Utilização de Processamento de Linguagem Natural para Desenvolvimento de Assistente Pessoal Virtual

Giovani Formaggio Mateus[1], Carlos Roberto dos Santos Junior[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Câmpus Hortolândia CEP 13183-250 - Hortolândia-SP – Brasil

giovanimateus99@gmail.com[1], carlos.rsantos@ifsp.edu.br[1]

Abstract. *With the fact that an individual constantly faces the challenge of executing multiple tasks at the same time, the present work shows the development of a software product that conciliates commands from a speech recognition system with the performance of the device's functionalities, aiming to allow the user to use them without drawing his attention. Therefore, it was studied natural language processing techniques to identify the command, allowing that the system interprets the user speech with a F1-Score of 87% and manages to correctly activate one of its functionalities.*

Resumo. *Diante do cenário de que o indivíduo constantemente se encontra no desafio de executar várias tarefas ao mesmo tempo, este trabalho traz o desenvolvimento de um software que concilia comandos recebidos de um sistema de reconhecimento de voz com a execução de funcionalidades do dispositivo, visando permitir que o usuário as realize sem precisar desviar sua atenção. Para isso, foram exploradas técnicas de processamento de linguagem natural para identificação do comando, permitindo que o sistema interprete a fala do usuário com uma taxa de F1-Score de 87% e consiga ativar corretamente uma das funcionalidades propostas.*

1. Introdução

A partir do período da Revolução Industrial, as máquinas passaram a realizar tarefas desenvolvidas por humanos de forma ágil e precisa, o que requer de qualquer tipo de produção a redução constante do tempo gasto para realizá-la.

Com isso, a agilidade passou a ser um fator essencial para qualquer indivíduo que constantemente se encontra no desafio de executar várias tarefas em um curto período de tempo. Como exemplo, temos uma situação na qual uma pessoa se encontra dirigindo até o trabalho, quando recebe uma mensagem urgente no celular. Como previsto no Art. 252 da Lei nº 9.503, de 23 de setembro de 1997, "caracterizar-se-á como infração gravíssima no caso de o condutor estar segurando ou manuseando telefone celular." [Brasil 2016], portanto é necessário que o condutor pare o carro para a leitura do conteúdo da mensagem, onde o conteúdo pode ser sobre um integrante de sua família precisando de sua ajuda. Para evitar problemas no trabalho, o indivíduo envia uma mensagem ao seu chefe avisando do ocorrido e que não chegará ao trabalho no horário previsto. Com isso, ele desperdiça um tempo que pode ser crítico em determinadas situações.

Paralelamente a isso, para melhorar a utilização de dispositivos eletrônicos, há o surgimento de diferentes métodos para interação, não existindo mais a necessidade da utilização de teclado e *mouse*, substituindo-os por telas sensíveis ao toque e comandos de voz, por exemplo. O controle de dispositivos através de comandos de voz, por exemplo, têm exemplos de utilização em aplicações existentes para aliviar tarefas repetitivas na área corporativa [Costa 2017] e na área hospitalar [Veiga 2017].

Essas aplicações existentes aplicam o comando de voz em assistentes pessoais virtuais, que são agentes de software que interpretam frases ditas por humanos, identificam e executam uma tarefa a partir dela [Tsiao et al. 2007].

Porém, os trabalhos existentes propõem uma aplicação específica de um assistente pessoal, limitando-o a um ambiente controlado e com tarefas bem definidas.

Aliando esses novos tipos de interação com a necessidade da agilidade na execução de múltiplas tarefas simultâneas, esse trabalho tem como objetivo geral permitir que o usuário ative algumas funcionalidades do dispositivo, sem precisar desviar o foco para o mesmo. Dado este objetivo, a proposta deste trabalho é o desenvolvimento de um sistema que interpreta comandos por voz, com a exploração de técnicas de Processamento de Linguagem Natural (PLN), verificando se esse comando é referente a alguma funcionalidade no escopo do projeto. Por fim, caso seja relacionada ao escopo, a funcionalidade é ativada. O sistema desenvolvido atingiu uma média de acertos da funcionalidade requisitada de 87%, ou seja, na grande maioria dos comandos solicitados a ele, a funcionalidade correta foi ativada.

Com isso, o usuário economiza o tempo necessário para a realização de uma tarefa e evita o desvio da atenção de atividades críticas e garante uma maior agilidade na execução das tarefas definidas, uma vez que o usuário poderá utilizar a voz para comandar o dispositivo e requisitar as respectivas funcionalidades.

Além disso, foi construída uma arquitetura para o desenvolvimento e implementação de um assistente pessoal virtual de maneira geral, que pode ser adaptada e aplicada para qualquer cenário, devido à modularização de tarefas do sistema.

Como objetivos específicos, o trabalho visa explorar tecnologias de reconhecimento de voz, técnicas de Inteligência Artificial e *Machine Learning* para tomadas de decisão, estudar o desenvolvimento de aplicações que executam funções semelhantes presentes no mercado e aplicar conceitos de modelagem de software para desenvolver um assistente pessoal virtual.

Neste artigo, na seção 2 são explorados trabalhos sobre a estrutura e aplicação de sistemas similares. Na seção 3 são explicados os principais conceitos relacionados a este trabalho. Na seção 4 é apresentada a arquitetura desenvolvida para o sistema, bem como o processo de construção de cada módulo e detalhes das tecnologias utilizadas. Na seção 5 são apresentados detalhes técnicos de desenvolvimento e de diagramação envolvidos no projeto. Na seção 6 há uma análise dos resultados apresentados pelo sistema, bem como a verificar se o objetivo principal do projeto foi atingido. Por fim, nas seções 7 e 8, estão descritas as considerações finais e os trabalhos futuros referentes a este artigo.

2. Trabalhos Correlatos

A patente US7216080B2, de 2007, apresenta a estrutura de um assistente pessoal comandado por voz dividido em *recognizer*, que é responsável pela transformação de uma expressão verbal em um tipo diferente de informação como, por exemplo, um texto, e *natural-language processor* que analisa essa expressão gramaticalmente e semanticamente e a transforma em uma instrução como uma lista de tarefas, um evento em um calendário ou um registro em uma lista de contatos [Tsiao et al. 2007]. Uma estrutura semelhante foi utilizada nesse trabalho, através da utilização de três módulos com respectivas funções. O primeiro módulo é responsável por ativar a Interface de Programação de Aplicações (API) de reconhecimento de fala. O segundo é responsável por interpretar a fala recebida, aplicando o PLN no processo. Por fim, o terceiro módulo é o responsável por ativar a funcionalidade requerida pelo usuário, através da integração com a respectiva API.

A criação de um assistente pessoal comandado por voz capaz de gerenciar diversas funcionalidades no mundo empresarial também foi analisada [Costa 2017]. Esse assistente visa o auxílio de funcionários que, no mundo de hoje, estão envolvidos em uma quantidade considerável de reuniões e tarefas, não conseguindo dar uma resposta imediata a todos que o contatam. Com isso, o sistema tem como objetivo gerenciar diversas funcionalidades em que é necessária uma ligação telefônica como, por exemplo, o próprio agendamento. O software desenvolvido utiliza a Google Cloud Speech API para converter a fala recebida pelo microfone em um texto que será interpretado. A mesma API será utilizada para o protótipo proposto neste trabalho, assim como o objetivo de automatizar tarefas em um determinado ambiente para que o usuário não necessite interromper tarefas mais importantes.

Existem propostas para a utilização de um software de assistente pessoal em ambientes hospitalares [Veiga 2017]. Esse assistente visa auxiliar a administração de ambientes hospitalares através da gestão automática do processo de atendimento dos usuários e gestão de senhas, de modo a evitar as longas filas de espera nos hospitais. O objetivo em melhorar a qualidade de serviços ou tarefas por conta da automação de alguns tipos de tarefas é um objetivo em comum com o protótipo proposto no presente artigo.

3. Conceitos

Nesta seção, há a apresentação dos principais conceitos relacionados a este trabalho, como a definição e estrutura de um assistente pessoal e a contextualização de discursos humanos.

3.1. Assistente Pessoal Virtual

Um assistente pessoal virtual é um agente de software que interpreta frases ditas por humanos, identificando e executando uma tarefa a partir dela [Tsiao et al. 2007]. Estas frases podem ser ditas de maneiras diferentes, alterando a sequência das palavras, e a interpretação feita pelo assistente não deve ser prejudicada.

É um tipo de software que atua em diferentes funções durante seu funcionamento, cada uma com um objetivo específico dentro de um sistema complexo. Essa diversidade de atuação permite que o assistente pessoal seja altamente modularizado, dividindo sua estrutura de acordo com a respectiva funcionalidade [Tsiao et al. 2007].

Levando em conta as principais características de um assistente virtual, que é receber comandos de voz e identificar e executar uma tarefa a partir dela, ele pode ser dividido em dois módulos. O primeiro é responsável pela transformação de uma expressão verbal em um tipo diferente de informação como, por exemplo, um texto. O segundo já é responsável por analisar essa expressão gramaticalmente e semanticamente, transformando-a em uma instrução como o registro de uma tarefa em uma lista ou de um evento em um calendário [Tsiao et al. 2007].

3.2. Processamento de Linguagem Natural

O programa ELIZA [Weizenbaum 1966] foi desenvolvido com o intuito de permitir conversas através de linguagem natural entre humanos e computadores. Ele é alimentado com sentenças que são analisadas, procurando palavras-chave que definem o contexto da frase.

Este contexto é utilizado para definir qual o significado das sentenças recebidas, permitindo elaborar uma resposta a elas, seja ela uma ação ou uma resposta verbal.

Portanto, é parte fundamental que um *software* leve em consideração o contexto das palavras ao interpretar um texto, atribuindo sempre a semântica correta a uma palavra. Essa importância pode ser observada quando palavras são sintaticamente diferentes mas são

aplicadas ao mesmo contexto, portanto o significado permanece o mesmo [Jurafsky e Martin 2008].

Por exemplo, nas frases “Este carro é muito caro e não tenho dinheiro suficiente” e “Este carro tem o preço alto e não tenho como pagar”, os termos “caro” e “preço alto”, apesar de serem escritas de maneiras diferentes possuem o mesmo significado.

Só é possível examinar esse significado ao analisar a frase como um todo, observando as palavras que acompanham o termo. Utilizando o exemplo anterior, os termos “caro” e “preço alto” podem ser interpretados como algo relacionado a custo, pois estão perto da palavra “carro”, que indica um objeto, e das palavras “pagar” e “dinheiro”, que indicam uma interação monetária.

Para que o *software* realize essa análise de palavras, é necessário que o formato dos dados seja de forma numérica. Para fazer essa transformação aplica-se o PLN, que consiste em aplicar uma série de procedimentos a um texto e o transforma em um formato apropriado para a análise [Jurafsky e Martin 2008].

4. Arquitetura Proposta

Observando a estrutura e as funções de um assistente pessoal, é possível identificar que as tarefas executadas por ele para atingir o objetivo são bem definidas, distintas entre si e seguem uma sequência de execução. Com isso, foram definidas três principais ações para o assistente aqui desenvolvido: reconhecimento de voz, detecção de intenção e a ativação de funcionalidades terceiras. Cada uma dessas ações possuem tecnologias e métodos específicos para a execução, como observado na Figura 1, representando o a sequência desempenhada pelo assistente pessoal.

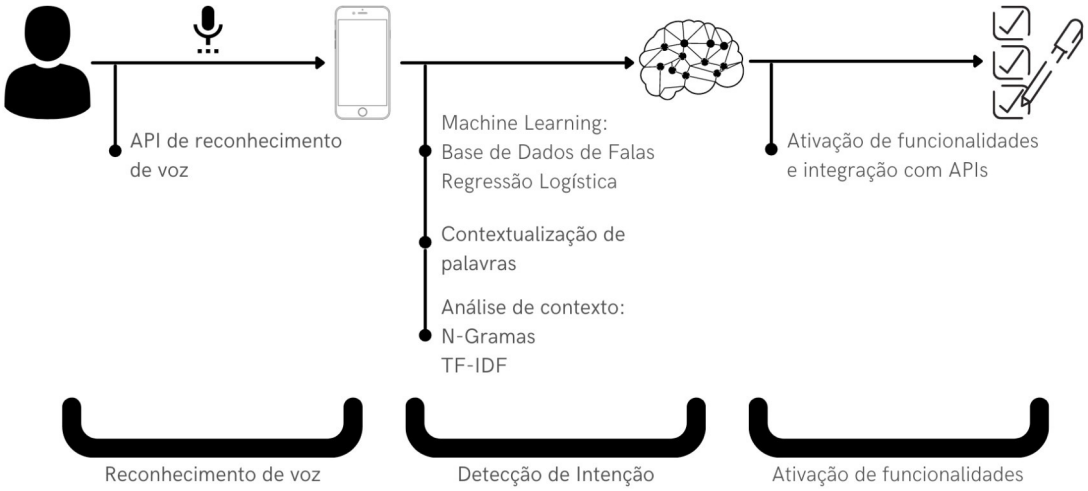

Figura 1. Fluxo de funcionamento, ações e tecnologias empregadas no assistente pessoal.

Ao definir essa estrutura, o software foi dividido em três módulos, em que cada um desempenha um papel e aplica as tecnologias e métodos para o funcionamento. São eles: Reconhecedor, para a tarefa de reconhecimento de voz, Interpretador, para a detecção da intenção do usuário, e Atuador, para ativar a funcionalidade requisitada. Essa divisão é baseada no padrão de projeto MVC (*model-view-controller*) [Johnson 1987].

O módulo Reconhecedor é responsável por receber o áudio através do microfone do dispositivo e utilizar a Google Cloud Speech API para o reconhecimento da fala, transformando a frase em um texto, representando qual o comando do usuário.

Este comando é enviado ao módulo Interpretador, onde é interpretado utilizando técnicas de PLN e *Machine Learning*. Após a interpretação é verificado se o comando é relacionado a um dos casos de uso do sistema.

Caso esteja relacionado a um dos casos de uso, aciona-se o módulo Atuador. Este, por sua vez, possibilita que o comando dito pelo usuário acione a funcionalidade proposta ao ativar a respectiva API, habilitando a integração com outros sistemas.

Utilizando um exemplo de como o sistema é aplicado, ao receber a frase “O que é inteligência artificial?” pelo Reconhecedor, o Interpretador utiliza PLN para pré-processar o texto de modo a permitir a leitura dos dados pelo modelo de *Machine Learning* e obter a intenção contida na frase. A partir disso, o sistema verificaria que a intenção é realizar uma pesquisa em um mecanismo de busca. Com isso, o Atuador fica responsável por utilizar uma integração com a ferramenta de busca do Google para efetivamente realizar a pesquisa e exibir os dados na tela.

4.1. Reconhecimento de Voz

Para o reconhecimento de fala, foi utilizado a biblioteca SpeechRecognition, que possibilita a integração com diversos mecanismos disponíveis para reconhecimento de fala.

Um desses mecanismos é o Google Cloud Speech API. Este mecanismo é desenvolvido pela Google e possibilita o reconhecimento de fala em diversos idiomas, porém necessita de uma conexão com a Internet para funcionar.

Essa mesma API foi utilizada na criação de outros assistentes pessoais comandados por voz em diversos cenários, como em assistente capaz de gerenciar diversas funcionalidades no mundo empresarial [Costa 2017]. Por conta dessa utilização com sucesso em projetos com objetivos semelhantes, a API foi escolhida para o reconhecimento de fala neste trabalho.

4.2. Interpretação

A partir do reconhecimento de fala, o assistente deve identificar qual foi o comando que o usuário gostaria de ativar. Os comandos considerados na elaboração do assistente foram: realizar uma pesquisa em um mecanismo de busca, enviar uma mensagem para outro dispositivo e verificar mensagens recebidas.

O primeiro passo para interpretar qual desses comandos foi escolhido pelo usuário, é a construção de uma base de dados com diversas frases que identificam esses comandos.

4.2.1. Construção da Base de Dados

Para as frases de pesquisa, foi utilizado a fórmula 5WH [Mühlhaus 2000], que estabelece as principais perguntas para a narrativa jornalística. São elas: “Quem”, “O que”, “Onde”, “Quando”, “Por que”, “Como”, “Quanto” e “Quão”. Para a base de dados, essas palavras foram inseridas em uma frase definida de forma aleatória para simular um comando de pesquisa do usuário. Ao criar esta categoria, houve uma preocupação de que a base de dados não ficasse desbalanceada, ou seja, que essa categoria tivesse mais frases para treino do que outras.

Diante disso, percebeu-se que as frases para a classificação de pesquisa eram muito distintas entre si, o que poderia dificultar o estabelecimento de um padrão. Com isso, ela foi dividida em oito classificações diferentes, uma para cada termo que define uma pergunta, citados anteriormente.

Para as frases de envio de mensagens, foi inserido um verbo imperativo no começo da frase que solicite uma ação de comunicação, como por exemplo “envie”, “fale”, e “diga”. Esses verbos foram atribuídos junto a um destinatário, com a utilização de nomes próprios. Então, o último elemento da frase seria a mensagem a ser enviada. Um exemplo dessa construção seria “Envie para Giovani que estou a caminho”.

Por fim, para as frases de recebimento de mensagens foram escolhidas individualmente frases comuns para verificação de mensagens como “tenho algum recado não lido?” ou “Me diga quais são as novas mensagens”.

Com isso, montou-se uma base de dados com oitenta frases no total, divididos em dez categorias, sendo oito frases para cada uma.

4.2.2. Modelo de *Machine Learning*

A construção da base de frases por si só expõe um problema de que nem sempre os usuários emitem um comando para o assistente usando a mesma frase. Frases podem ter significados semelhantes, mas serem ditas de diferentes maneiras durante uma conversa entre humanos. Contudo, essa diferença na estrutura da frase não deve interferir na interpretação pelo interlocutor. Isso dado a diversos fatores como, por exemplo, a habilidade de contextualização dos seres humanos.

Perante essas diferenças, uma comparação simples feita pelo *software* não é o suficiente para identificar o significado. Para isso, um modelo de *Machine Learning* é utilizado para obter uma generalização maior ao comparar, ou seja, pode detectar padrões e semelhanças de significados entre frases sintaticamente diferentes. Após essa detecção, é possível analisar o significado de uma nova frase através de uma comparação com o padrão observado.

Este modelo de *Machine Learning* foi aplicado em conjunto com a técnica estatística de regressão logística. Esta técnica consiste em classificar um objeto entre duas ou mais categorias [Jurafsky e Martin 2008].

A regressão logística aplica um classificador discriminativo. Isso significa que o modelo tenta identificar características diferentes entre as classificações [Jurafsky e Martin 2008]. Por exemplo, ao tentar distinguir carros de motocicletas, o modelo não deduz que possuir rodas é um fator que diferencia as classes, pois ambas as possuem. Porém, ao comparar a utilização de um volante, o modelo deduz que um carro utiliza um volante para ser guiado, enquanto a motocicleta utiliza um guidão.

A implementação da regressão logística é feita em duas etapas. A primeira é a etapa de treino, onde são inseridas amostras de dados e suas respectivas classificações. A segunda etapa é a de teste, onde um objeto de exemplo é aplicado ao modelo treinado com todas as amostras, e há a previsão de qual classificação este objeto pertence. Após isso, é possível analisar a qualidade destas previsões através de métricas que comparam os acertos e erros de classificações [Jurafsky e Martin 2008].

4.2.3. Processamento de Linguagem Natural

Entretanto, o modelo de *Machine Learning* não consegue analisar frases, visto que é utilizada uma medida estatística na sua construção. Para tornar esse texto compatível com o modelo, aplica-se o PLN, através do processo de normalização, que consiste em aplicar uma série de ações a um texto, transformando em um formato padrão, apropriado para a análise [Jurafsky e Martin 2008].

Uma dessas ações é chamada de *tokenization*. Ela consiste em dividir uma frase em suas palavras ou caracteres, utilizando o delimitador de palavras de acordo com o idioma. No caso da língua portuguesa, se refere ao espaço entre elas.

Porém, o processo de *tokenization* aplicado de forma simples, palavra por palavra, não tem um bom resultado com termos com mais de uma palavra como, por exemplo, “Juiz de Fora” ou “por que”. Outro problema é quando se trata de identificação de contexto de palavras escritas da mesma forma, porém com significado distinto, como nos exemplos citados na seção anterior [Jurafsky e Martin 2008].

Para solucionar este problema, deve se observar corretamente o contexto de uma palavra em uma frase. Dito isso, é possível aplicar o conceito de n-gramas. N-gramas são junções de uma palavra com um número *n* de palavras que a precedem em uma frase [Jurafsky e Martin 2008]. Por exemplo, na frase “Este carro é muito caro e não tenho dinheiro suficiente”, ao analisar a palavra “caro” em um quatro grama, ou seja, a palavra com suas três antecessoras, o resultado é “carro é muito caro”.

4.24. Vetorização

Para o processamento dos dados pela regressão logística é necessário que os n-gramas sejam convertidos para o formato numérico. O cálculo é realizado através de um método de representação chamado de vetorização [Jurafsky e Martin 2008]. Comparando as vetorizações de duas frases, se os números atribuídos entre as duas são próximos ou equivalentes, o significado é semelhante.

A atribuição desse número aos n-gramas é feita através de um tipo de vetorização chamado frequência do termo ao inverso da frequência nos documentos (TF-IDF). O TF-IDF calcula qual a importância de um termo, baseado na frequência dele em um conjunto de textos [Jurafsky e Martin 2008]. O resultado atribuído a um termo é um número entre zero e um que pode ser comparado com o de outras frases.

Por fim, após as frases serem processadas pelo TF-IDF combinado com um número de n-gramas, a base de dados mostra qual sequência de valores representa cada funcionalidade que será ativada.

A Figura 2 representa essa vetorização de uma frase específica retirada do banco de dados após todas os procedimentos de normalização e a aplicação do TF-IDF.

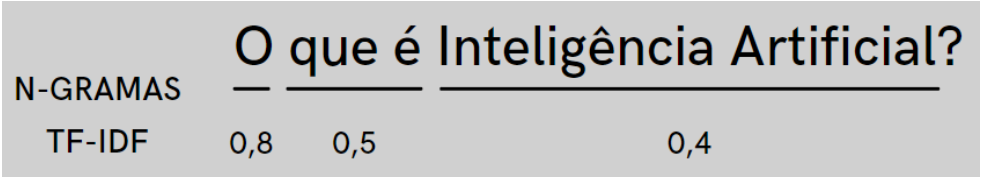

Figura 2. Exemplo de vetorização feita pelo TF-IDF em uma frase do banco de dados.

4.3. Atuador

A integração das funcionalidades foi feita através da utilização de APIs e bibliotecas que fornecem a utilização das funções necessárias.

Para a funcionalidade de pesquisa na internet foi utilizada a biblioteca *browser* do Python, que é responsável por abrir o navegador do dispositivo. Com isso, o comando recebido pode ser utilizado como parâmetro na construção de uma URL para pesquisa no mecanismo de pesquisa do Google. Ao inserir essa URL na barra do navegador, o resultado da pesquisa é exibido.

A integração, tanto da funcionalidade de envio, quanto de recebimento de mensagens foi feita através da utilização da API Telegram Bot API do aplicativo de troca de mensagens Telegram. Esta API permite que, através de solicitações HTTP, alguns comandos do aplicativo sejam emitidos por um outro software programado, por exemplo, em Python.

Os métodos utilizados foram *getUpdates*, para verificar mensagens recebidas, e *sendMessage*, para enviar uma mensagem a um determinado *chat*. Por uma limitação do uso da API, essas funcionalidades foram todas centralizadas em um *bot* criado no próprio Telegram. Esse *bot* foi inserido em um grupo, onde pôde enviar, receber e reproduzir mensagens através dos comandos de voz inseridos no assistente pessoal.

5. Desenvolvimento

5.1. Metodologia de Desenvolvimento

Dentre as diversas metodologias de desenvolvimento existentes, adotou-se o Modelo Evolutivo em Espiral, representado na Figura 3. Esse modelo permite que o sistema seja desenvolvido em etapas (ou ciclos). Cada ciclo parte da coleta requisitos e definição de funcionalidades, avançando pela análise de riscos e prototipação, etapas de testes, e novas funcionalidades são acrescentadas à medida que o processo avança na espiral [Sommerville 2018].

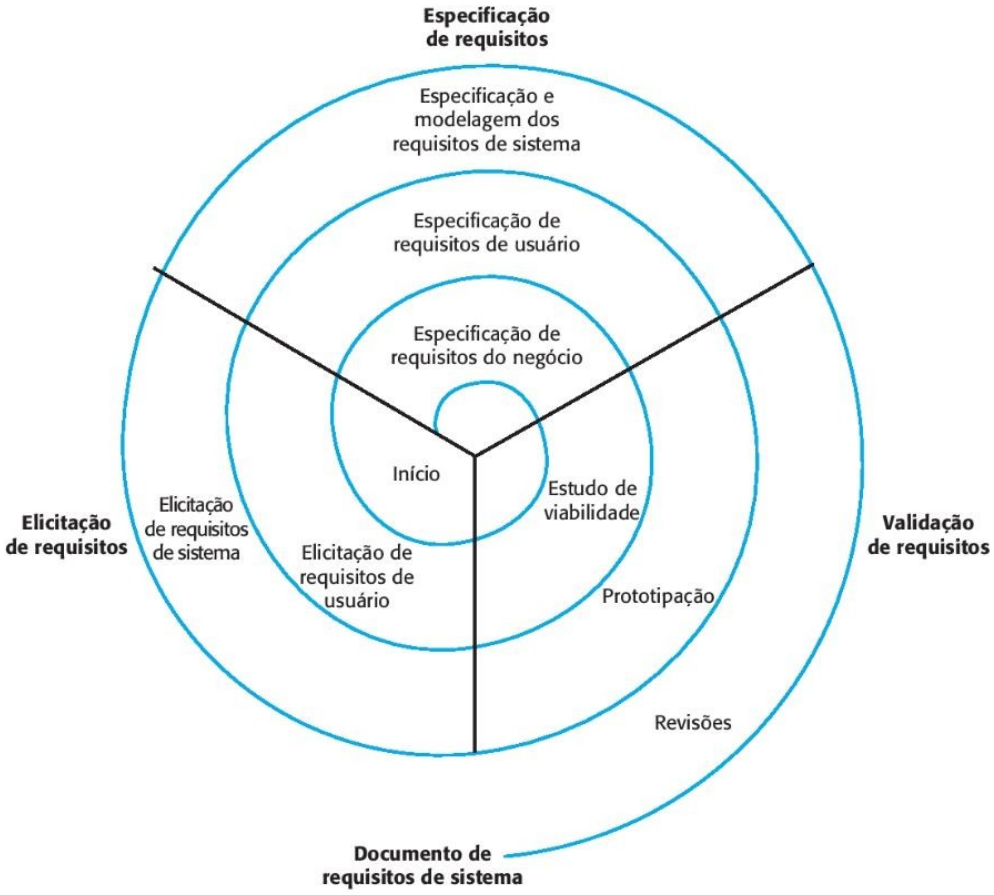

Figura 3. Modelo de desenvolvimento em espiral. Fonte: Sommerville (2018, p. 95).

Com a utilização desse tipo de metodologia, houve a divisão do projeto em módulos, cada um com uma respectiva funcionalidade e desenvolvidos de forma incremental e em fases.

A primeira etapa foi a definição dos requisitos e objetivos do módulo seguido da análise e aprendizado das tecnologias necessárias para que estes sejam cumpridos. Após adquirir conhecimento sobre essas tecnologias (considerando métricas como facilidade de adaptação, facilidade de aprendizado, compatibilidade com o escopo e custo de licença), a segunda etapa foi referente à adaptação da tecnologia mais apropriada para que se encaixasse com o escopo do projeto.

Por fim, a terceira etapa foi referente à avaliação do cumprimento dos requisitos. Caso o requisito fosse atingido de forma correta, a funcionalidade é dada como pronta e, se existirem mais funcionalidades para serem incrementadas, esse ciclo se repete e a nova funcionalidade é submetida às mesmas etapas.

5.2. Diagramas

Para o desenvolvimento deste trabalho, a princípio foram definidas estórias de usuário que demonstram cenários em que fosse necessário o uso de comandos de voz. A partir do estudo dessas estórias, definiu-se as principais funcionalidades que seriam ativadas pelo assistente. Essas funcionalidades foram colocadas em um diagrama de Casos de Uso, que constrói um modelo ilustrativo que facilita o entendimento das interações do sistema [Booch 2012]. Este modelo representa as interações entre um ator e o sistema. Para este trabalho o diagrama mostrado na Figura 4 foi elaborado.

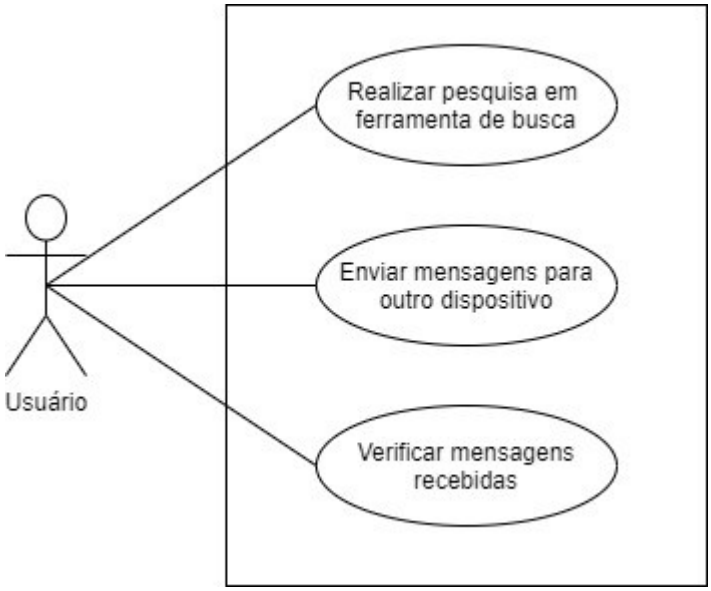

Figura 4. Diagrama de Casos de Uso do projeto.

Para representar a estrutura do sistema de forma visual, pode ser utilizado um diagrama de classe. O diagrama de classe representa objetos de forma estruturada, constando seus tipos, relacionamentos com outros objetos e como se organizam no sistema [Lee 2001]. Dois conceitos do diagrama de classe foram aplicados ao projeto: classes e relacionamentos.

As classes representam os tipos de objetos presentes no sistema. São divididos em atributos e métodos. Os atributos são os dados que definem a classe e os métodos são operações que são aplicadas sobre ela, seja transformando a classe, acessando ou alterando algum de seus atributos [Lee 2001].

Desenvolveu-se o diagrama de classe com base na estrutura mencionada do sistema, com cada módulo representado em uma classe, conforme Figura 5:

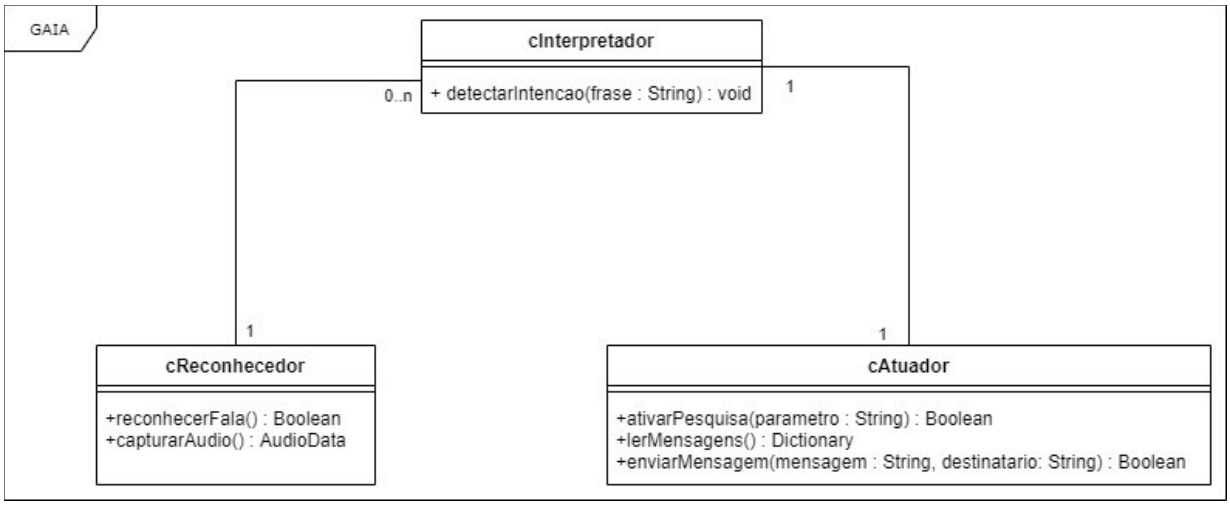

Figura 5. Diagrama de Classe do projeto.

5.3. Tecnologias Utilizadas

A tabela 1 sintetiza as ferramentas e tecnologias utilizadas no desenvolvimento do projeto.

Tabela 1. Ferramentas e tecnologias utilizadas no desenvolvimento do projeto

Python 3.6	Linguagem de programação de alto nível
SpeechRecognition 3.7.1	Módulo da linguagem Python para reconhecimento de fala através de APIs
Google Cloud Speech API	API desenvolvido pela Google para reconhecimento de fala
Pandas 1.2.0	Biblioteca da linguagem Python para leitura de bases de dados em CSV
Scikit-learn 0.23.2	Biblioteca da linguagem Python para Machine Learning
NLTK 3.5	Biblioteca de Python para aplicação de PLN

6. Resultados e Discussões

As previsões emitidas pelo modelo de *Machine Learning* são as responsáveis pela interpretação da fala do usuário, identificando qual o comando emitido pelo usuário. Diante disso, para verificar que o assistente pessoal desenvolvido cumpre seu objetivo, identificando a funcionalidade correta, sua avaliação foi baseada no desempenho do modelo de *Machine Learning*.

Essa avaliação foi através de validação cruzada. Esta é uma técnica de avaliação de modelos de *Machine Learning* que divide uma base de dados em vários segmentos para treino e teste [Devijver 1985], permitindo o teste com partes diversas da base de dados e evitando com que uma classificação específica seja testada mais do que outras.

A medida de avaliação utilizada foi *F1-score*, mostrado na fórmula 3. Esta medida consiste no cálculo da precisão, indicado na fórmula 1 e revocação, também conhecido como *recall*, representado na fórmula número 2. Os cálculos são feitos baseados nos resultados

verdadeiros positivos (VP), falsos positivos (FP), verdadeiros negativos (VN) e falsos negativos (FN) previstos pelo modelo.

$$Precisão = \frac{Verdadeiros\ Positivos\ (VP)}{Verdadeiros\ Positivos\ (VP) + Falsos\ Positivos\ (FP)} \quad (1)$$

$$Recall = \frac{Verdadeiros\ Positivos\ (VP)}{Verdadeiros\ Positivos\ (VP) + Falsos\ Negativos\ (FN)} \quad (2)$$

$$F1 - Score = \frac{2 * Precisão * Recall}{Precisão + Recall} \quad (3)$$

No caso do assistente pessoal aqui desenvolvido, foram testados diferentes números de subconjuntos no teste de validação cruzada. Conforme apresenta a Figura 6, a melhor média da métrica F1-Score foi de 87% encontrada utilizando 8 subconjuntos.

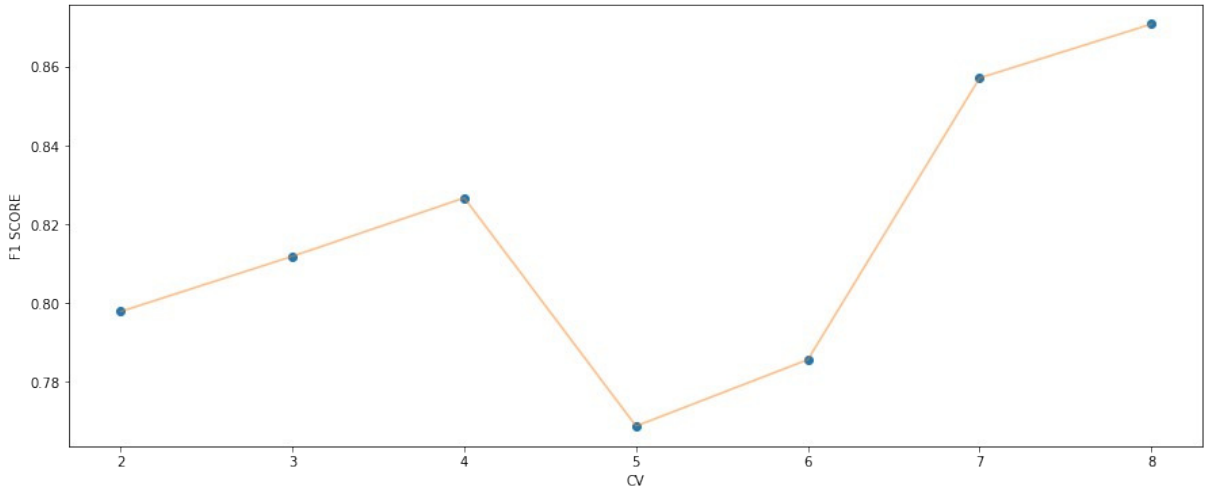

Figura 6: Análise de qual a maior quantidade de subconjuntos para a validação cruzada.

Os resultados detalhados de cada classe podem ser visualizados na matriz confusão da Figura 7, onde os eixos X e Y representam as categorias da base de dados. O objetivo da matriz confusão é cruzar os valores previstos pelo modelo com os valores esperados das base de dados. Nos eixos, os números de 0 a 7 representam a funcionalidade de pesquisa na internet, o número 8 representa a funcionalidade de envio de mensagens e o número 9 representa a funcionalidade de recebimento de mensagens.

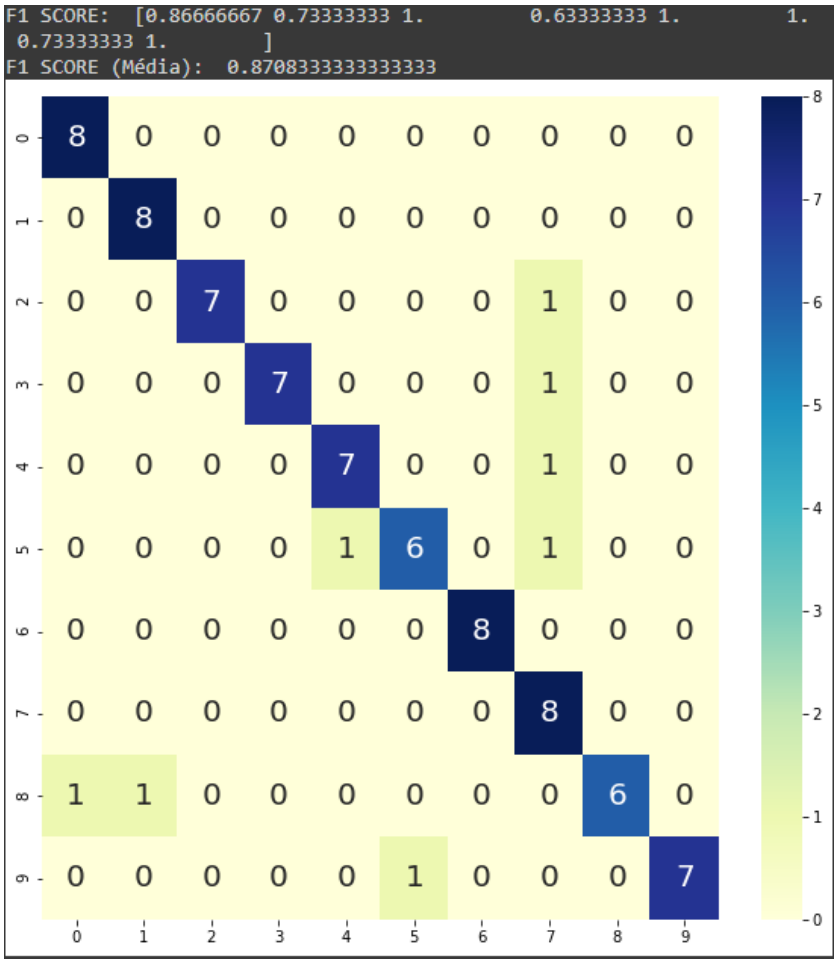

Figura 7: F1-Score por subconjunto e matriz confusão dos resultados do modelo.

Pode ser observado na matriz confusão que, mesmo que algumas previsões tenham sido em uma classificação errada, a ativação da funcionalidade não foi afetada por completo, uma vez que em apenas 3,75% a previsão acabou indicando a funcionalidade errada.

7. Conclusão

O objetivo de desenvolver um assistente pessoal que consiga interpretar comandos recebidos pela fala do usuário foi cumprido. O modelo de *Machine Learning* desenvolvido obteve uma alta taxa de acertos ao prever as funcionalidades requisitadas. Para isso, foram estudados e utilizados conceitos de PLN, como os n-gramas e a combinação com o TF-IDF para contextualização e vetorização de textos.

Estudou-se tecnologias de reconhecimento de fala e bibliotecas da linguagem Python para treinamento, aplicação e avaliação de modelos de *Machine Learning* e a utilização de API para reconhecimento de voz.

Como Trabalho de Conclusão de Curso do curso de Tecnologia em Análise e Desenvolvimento de Sistemas, foram utilizados conhecimentos das matérias de Inteligência Artificial, para criação do modelo de *Machine Learning*, Programação Orientada a Objetos e Linguagens de Programação para estruturação e programação em classes e Engenharia de Software para diagramação.

Por fim, as APIs e bibliotecas de integração foram estudadas e implementadas com sucesso. Isso possibilita a integração com as funcionalidades propostas, dentro de um ambiente modularizado e com alta escalabilidade, permitindo, através do acréscimo de novas

classificações no modelo e novas integrações através de APIs, expandir a área de utilização do assistente desenvolvido.

8. Trabalhos Futuros

A elaboração desta primeira versão do assistente possibilita um desenvolvimento incremental das funcionalidades do mesmo, em que novas funcionalidades podem ser facilmente adicionadas ao escopo devido à alta modularização aplicada à estrutura do projeto, e da possibilidade de acrescentar novas classificações à base. Com isso, é possível, futuramente, adicionar novas funções e integrações e permitir a adaptação a um cenário específico.

Além da escalabilidade do projeto, outra possibilidade a se avaliar seria a aplicabilidade do assistente com usuários reais, através de testes e pesquisas de usabilidade.

Referências

BRASIL (2016). Código de Trânsito Brasileiro (CTB). Lei nº 13.281, de 4 de maio de 2016 que institui o Código de Trânsito Brasileiro. Brasília: Presidência da República.

TSIAO, J. C.-S., Chao, D. Y. and Tong, P. P. (2007). US007216080B2. Cincinatti: EUA Disponível em: <https://patents.google.com/patent/US7216080B2/en>. Acesso em: 07 abr. 2019.

COSTA, J. C. G. (2017). Assistentes Virtuais para Comunicação Empresarial. Porto: Universidade do Porto. Disponível em: <https://hdl.handle.net/10216/107201>. Acesso em: 12 mai. 2019.

VEIGA, A. M. Da (2017). Assistente Pessoal Hospitalar. Minho: Universidade do Minho - Escola de Engenharia. Disponível em: <http://hdl.handle.net/1822/54118>. Acesso em: 12 mai. 2019.

WEIZENBAUM, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1), 36-45.

JURAFSKY, D., & MARTIN, J. H. (2008). Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing. Upper Saddle River, NJ: Prentice Hall.

JOHNSON R. E. (1987). Model/view/controller. Department of Computer Science, University of Illinois at Urbana-Champaign.

MÜHLHAUS, C. (2000). 5W + H: seis questões milenares para a entrevista jornalística. Rio de Janeiro: Universidade Federal do Rio de Janeiro.

SOMMERVILLE, I. (2011). Engenharia de Software. 9a ed. São Paulo: Pearson.

BOOCH, J. (2012). UML, Guia do Usuário. Elsevier.

LEE, R. C., TEPFENHART, W. M. (2001). UML e C++: guia prático de desenvolvimento orientado a objeto. 1a ed. São Paulo: Pearson.

DEVIJVER, P. A., KITTLER, J. (1982). Pattern Recognition: A Statistical Approach, Londres: Prentice-Hall.

Documento Digitalizado Restrito

Anexo I (Artigo) - Giovani Formaggio Mateus - HT1720287

Assunto: Anexo I (Artigo) - Giovani Formaggio Mateus - HT1720287
Assinado por: Carlos Junior
Tipo do Documento: Outro
Situação: Finalizado
Nível de Acesso: Restrito
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Carlos Roberto dos Santos Junior**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 12/08/2021 11:12:36.

Este documento foi armazenado no SUAP em 12/08/2021. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 741620
Código de Autenticação: 2764cb1cf6