Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uso de Chatbots para Apoio ao Atendimento de Clientes no Aplicativo Telegram

Amanda Oliveira Alves

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Dibio Leandro Borges

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uso de Chatbots para Apoio ao Atendimento de Clientes no Aplicativo Telegram

Amanda Oliveira Alves

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Dibio Leandro Borges (Orientador)
CIC/UnB

Prof. Dr. Jan Mendonca Correa
CIC/UnB

Prof. Dr. Wilson Henrique Veneziano
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 24 de Maio de 2021

Dedicatória

Dedico este trabalho ao meu pai Roberlandio e a minha mãe Andreia, que desde sempre me apoiaram e fizeram de tudo para que eu pudesse chegar até aqui. Minha irmã Mariana, que me enche mais de orgulho a cada dia que passa. Dedico também a minha amiga Rebeca, que com paciência respondeu as minhas dúvidas e me ajudou a realizar este trabalho.

Agradecimentos

Agradeço, primeiramente, à orientação do Professor Díbio. Agradeço também a todos os professores inspiradores que contribuíram para o meu desenvolvimento pessoal e profissional. Expresso minha gratidão também aos autores que disponibilizaram conteúdo de qualidade publicamente na Internet. Por fim, agradeço a todas as pessoas que me auxiliaram, de forma direta ou indireta, a produzir este trabalho.

Resumo

Atualmente o uso de chatbots em aplicações online de larga escala, se tornou comum em diversas áreas, principalmente nos sistemas com módulos especializados em atendimento ao cliente. Para oferecer um suporte inteligente para os usuários, os bots precisam interpretar as interações feitas e fornecer as respostas corretas, de forma rápida. O trabalho apresenta o Lu, um protótipo de chatbot que auxilia os clientes da Pizzaria Smart (uma pizzaria fictícia), a realizarem os seus pedidos para entrega. Os passos necessários para a realização do trabalho foram: definir a identidade do bot, construir o fluxo conversacional e desenhar a arquitetura do sistema. Por fim, após a implementação foi conduzido um experimento com o objetivo de demonstrar o desempenho do modelo projetado.

Palavras-chave: Chatbot, Telegram, Desenvolvimento de Software

Abstract

Currently, the use of chatbots in large-scale online applications has become common in several areas, mainly in systems with specialized modules in customer service. In order to provide intelligent support for users, bots need to interpret the interactions made and provide the correct answers quickly. The work presents Lu, a chatbot prototype that helps Pizzaria Smart customers (a fictional pizzeria), to place their orders for delivery. The necessary steps to carry out the work were: define the bot's identity, build the conversational flow and design the system architecture. Finally, after implementation, an experiment was conducted with the objective of demonstrating the performance of the projected model.

Keywords: Chatbot, Telegram, Software Development

Sumário

Lista de Figuras

Capítulo 1

Introdução

Um *chatbot* é um assistente virtual capaz de responder perguntas feitas por um usuário humano dando respostas corretas[1]. É uma tecnologia versátil, existem *chatbots* sendo utilizados tanto para automatização de tarefas complexas, como também para auxiliar seres humanos em atividades do cotidiano.

Plataformas como Facebook, WhatsApp e Telegram já hospedam aplicações deste tipo para diversos propósitos como: *e-commerce*, educação, saúde, produtividade, entretenimento e muitos outros[2].

1.1 História

Alan Turing, mundialmente conhecido como o pai da ciência da computação, em 1950 trouxe o seguinte questionamento: "Máquinas são capazes de pensar?", a partir disso ele criou o "Jogo da Imitação", atualmente conhecido como o Teste de Turing[3]. O jogo funciona como um interrogatório, o interrogador faz perguntas para outros dois participantes, através das perguntas ele tenta descobrir qual deles é um humano e qual deles é uma máquina. Se o interrogador for incapaz de encontrar a resposta correta, isso significa que a máquina passou no teste, o que equivale a dizer que ela conseguiu se passar por um humano[3].

Em 1966, Joseph Weizenbaum, considerado um dos pais da inteligência artificial criou o primeiro *chatbot* ELIZA. O sistema era capaz de identificar palavras-chave e padrões, combinando estes elementos produzia respostas apropriadas para as suas interações. Depois da criação de ELIZA, a indústria percebeu o valor dos *chatbots* e começou a investir no desenvolvimento de *chatbots* inteligentes[3].

1.2 Atualidade

A implementação de *chatbots* em serviços de atendimento ao cliente vêm crescendo com velocidade nos últimos anos, eles são utilizados para auxiliar ou até mesmo substituir a força de trabalho humana. Para criar uma sensação de proximidade com o cliente, muitos *chatbots* possuem nomes e até mesmo personalidades definidas.

A tecnologia utilizada nessa categoria de software não é novidade, porém se tornou mais popular com o passar do tempo, principalmente com a adição de técnicas de aprendizagem de máquina. Entre 2007 e 2015, os *chatbots* participaram de cerca de um terço a metade de todas as interações online e este número continua aumentando[4].

Atualmente, esta tecnologia é utilizada de diversas formas, uma das mais comuns que encontramos é como ferramenta para atendimento ao cliente. O site JusBrasil[5] visa ajudar pessoas a encontrarem advogados para resolverem questões jurídicas. Para encontrar estes profissionais os usuários interagem com um *chatbot*, a partir das respostas inseridas o sistema fornece opções clicáveis, para que o usuário consiga navegar pelo fluxo conversacional do bot.

A rede Magazine Luiza[6], possui em torno de 20 mil funcionários e um total de 830 unidades, contando as lojas físicas e o site. A Lu é a *chatbot* que fica hospedada no site da empresa, ela viabiliza o atendimento ao cliente de forma automática e conversacional. Ela atende solicitações de rastreamento de entrega, emissão de segunda via de boleto, informações sobre status de pedidos, entre outras. A Lu foi reconhecida como melhor *chatbot*, na categoria 'Serviços' pela premiação Bots Brasil Awards 2017, a indicação foi feita pelo público[7].

1.3 Problema

Devido a sua flexibilidade e facilidade de uso, alguns especulam que os *chatbots* podem se tornar uma interface de uso universal e talvez substituir os aplicativos móveis[8]. Por isso, é importante entender os problemas associados ao desenvolvimento e implementação deste tipo de software.

Existem atributos que podem ser utilizados para mensurar a qualidade de um *chatbot*, com estes podemos avaliar se um *chatbot* possui um bom nível de qualidade, ou não[4]. A construção de bons fluxos conversacionais é uma parte muito importante no desenvolvimento de um *chatbot*. Para que uma conversa seja bem sucedida, é importante responder a todas as interações do usuário com respostas úteis e adequadas ao domínio do problema[1].

Vivemos em um mundo diverso, cada pessoa se encontra em um contexto específico, por isto não podemos esperar que todos interajam com os produtos digitais da mesma forma[9]. Ao iniciar o desenvolvimento de um novo agente de conversação, especialmente para atendimento ao cliente, o ideal é saber de antemão quais perguntas os usuários em potencial farão[10].

Embora o bot seja projetado para funcionar em uma esfera particular, as pessoas tendem a fazer perguntas de natureza geral ou até mesmo aleatórias, para induzir o bot ao erro e encontrar seus pontos fracos[10]. As pontuações positivas e negativas são assimétricas, estudos sugerem que respostas inadequadas têm um efeito negativo mais forte na percepção do usuário do que as respostas incorretas, mas que são adequadas ao contexto[10].

1.4 Objetivo Geral

O trabalho em questão apresenta a implementação de um *chatbot* no aplicativo Telegram[11], o seu objetivo é se comunicar com os clientes de uma pizzaria fictícia e permitir que estes realizem pedidos para entrega.

1.5 Objetivos Específicos

Os atributos utilizados para aferir a qualidade de um sistema, estão geralmente alinhados com o conceito de usabilidade do ISO 9241: "A eficácia, eficiência e satisfação com quais usuários especificados alcançam objetivos específicos em ambientes específicos"[4].

A avaliação destes critérios nem sempre é viável devido a restrições de tempo e recursos[10]. Neste trabalho foi utilizada uma abordagem simplificada que permite avaliar rapidamente a performance do *chatbot* implementado.

O trabalho aconteceu através dos seguintes passos:

- Definição da identidade do bot.
- Construção do fluxo conversacional.
- Implementação do bot utilizando API Telegram[12], Python[13] e Replit[14].
- Execução de experimento com trinta voluntários para avaliar o desempenho do sistema.
- Armazenamento de dados em uma planilha via API Google Sheets[15].
- Análise dos dados para extração de resultados.

1.6 Organização do trabalho

Os próximos capítulos estão organizados da seguinte forma:

Capítulo 2: Desenvolvimento. Este capítulo contém o passo a passo da implementação do *chatbot*.

Capítulo 3: Resultados. Este capítulo contém a explicação do experimento e os resultados obtidos.

Capítulo 4: Conclusão. Este capítulo contém a conclusão do trabalho e sugestões para melhorias futuras.

Capítulo 2

Desenvolvimento

Os *chatbots* são interfaces conversacionais, o seu funcionamento pode ser construído com base em regras, ou inteligência artificial[9]. Para a solução exposta neste trabalho foi utilizada a abordagem baseada em regras, as principais diferenças entre os dois tipos são:

Baseado em regras

- As respostas acontecem através de comandos específicos.
- Segue um fluxo conversacional bem definido.
- Em caso de erro, o sistema é incapaz de interpretar a entrada.
- A sua inteligência vai até onde o código permite.

Inteligência artificial

- O sistema possui uma "mente artificial", o usuário não precisa fornecer comandos precisos, o bot compreende linguagem natural.
- O bot aprende e se torna mais inteligente à medida que interage com outros usuários.

2.1 Identidade

O nome do *chatbot* é Lu, ele se identifica com o gênero masculino e dialoga utilizando português coloquial. Foram adicionados ao fluxo conversacional emojis e respostas descontraídas, com isso é transmitida a ideia de uma personalidade divertida.

2.2 Fluxo Conversacional

O Lu consegue entender áudio e texto, o que aumenta as opções de comunicação com o usuário. As interfaces de conversação são capazes de oferecer para os usuários a oportuni-

dade de declarar o que desejam em seus próprios termos, assim como fariam se estivessem interagindo com um humano, o sistema deve lidar com a complexidade envolvida na interação.

O *chatbot* deve ter um pensamento lógico que se encaixe com o domínio do problema, tarefas e comportamentos. Os usuários esperam que o bot lembre-se de informações, como o seu nome ou tópicos já ditos na conversa. Além disso, eles esperam flexibilidade na comunicação: múltiplas opções de feedback e liberdade de expressão.[2].

Ao percorrer o fluxo conversacional o usuário basicamente responde a várias perguntas do *chatbot*, este precisa de detalhes para confirmar o pedido do cliente, a figura 2.1 ilustra algumas questões que são tratadas em um diálogo prático.

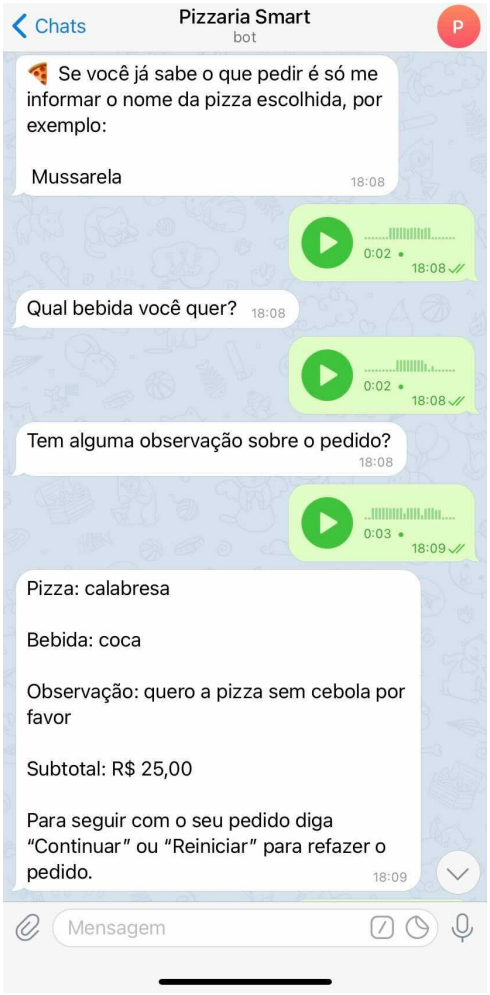

Figura 2.1: Interação entre Lu e cliente.

2.3 Metodologia Ágil

A metodologia ágil foi criada por volta de 1990, visando ser uma alternativa a outras metodologias da época, como o modelo cascata[16]. Na grande maioria dos casos, sistemas de software estão em constante evolução e as mudanças acontecem de forma acelerada.

Para satisfazer esta realidade surgiu a Metodologia Ágil, que prioriza "Indivíduos e interações mais que processos e ferramentas"[17], esta e outras práticas da metodologia estão registradas no Manifesto Ágil. A abordagem ágil posiciona o usuário no centro do desenvolvimento de software, exigindo que aconteça grande interação entre os desenvolvedores e o cliente.

Um projeto que segue a metodologia ágil é construído de forma incremental, sendo assim ele é dividido em vários projetos menores, conhecidos como Sprints. A cada nova Sprint são definidos os próximos passos do projeto, o time cria a documentação, implementa as funcionalidades combinadas, e por fim, ao final do ciclo, que geralmente dura de uma a duas semanas, o que foi feito é validado diretamente com o cliente. Depois, a equipe segue para a próxima Sprint repetindo o processo[16].

Por conta das características citadas, o desenvolvimento ágil casou-se bem com o aparecimento de outras tecnologias que estavam em ascensão na época, conquistando grande espaço no mundo inteiro. Para o desenvolvimento do *chatbot* Lu a autora seguiu, parcialmente, as práticas da metodologia ágil. Selecionando somente aquelas que fariam sentido para a realidade do projeto, de modo a facilitar e acelerar o seu desenvolvimento.

2.4 Telegram

Com o desenvolvimento de dispositivos móveis, como smartphones, tablets, etc, a facilidade de acesso à internet móvel e o aumento da popularidade de aplicativos móveis, os serviços de mensageria ganharam espaço na vida de pessoas de todas as idades. Os serviços de mensageria são ferramentas capazes de compartilhar mensagens de texto, áudio, vídeo, imagens, localização e mais. O aplicativo de mensagens mais popular no mundo é o WhatsApp[18], que pertence ao Facebook[19], a empresa líder no segmento de redes sociais. Em 2018, o número de usuários ativos em serviços de mensageria atingiu a marca de 1,5 bilhão e continua crescendo[20].

De acordo com pesquisas, O Telegram é o serviço de mensageria que mais cresce no Brasil e já está presente em 45% dos smartphones dos brasileiros[21]. Esse tipo de ferramenta forma uma parte importante da comunicação que acontece na internet, oferecendo aos usuários praticidade, economia e velocidade.

Existem várias aplicações deste tipo, entretanto o Telegram se destaca pela facilidade na criação de bots e canais de comunicação diversificados[20]. Por este motivo a API do Telegram foi escolhida para a construção do bot, porém o design proposto pode ser replicado em outras plataformas, sendo necessárias algumas modificações. Para que o bot funcione dentro do Telegram, é necessário registrá-lo no BotFather[22], que é um bot criado para registrar outros bots dentro do Telegram.

2.5 Arquitetura do Sistema

A arquitetura é composta por três módulos: Front-end, back-end e base de dados. O código do *chatbot* pode ser encontrado nesse repositório do Github[23].

- O front-end é a camada de apresentação, ela oferece a interface de interação com o usuário, que pode ser acessada através de dispositivos como tablets, smartphones e computadores. O front-end recebe comandos do usuário e fornece as saídas conforme as regras implementadas no back-end.
- O back-end gerencia as operações que passam despercebidas pelo usuário final. Ele implementa o fluxo conversacional, executa as regras de negócio e comanda o armazenamento dos dados coletados, trabalhando em conjunto com a base de dados.
- A base de dados armazena os dados indicados pelo back-end que foram fornecidos pelo usuário através do front-end.

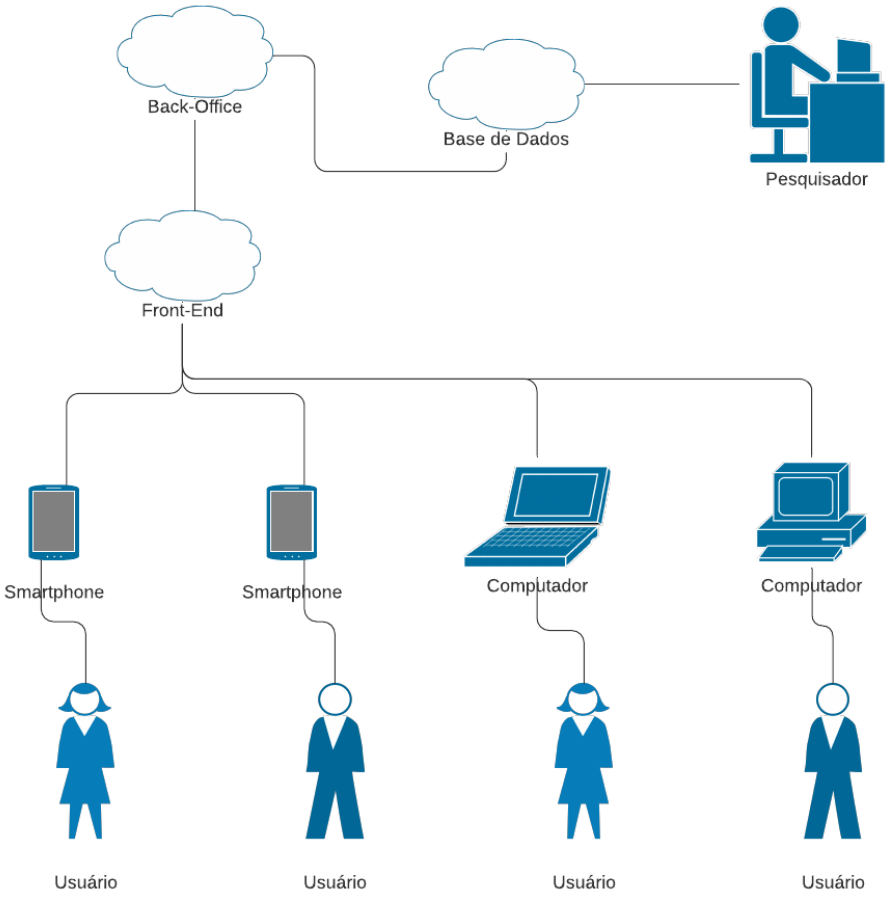

Figura 2.2: Arquitetura do sistema.

2.6 Front-end

O front-end é integralmente representado pelo aplicativo do Telegram, tanto as versões publicadas para o sistema operacional iOS[24], como as versões publicadas para o sistema operacional Android[25].

Aqui acontece o contato com o usuário, ou seja, a troca de mensagens entre os agentes envolvidos na comunicação. Este módulo, também é responsável pelo comportamento da interface, controla as ações de cliques e o carregamento de dados. Os módulos estão sempre conversando, as mensagens enviadas pelo usuário são transmitidas para o back-end, que retorna as respostas do Lu através do front-end. Quando o usuário acessa o Lu pela primeira vez ele encontra a tela ilustrada na figura 2.3.

Figura 2.3: Tela inicial do sistema.

2.7 Back-end

Para construir o back-end da aplicação foi utilizada a linguagem Python, este módulo é responsável por executar a lógica do sistema, sendo formada por regras e comandos que não são percebidos pelo usuário final. É a camada que recebe as requisições do front-end, a responsável por de fato enviar e receber as mensagens do Lu. Através da função **criar_resposta()** foi programado o fluxo conversacional construído para o *chatbot*.

```
1 def criar_resposta(self, mensagem, primeira_mensagem):
2         if primeira_mensagem == True or mensagem.lower() in ('oi', 'ol ',
    'boa noite', 'bom dia', 'boa tarde', 'eai', 'ola', '/start'):
3             return apresentacao
4 
5         elif mensagem.lower() in ('menu', 'card pio', 'cardapio'):
6             chat_id = self.bot.get_updates()[-1].message.chat_id
```

```
7
8           self.bot.send_document(chat_id=chat_id, document=open('
    cardapio_pizzaria_smart.pdf', 'rb'))
9           return cardapio
10
11      elif mensagem.lower() in ('como funciona?', 'como funciona'):
12        return explicacao
13
14      elif mensagem.lower() in ('calabresa', 'mussarela', 'portuguesa', '
    bacon', 'chocolate', 'milho', 'milho verde', 'portugueza', 'pizza
    calabresa', 'pizza mussarela', 'pizza bacon', 'pizza chocolate', 'pizza
    milho', 'pizza milho verde', 'pizza portugueza'):
15        self.sabor = mensagem
16        return 'Qual bebida voc  quer?'
17
18      elif mensagem.lower() in ('coca-cola', 'coca', 'cocacola', 'fanta
    laranja', 'fanta', 'sprite', 'esprite', 'colombina', 'colorado', '
    limonada', 'nenhuma', 'n o quero', 'n', 'nao quero'):
19        self.bebida = mensagem
20        return 'Tem alguma observa  o sobre o pedido?'
21
22      elif self.observacao is None:
23        self.observacao = mensagem.lower()
24        return 'Pizza: {}\n\nBebida: {}\n\nObserva  o: {}\n\nSubtotal:
    R$ 25,00\n\nPara seguir com o seu pedido diga     Continuar     ou
     Reiniciar     para refazer o pedido.'.format(self.sabor, self.bebida,
    self.observacao)
25
26      elif mensagem.lower() in ('continuar', 'continuar pedido'):
27        return frete
28
29      elif self.endereco is None:
30        self.endereco = mensagem
31        return pagamento
32
33      elif mensagem.lower() in ('cr dito', 'credito' 'cart o', 'cartao'
    , 'debito', 'd bito', 'dinheiro', 'cart o', 'cartao'):
34        self.forma_pagamento = mensagem
35        if self.observacao in ('n o', 'nao', 'n', 'sem obs'):
36          return '      SACOLA\n\nPedido para entrega\n\nSubtotal: R$
    25,00\n\nFrete: R$ 5,00\n\nValor total: R$ 30,00\n\nTempo estimado: 30 a
     60 minutos\n\nPizza: {}\n\nBebida: {}\n\nEndere o: {}\n\nForma de
    pagamento: {}\n\nPara seguir com o seu pedido diga     Finalizar     ou
     Reiniciar     para refazer o pedido.'.format(self.sabor, self.bebida,
    self.endereco, self.forma_pagamento)
```

```
37          else:
38            return '       SACOLA\n\nPedido para entrega\n\nSubtotal: R$
    25,00\n\nFrete: R$ 5,00\n\nValor total: R$ 30,00\n\nTempo estimado: 30 a
     60 minutos\n\nPizza: {}\n\nBebida: {}\n\nObserva  o: {}\n\nEndere o:
     {}\n\nForma de pagamento: {}\n\nPara seguir com o seu pedido diga
        Finalizar      ou      Reiniciar      para refazer o pedido.'.format(self.
    sabor, self.bebida, self.observacao, self.endereco, self.forma_pagamento
    )
39
40        elif mensagem.lower() in ('finalizar', 'finalizar pedido'):
41          return avaliacao
42
43        elif mensagem.lower() in ('1', '2', '3', '4', '5', 'p ssimo', '
    pessimo', 'ruim', 'regular', 'bom', ' timo ', 'otimo'):
44          self.nota = mensagem
45          values_list = worksheet.col_values(1)
46          worksheet.update_cell(len(values_list)+1, 1, self.nota)
47          self.endereco = None
48          self.observacao = None
49          return 'Anotei aqui! Obrigado pela participa  o.'
50        elif mensagem.lower() in ('reiniciar', 'reiniciar pedido'):
51          return apresentacao
52        else:
53            return 'Ops, n o consegui te entender.'
```

O back-end permite a comunicação entre aplicações diferentes através de diversas bibliotecas e API's disponíveis para uso. A biblioteca python-telegram-bot[26] fornece uma interface Python pura para a API do Telegram. Além da implementação pura da API, esta biblioteca apresenta várias classes de alto nível para tornar o desenvolvimento de bots fácil e direto.

Para implementar o reconhecimento de voz a função **transcribe_voice()** foi construída e a biblioteca speech_recognition[27] utilizada. A biblioteca speech_recognition possui dependência com a biblioteca PyAudio[28], portanto as duas devem ser instaladas, para que o bot funcione corretamente.

```
1 # Transcreve audio para texto
2     def transcribe_voice(self,context):
3         # Captura mensagem de voz
4         voice = self.bot.getFile(context['message']['voice']['file_id'])
5
6         # Transforma a mensagem de voz de audio/x-opus+ogg para audio/x-wav
7         ft.transcode(voice.download('file.ogg'), 'wav')
8
9         # Extrai a voz do arquivo de audio
```

```
        r = sr.Recognizer()
        with sr.WavFile('file.wav') as source:
            #r.adjust_for_ambient_noise(source)
            audio = r.record(source)

        # Converte audio para texto
        try:
            txt = r.recognize_google(audio, language='pt')

        except sr.UnknownValueError:
            txt = 'Erro'

        print(txt)
        return txt
```

As funções **obter_novas_mensagens()** e **responder()** controlam as requisições de troca de mensagens entre o usuário e o *chatbot*.

```
def obter_novas_mensagens(self, update_id):
        link_requisicao = f'{self.url_base}getUpdates?timeout=100'
        if update_id:
            link_requisicao = f'{link_requisicao}&offset={update_id + 1}'
        resultado = requests.get(link_requisicao)

        return json.loads(resultado.content)
```

```
def responder(self, resposta, chat_id):
        link_requisicao = f'{self.url_base}sendMessage?chat_id={chat_id}&
    text={resposta}'
        requests.get(link_requisicao)
```

Todo o módulo backend foi desenvolvido dentro do Replit, que é uma IDE[29] baseada em nuvem[30], com ele é possível gerenciar dependências, criar scripts, executar comandos shell e colocar aplicações no ar instantaneamente. A ferramenta funciona como um conteiner[31] em execução dentro de uma máquina virtual[32]. Assim, o Lu foi compartilhado rapidamente, facilitando a execução do experimento e a obtenção dos feedbacks.

2.8 Base de Dados

A base de dados funciona da seguinte forma: O back-end se conecta com a API Google Sheets, para isto foi utilizada a biblioteca gspread[33], captura as mensagens indicadas e armazena na planilha ilustrada na figura 2.4.

```
import gspread # pip install gspread
gc = gspread.service_account(filename = 'credentials.json')
sh = gc.open_by_key('')
worksheet = sh.sheet1
```

A avaliação aconteceu no final do fluxo conversacional, foram capturadas trinta respostas, cada resposta foi gravada em uma linha da coluna Notas.

	A	B	C	D	E	F	G	H	I
1	**Notas**	**Notas Numéricas**	**Média**	**Mediana**	**Moda**	**Desvio Padrão**	**Maior Nota**	**Menor Nota**	**Escala**
2	Ótimo	5	3,87	4	4	0,90	5	2	1. Péssimo
3	4	4							2. Ruim
4	5	5							3. Regular
5	Ótimo	5							4. Bom
6	regular	3							5. Ótimo
7	4	4							
8	5	5							
9	5	5							
10	2	2							
11	4	4							
12	regular	3							
13	4	4							
14	4	4							
15	4	4							
16	4	4							
17	Regular	3							
18	5	5							
19	4	4							
20	ruim	2							
21	3	3							
22	3	3							
23	4	4							

+ ≡ Experimento Participantes

Figura 2.4: Planilha de resultados.

Capítulo 3

Resultados

O cliente após tomar conhecimento sobre o canal de atendimento, entra em contato com a Pizzaria Smart, buscando assim satisfazer uma necessidade. O *chatbot* continua o contato, interagindo com o cliente até este conseguir finalizar a sua compra, no final o cliente avalia a qualidade do atendimento recebido. Na figura 3.1 é possível ver que o usuário pode conversar com o Lu através de áudio e texto.

Figura 3.1: Cliente fazendo pedido na Pizzaria Smart.

3.1 Experimento

Para avaliar a atuação da solução foi conduzido o seguinte experimento:

- O *chatbot* foi disponibilizado online através do link `t.me/PizzariaSmartBot`.
- Trinta pessoas de diferentes idades foram escolhidas aleatoriamente e convidadas para participar.
- Após aceitar o convite, a pessoa recebeu o link de acesso ao Lu, com uma rápida explicação sobre o objetivo do teste.
- Ao chegar no final do fluxo os usuários deveriam avaliar o atendimento do Lu.

3.2 Amostra

Trinta pessoas participaram do experimento, a ideia foi convidar um grupo de usuários diverso, de diferentes idades e gêneros. Na figura 3.2 podemos visualizar a composição de gênero e na figura 3.3 a composição de idade dos voluntários.

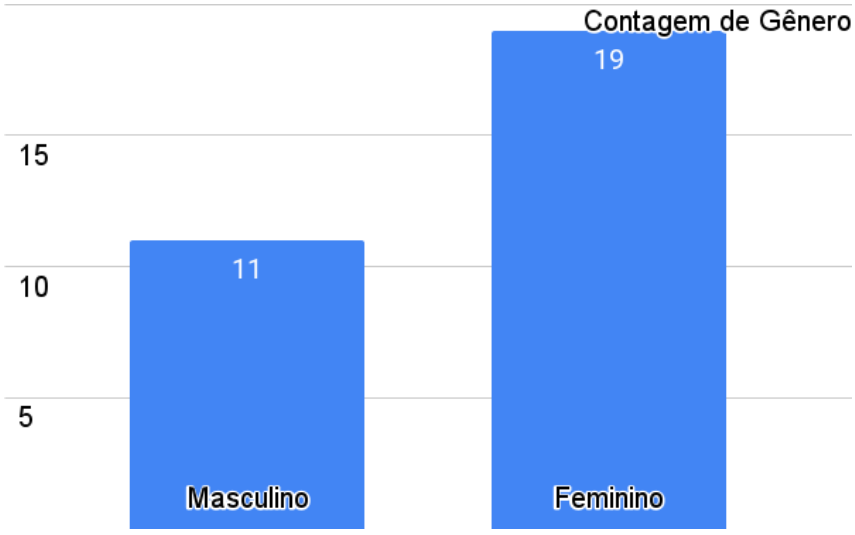

Figura 3.2: Composição de gênero da amostra.

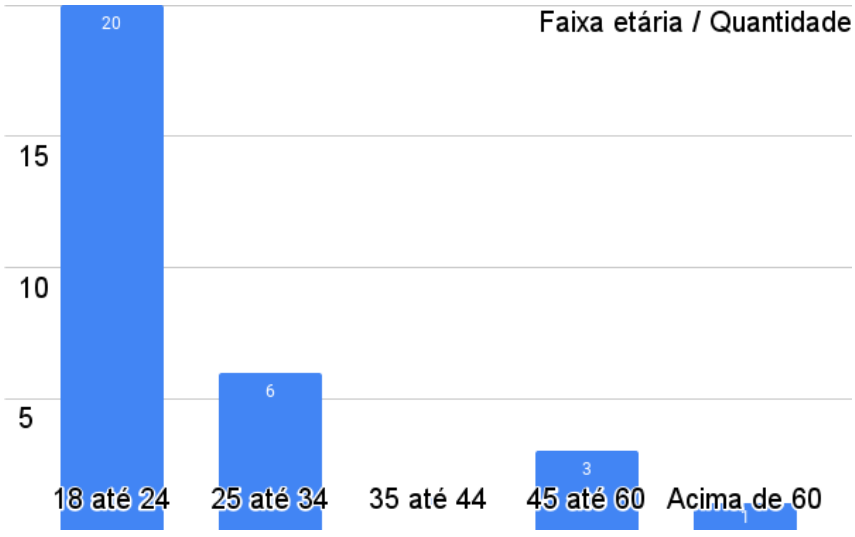

Figura 3.3: Composição de idade da amostra.

3.3 Análise

Sem dúvida, o atendimento ao cliente no formato digital está desempenhando um papel cada vez mais importante na modernização dos estabelecimentos comerciais em todo o Brasil. Por isso, os proprietários devem se concentrar na melhoria dos seus serviços, focando em usabilidade para que todos possam usufruir da tecnologia.

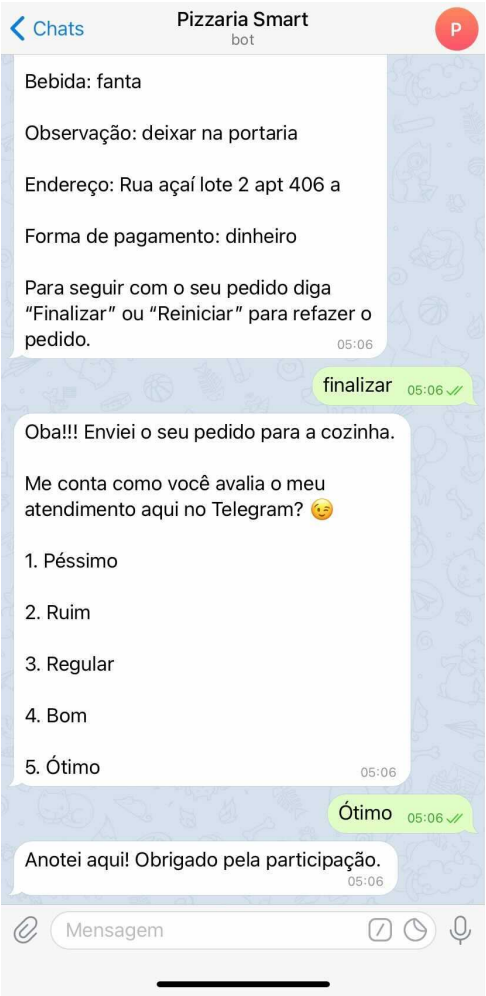

Figura 3.4: Cliente avaliando o atendimento do Lu.

As notas coletadas foram armazenadas na coluna Notas da planilha para análise e extração dos resultados. A escala de pontuação foi definida utilizando a escala Likert[34], ou seja, vai de um a cinco.

Escala:

- 1 = Péssimo

- 2 = Ruim
- 3 = Regular
- 4 = Bom
- 5 = Ótimo

Resultados:

- Média: 3,87
- Mediana: 4
- Moda: 4
- Desvio Padrão: 0,90
- Maior Nota: 5
- Menor Nota: 2

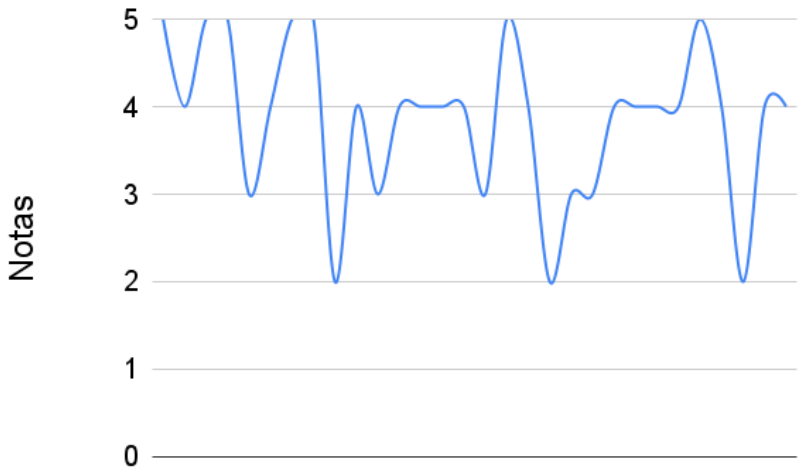

Figura 3.5: Distribuição das notas dadas pelos clientes.

O Telegram é uma ferramenta de comunicação gratuita e conveniente para os prestadores de serviços e clientes, com ela estes podem resolver os seus problemas de forma rápida e simples. Os resultados obtidos com o experimento foram positivos e demonstram a importância de mais pesquisas nessa área de conhecimento.

Estatisticamente a melhor forma de analisar este conjunto de dados é através dos indicadores: mediana e moda[35]. Considerando os valores extraídos, é possível avaliar a usabilidade do *chatbot* e mensurar a sua eficiência. A nota final foi 4, portanto o atendimento do Lu foi classificado como Bom pelos clientes da Pizzaria Smart, a métrica teve um resultado satisfatório.

Capítulo 4

Conclusão

Este trabalho apresenta o desenvolvimento e aplicação de um *chatbot* no serviço de atendimento ao cliente da Pizzaria Smart. Este sistema visa proporcionar uma ferramenta de atendimento automatizado aos clientes. O sistema foi desenvolvido através de estudos sobre usabilidade e teve como base a plataforma Telegram, que fornece funcionalidades que facilitam a implementação de *chatbots*.

Muitos estabelecimentos negligenciam o atendimento ao cliente, devido ao alto nível de dedicação necessário para realizar a tarefa, esta realidade acaba prejudicando o relacionamento da empresa com um dos seus principais stakeholders. Conquistar um bom relacionamento com o público, está se tornando cada vez mais importante com o passar do tempo e deve ser prioridade para qualquer negócio.

Como solução, foi apresentado o Lu, um *chatbot* que tem o objetivo de auxiliar os clientes de um estabelecimento, para que o atendimento se torne uma tarefa menos custosa. Apesar dos resultados positivos, o *chatbot* ainda precisa ser aprimorado para se tornar escalável.

Os trinta clientes da Pizzaria Smart avaliaram o atendimento recebido, a nota final foi 4, que corresponde a Bom. A métrica demonstra o nível de eficiência do Lu e pode ser usada para mensurar se ele está pronto para ser utilizado por usuários reais.

4.1 Trabalhos futuros

Os trabalhos futuros envolvem:

- Adicionar síntese de voz em português na aplicação. Assim além de compreender áudios enviados pelos clientes, o Lu seria capaz de responder no formato de áudio, expandindo os limites da solução. Atualmente existem limitações técnicas para implementar esta funcionalidade, a maioria das bibliotecas suporta apenas o idioma inglês, representando uma barreira para falantes de outros idiomas.

- Replicar o experimento em outros contextos. Um bot similar ao Lu pode ser útil para qualquer categoria de negócio que necessite implementar atendimento ao cliente no formato digital.
- Fluxo conversacional mais completo. O Lu ainda possui um linguajar limitado, aumentar as opções de interação deixaria a solução mais robusta.

Referências

[1] Francesco Colace, Massimo De Santo, Marco Lombardi Francesco Pascale Antonio Pietrosanto: *Chatbot for e-learning: A case of study.* International Journal of Mechanical Engineering and Robotics, 7(5):528–533, 2018. 1, 2

[2] Fadhil, Ahmed: *Can a chatbot determine my diet?: Addressing challenges of chatbot application for meal recommendation.* University of Trento. 1, 6

[3] Hosseini, Samane: *Using a chatbot to increase tourists' engagement.* 1

[4] Nicole Radziwill, Morgan Benton: *Evaluating quality of chatbots and intelligent conversational agents.* 2, 3

[5] Jusbrasil: *Jusbrasil.* `https://www.jusbrasil.com.br/`, acesso em 2021-05-10. 2

[6] S/A, Magazine Luiza: *Magazine luiza.* `https://www.magazineluiza.com.br/`, acesso em 2021-05-10. 2

[7] Nama: *Lu: o chatbot que é o queridinho do público.* `https://www.nama.ai/recursos/cases/magalu`, acesso em 2021-05-10. 2

[8] Dale, Robert: *The return of the chatbots.* Natural Language Engineering, 22(5):811–817, 2016. 2

[9] Cecília Torres, Walter Franklin, Laura Martins: *Accessibility in chatbots: The state of the art in favor of users with visual impairment.* Universidade Federal de Pernambuco. 3, 5

[10] Daiga Deksne, Andrejs Vasiljevs: *Collection of resources and evaluation of customer support chatbot.* Human Language Technologies - The Baltic Perspective, páginas 30–37, 2018. 3

[11] Telegram: *Telegram.* `https://telegram.org/`, acesso em 2021-05-04. 3

[12] Telegram: *Telegram apis.* `https://core.telegram.org/`, acesso em 2021-05-04. 3

[13] Foundation, Python Software: *Python 3.9.5 documentation.* `https://docs.python.org/3/`, acesso em 2021-05-04. 3

[14] Replit: *Replit.* `https://replit.com/`, acesso em 2021-05-04. 3

[15] Google: *Google sheets for developers.* `https://developers.google.com/sheets/api`, acesso em 2021-05-04. 3

[16] Castro, Vinicius Almeida: *Introdução ao desenvolvimento Ágil.* https://www.devmedia.com.br/introducao-ao-desenvolvimento-agil/5916, acesso em 2021-05-10. 7

[17] Kent Beck, James Grenning, Mike Beedle: *Manifesto Ágil.* http://agilemanifesto.org/iso/ptbr/manifesto.html, acesso em 2021-05-10. 7

[18] LLC, WhatsApp: *Whatsapp.* https://www.whatsapp.com/?lang=pt_br, acesso em 2021-05-11. 7

[19] FACEBOOK: *Facebook.* https://about.fb.com/br/company-info/, acesso em 2021-05-11. 7

[20] Ivan M. Tsidylo, Sergiy I. Samborskiy, Stanislav Ivan V. Mazur Maria P. Zamoroz: *Designing a chatbot for learning a subject in a telegram messenger.* Ternopil Volodymyr Hnatyuk National Pedagogical University, 2020. 7, 8

[21] Digital, Olhar: *Telegram amplia base de usuários e já está em 45% dos smartphones brasileiros.* https://olhardigital.com.br/2021/03/05/internet-e-redes-sociais/telegram-base-usuarios-celulares-brasileiros/, acesso em 2021-05-10. 7

[22] Telegram: *Bots: An introduction for developers.* https://core.telegram.org/bots#, acesso em 2021-05-11. 8

[23] GitHub, Inc.: *Github docs.* https://docs.github.com/pt, acesso em 2021-05-10. 8

[24] Inc., Apple: *ios 14.* https://www.apple.com/br/ios/ios-14/, acesso em 2021-05-10. 9

[25] Android: *Apresentamos o android 11.* https://www.android.com/intl/pt-BR_br/, acesso em 2021-05-10. 9

[26] Foundation, Python Software: *python-telegram-bot 13.5.* https://pypi.org/project/python-telegram-bot/, acesso em 2021-05-04. 12

[27] Foundation, Python Software: *Speechrecognition 3.8.1.* https://pypi.org/project/SpeechRecognition/, acesso em 2021-05-04. 12

[28] Foundation, Python Software: *Pyaudio 0.2.11.* https://pypi.org/project/PyAudio/, acesso em 2021-05-04. 12

[29] Red Hat, Inc.: *O que é ide?* https://www.redhat.com/pt-br/topics/middleware/what-is-ide, acesso em 2021-05-10. 13

[30] Red Hat, Inc.: *O que é cloud computing?* https://www.redhat.com/pt-br/topics/cloud, acesso em 2021-05-04. 13

[31] Red Hat, Inc.: *O que é docker?* https://www.redhat.com/pt-br/topics/containers/what-is-docker, acesso em 2021-05-04. 13

[32] Red Hat, Inc.: *Máquina virtual (vm).* https://www.redhat.com/pt-br/topics/virtualization/what-is-a-virtual-machine, acesso em 2021-05-04. 13

[33] Foundation, Python Software: *gspread 3.7.0.* https://pypi.org/project/gspread/, acesso em 2021-05-04. 13

[34] SurveyMonkey: *O que é uma escala likert?* https://pt.surveymonkey.com/mp/likert-scale/, acesso em 2021-05-05. 18

[35] Harry N. Boone, Jr., Deborah A. Boone: *Analyzing likert data.* Journal of Extension, 50(2), 2012. 19