o reconhecimento de palavras-chave - e a alteração na ferramenta.

Figura 23 – Esquema do funcionamento do sistema desenvolvido.

Fonte – Radix.

Durante a realização do projeto foram encontrados alguns impedimentos. Além das dificuldades iniciais na negociação com o cliente e a clareza na comunicação do que era realmente o esperado versus o que seria desenvolvido, os impedimentos mais técnicos ficaram nas questões das tomadas de decisão das ferramentas a serem utilizadas. Também foram impedimentos os acessos aos serviços cognitivos para efetuar testes nas respectivas ferramentas. E após a decisão da utilização do serviço da Microsoft, a conta gratuita que foi utilizada inicialmente possui limitações em seu uso,

dificultando o avanço das atividades. Não bastou a criação de uma conta no Azure DevOps para o repositório de código, pois também era preciso um diretório no Portal da Azure para criar os serviços cognitivos.

Enfim, o projeto - que possuía promessas de continuidade - foi suspenso por causa da pandemia do Coronavírus. Mas abriu portas para um retorno no futuro e também uma possibilidade da criação de uma ferramenta própria da Radix de reconhecimento de fala com dicionário médico.

8 CONCLUSÃO

O projeto de criação de uma ferramenta inteligente para geração de laudos médicos através do reconhecimento de fala foi, sem dúvidas, um desafio. Frente ao pouco tempo que o projeto teve para ficar pronto para uma prova de conceito, os estudos, as decisões e as implementações realizadas tiveram que ocorrer de forma muito dinâmica.

O passo com maior dificuldade foi a criação do conjunto de dados personalizado com os termos médicos. Após avançar desta fase, o projeto ocorreu de forma fluida e alcançou o resultado esperado tanto para a própria empresa quanto para o cliente do projeto.

O impacto deste projeto para o cliente se reflete em muitos benefícios, não apenas financeiros, mas aumentando a produtividade dos médicos e da equipe técnica. O desenvolvimento da aplicação conta com a possibilidade da utilização do microfone do celular pessoal do profissional da saúde, dispensando a necessidade de aquisição de dispositivos dedicados de microfone. O projeto permite que a ferramenta seja utilizada com o microfone da própria máquina, do fone de ouvido, ou ainda de um dispositivo móvel, como no caso do celular.

A aplicação desenvolvida também que o conjunto de dados tenha um crescimento orgânico. Ou seja, permite que o próprio médico, ao utilizar o sistema, indique quais as palavras que não foram reconhecidas. Isso automaticamente entra no pipeline de transformação e ao chegar na equipe técnica já é realizado um treinamento com as novas palavras que passarão a fazer parte do vocabulário personalizado.

Por fim, o projeto realizado é constituído por multiplataformas, utilizando tanto o aplicativo para celular quanto a aplicação na rede que pode estar em servidores ou plataformas diferentes. Por possuir poucas ou nenhuma restrição, este sistema pode ter diversas possibilidades futuras, como aplicação de técnicas de mineração de texto e de visão computacional.

A mineração de texto envolve aspectos como análise lexical a fim de estudar a frequência de distribuição das palavras, reconhecimento de padrões, extração de informações, entre outros. Caso seja aplicada esta técnica, os textos dos laudos podem ser salvos para realizar análises e possuirá um vasto conjunto de dados para serem estudados.

A visão computacional, por sua vez, é a ciência das máquinas capazes de enxergar. Esta tecnologia desenvolve a construção de sistemas artificiais que obtém informações de imagens ou quaisquer dados multidimensionais. Com a aplicação desta tecnologia, poderiam ser realizados treinamentos das imagens dos exames mais recorrentes, podendo levar a geração automatizada de diagnósticos ou até ideias para novos exames médicos a fim de aprimorar as técnicas já existentes.

REFERÊNCIAS

ALECRIM, Emerson. Machine learning: o que é e por que é tão importante. O que é e por que é tão importante. Disponível em: https://tecnoblog.net/247820/machine-learning-ia-o-que-e/. Acesso em: 21 nov. 2020.

ALVES, Gisely. Entendendo Redes Convolucionais (CNNs). 2018. Disponível em: https://medium.com/neuronio-br/entendendo-redes-convolucionais-cnns. Acesso em: 21 nov. 2020.

BHATTACHARYYA, Jayita. Complete Tutorial On LeNet-5: guide to begin with cnns. Guide To Begin With CNNs. 2020. Disponível em: https://analyticsindiamag.com/complete-tutorial-on-lenet-5-guide-to-begin-with-cnns/. Acesso em: 21 nov. 2020.

DESENVOLVIMENTO ÁGIL. Scrum: metodologia ágil para gestão e planejamento de projetos de software. Disponível em: https://www.desenvolvimentoagil.com.br/scrum/. Acesso em: 12 nov. 2020.

GÉRON, Aurélien. Hands-On Machine Learning with Scikit-Learn and TensorFlow. Sebastopol: O'Reilly Media, 2017.

LeCun, Yann & Bengio, Y. & Hinton, Geoffrey. (2015). Deep Learning. Nature. 521. 436-44. 10.1038/nature14539.

LIMA, Klebiano Kennedy da Silva; MENEZES, Matheus da Silva. Desenvolvimento e Comparação de Redes Neurais Convolucionais para Classificação de Objetos. 2019. 12 f. TCC (Graduação) - Curso de Ciência e Tecnologia, Universidade Federal Rural do Semiárido, Mossoró, 2019.

MACHINE LEARNING. Palo Alto: Stanford University, 2016. Disponível em: https://www.youtube.com/playlist?list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN. Acesso em: 02 jun. 2020.

PYSCIENCE BRASIL. Python: O que é? Por que usar? Disponível em: http://pyscience-brasil.wikidot.com/python:python-oq-e-pq. Acesso em: 12 nov. 2020.

RADIX (Brasil). A Radix. Disponível em: https://www.radixeng.com.br/sobre. Acesso em: 12 nov. 2020.

SUTHERLAND, Jeff; SUTHERLAND, J.J.. Scrum: a arte de fazer o dobro do trabalho na metade do tempo. Rio de Janeiro: Sextante, 2014.

TENSORFLOW. Simple Audio Recognition. Disponível em: https://www.tensorflow.org/tutorials/audio/simple_audio. Acesso em: 12 nov. 2020.

VAN ROSSUM, Guido; DRAKE JR, Fred L. Python tutorial. Amsterdam: Centrum voor Wiskunde en Informatica, 1995.

WIKIPÉDIA. CRUD. Disponível em: https://pt.wikipedia.org/wiki/CRUD. Acesso em: 19 nov. 2020.