UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUIZ YOKOYAMA FELIX DE SOUZA

Vendedor Virtual

com PLN, Machine Learning e Grafos de Conhecimento

Goiânia
2021

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC nº 1204/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG):

Nome completo do autor: Luiz Yokoyama Felix de Souza

Título do trabalho: Vendedor Virtual com PLN, Machine Learning e Grafos de Conhecimento.

2. Informações de acesso ao documento:

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

Independente da concordância com a disponibilização eletrônica, é imprescindível o envio do(s) arquivo(s) em formato digital PDF do TCCG.

Luiz Yokoyama Felix de Souza
Assinatura do autor[2]

Ciente e de acordo:

Elisângela S. Dias
Assinatura da orientadora[2] Data: 02 / 06 / 2021

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: (a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;

[2] As assinaturas devem ser originais sendo assinadas no próprio documento, imagens coladas não serão aceitas.

LUIZ YOKOYAMA FELIX DE SOUZA

Vendedor Virtual

com PLN, Machine Learning e Grafos de Conhecimento

Trabalho de conclusão de curso apresentado na Escola de Engenharia Elétrica, Mecânica e de Computação como requisito para a conclusão do curso de Engenharia de Computação e obtenção do título de Engenheiro de Computação.

Orientadora: Profa. Dra. Elisângela Silva Dias

Goiânia
2021

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Souza, Luiz Yokoyama Felix de
Vendedor Virtual [manuscrito] : com PLN, Machine Learning e Grafos de Conhecimento / Luiz Yokoyama Felix de Souza. - 2021.
LXXII, 72 f.: il.

Orientador: Profa. Dra. Elisângela Silva Dias.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Engenharia da Computação, Goiânia, 2021.
Bibliografia. Anexos. Apêndice.
Inclui lista de figuras.

1. Chatbot. 2. Rasa. 3. Inteligência artificial. 4. Grafos de conhecimento. I. Dias, Elisângela Silva, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

ATA EMC 23070.023483/2021-25/2021

ATA DE AVALIAÇÃO DE PROJETO FINAL

Curso

( ) Eng Elétrica	( ) Eng Mecânica	(X) Eng Computação PFC 1 ( ) PFC 2 (X)

Título do Trabalho

Vendedor Virtual com PLN, Machine Learning e Grafos de Conhecimento

Banca Avaliadora

Membro 1	Profa. Dra. Elisângela Silva Dias (INF/UFG)
Membro 2	Profa. Dra. Maria Leonor Silva de Almeida (EMC/UFG)
Membro 3	Profa. Dra. Nádia Félix Felipe da Silva (INF/UFG)

Discente

Matrícula	Nome
201602428	LUIZ YOKOYAMA FELIX DE SOUZA

NOTAS										
Matrícula	**Membro 1**			**Membro 2**			**Membro 3**			**Média***
	NPT	**NTE**	**NAA**	**NPT**	**NTE**	**NAA**	**NPT**	**NTE**	**NAA**	
201602428	10,0	10,0	10,0	10,0	10,0	10,0	10,0	10,0	10,0	10,0

NPT – Nota plano de trabalho; NTE – Nota do trabalho escrito; NAA – Nota de apresentação e arguição
Para Eng. Elétrica, Mecânica e PFC2 da Eng. Da Computação: NF = 0,1 x NPT + 0,45xNTE + 0,45 x NAA
Para PFC1 da Eng. Da Computação: NF = 0,3 x NPT + 0,7 x NAA

* A APROVAÇÃO DO(S) ALUNO(S) ESTÁ CONDICIONADA À APRESENTAÇÃO DO TRABALHO FINAL AO ORINETADOR COM TODAS AS CORREÇÕES SUGERIDAS PELA BANCA.

OBSERVAÇÕES:

Preencher com modificações solicitadas, caso existam. Em caso de reprovação, informar a justificativa.

Com base no trabalho final apresentado, a banca decidiu pela aprovação do aluno. No mais, poucas alterações foram solicitadas, as quais foram encaminhadas ao discente, para que ele pudesse realizá-las na versão final do texto.

Documento assinado eletronicamente por **Maria Leonor Silva De Almeida**, **Professora do Magistério Superior**, em 04/06/2021, às 11:29, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Elisângela Silva Dias**, **Professor do Magistério Superior**, em 04/06/2021, às 16:13, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Nadia Felix Felipe Da Silva**, **Professor do Magistério Superior**, em 04/06/2021, às 16:15, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2111398** e o código CRC **67988F8E**.

Luiz Yokoyama Felix de Souza

É graduando em Engenharia de Computação e um ávido interessado por conhecimentos sobre as mais variadas áreas.

Dedico este trabalho a minha família, amigos, colegas e professores.

Agradecimentos

Primeiramente, por tornar este projeto possível, gostaria de agradecer à empresa Multitrem Comércio Digital LTDA (multitrem.com). Sua parceria e suporte foram fundamentais para o desenvolvimento deste projeto.

Agradeço à UFG e aos meus professores, que me ajudaram a trilhar o caminho do conhecimento. Um agradecimento especial a minha orientadora Elisângela Silva Dias, que me acompanhou e orientou neste projeto com muita dedicação e sabedoria.

Agradeço também aos meus colegas e amigos da faculdade, pelos altos e baixos da vida de estudante que vivemos juntos. Como alguns dizem: “A faculdade passa, mas os amigos ficam”.

Por fim, agradeço a minha família, por tudo. Eu não chegaria até aqui sem ela.

"O mundo real é turbulento e exige que as organizações e seus membros passem por mudanças dinâmicas para que continuem competitivos."

ROBBINS, Stephen P.,
Comportamento Organizacional.

Resumo

de Souza, Luiz Yokoyama Felix. **Vendedor Virtual com PLN, Machine Learning e Grafos de Conhecimento**. Goiânia, 2021. 64p. Monografia de Graduação. Curso de Engenharia de Computação, Escola de Engenharia Elétrica, Mecânica e de Computação, Universidade Federal de Goiás.

A era da informação, proporcionada pela internet, está revolucionando as formas como os serviços são oferecidos. Neste cenário, os chatbots, programas de respostas automatizadas em interfaces de diálogo, estão em plena ascensão. Entretanto, eles normalmente só possuem respostas pré-programadas e as opções mais comuns, podendo causar descontentamento no usuário. O objetivo deste trabalho é uma pesquisa aplicada ao projeto de um modelo de chatbot, com inteligência artificial, capaz de interagir na linguagem natural do usuário, emitindo respostas como as de um humano dentro de sua área de atuação, que, neste caso, é a área de vendas. Assim, desenvolveu-se e avaliou-se um modelo de chatbot, com o *framework* Rasa, a partir de novos métodos para a obtenção de dados de treinamento da inteligência artificial. Também se desenvolveu um modelo de Grafo de Conhecimento que pode proporcionar mais qualidade ao atendimento prestado, oferecendo produtos e entendendo detalhes, inclusive os escritos errados das mensagens.

Palavras–chave

Chatbot; Rasa; Inteligência Artificial; Grafos de Conhecimento.

Abstract

de Souza, Luiz Yokoyama Felix. **Virtual Seller with NLP, Machine Learning and Knowledge Graphs**. Goiânia, 2021. 64p. Monografia de Graduação. Curso de Engenharia de Computação, Escola de Engenharia Elétrica, Mecânica e de Computação, Universidade Federal de Goiás.

The information age, provided by the internet, is revolutionizing the ways in which services are offered. In this scenario, chatbots, automated response programs in dialog interfaces, are on the rise. However, it usually only have pre-programmed responses and the most common options, which can cause discontent in the user. The objective of this work is a research applied to the design of a chatbot model, with artificial intelligence, capable of interacting in the user's natural language, emitting responses like those of a human within his area of activity, which, in this case, is the sales area. Thus, a chatbot model was developed and evaluated using the Rasa framework, based on new methods for obtaining artificial intelligence training data. A Knowledge Graph model has also been developed that can provide more quality to the service provided, offering products and understanding details, including the misspellings of messages.

Keywords

Chatbot; Rasa; Artificial Intelligence; Knowledge Graphs.

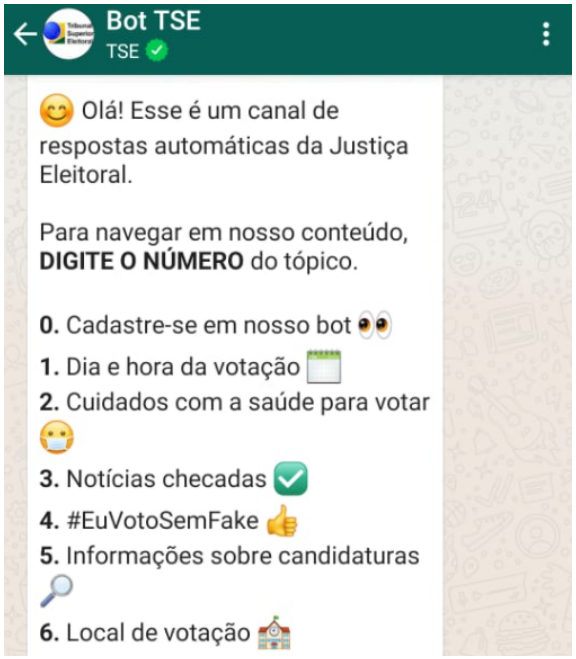
Sumário

Lista de Figuras

CAPÍTULO 1

Introdução

A palavra chatbot é resultado da junção das palavras da língua inglesa *chatter* (a pessoa que conversa) e *bot* (abreviatura de *robot*, robô em inglês). Os chatbots são sistemas computacionais capazes de simular a interatividade de uma conversa entre duas pessoas, assumindo totalmente o papel desempenhado por uma pessoa no diálogo. Mas existem chatbots limitados que podem apenas disponibilizar opções para o usuário escolher, seguindo um fluxo de opções pré-definido, como pode ser visto na Figura 1.1. Já os chatbots mais avançados podem usar técnicas de inteligência artificial para interagir em linguagem natural com o usuário, por texto e até por voz, e por isso podem ser chamados de humanizados.



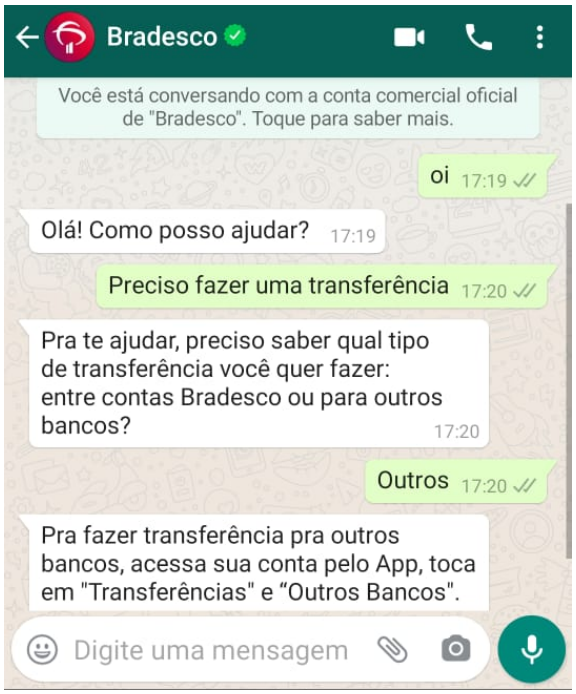
Fonte: TSE.

Figura 1.1: *Chatbot do TSE oferecendo opções numéricas.*

Conforme Ainley et al. [4], os chatbots representam softwares com capacidade limitada, mas crescente, de conversar com seres humanos. Então, as pesquisas para

expandir as capacidades de argumentação dos chatbots vem ganhando destaque nos trabalhos de inteligência artificial.

As pessoas estão cada vez mais conectadas pela internet e muitas já acham mais fácil e cômodo resolver tudo pela internet. Segundo Farreras et al. [17], os chatbots vêm despertando interesse crescente nos usuários, com muitas pessoas desejando ter extensas interações com eles. Neste sentido, os chatbots vêm sendo cada vez mais empregados na interação com usuários de plataformas digitais e no atendimento de clientes, como pode ser visto na Figura 1.2. Muitas empresas estão tentando reduzir os custos com mão de obra substituindo o atendimento humano por chatbots. Mas é certo que existem *trade-offs* nesta prática. *Trade-off* é uma expressão muito usada por economistas e significa que pode trazer benefícios e prejuízos. Neste caso, pode tanto melhorar a qualidade do atendimento ao cliente, quando reduz ou evita uma fila de espera por atendimento, como pode também piorar a qualidade do atendimento, com o cliente frustrado por não gostar da interatividade do chatbot, que ainda possui limitações de capacidade de entendimento e de fornecer respostas que possam atender às expectativas do cliente.



Fonte: Banco Bradesco.

Figura 1.2: *Diálogo com o chatbot do Bradesco.*

Ainda conforme Farreras et al. [17], apesar dos chatbots conseguirem interagir e manter a atenção dos usuários, eles ainda não têm capacidade de simular todo o leque de conversação da inteligência humana. Neste sentido, humanizar chatbots significa melhorar sua capacidade de simular uma boa conversa entre pessoas, mesmo que limitado a determinado assunto ou área de conhecimento. Assim, essa ferramenta pode se tornar

uma forma de aumentar a satisfação do cliente da empresa ou do usuário do sistema. Para isso, o *bot* (programa robô) deve ser capaz de compreender o que a pessoa disse ou escreveu e, então, ser capaz de responder como uma pessoa responderia. Assim, com o chatbot sendo capaz de interagir como um humano, ele poderia ser empregado, por exemplo, para melhorar a qualidade do serviço prestado por uma empresa, já que um programa pode obter informações em uma base de dados de maneira muito mais rápida que um humano consultando via teclado e mouse. Então, o que antes seria geralmente empregado para o corte de custos com mão de obra, como nos atendimentos de *callcenter* de companhias telefônicas, passaria a ser um diferencial na qualidade do serviço prestado ao cliente.

Fonte: Detran-GO.

Figura 1.3: *Chatbot do Detran-GO oferecendo opções fixas.*

Os chatbots vieram para ficar, pois estão sendo implementados para os mais diversos fins, pelos mais diversos proponentes. Neste projeto, será usado um *framework open source* para a criação de um modelo de chatbot que, diferentemente dos chatbots que só oferecem opções para selecionar, como na Figura 1.3, possa ter um diálogo mais assemelhado com o que um humano real teria. O modelo a ser desenvolvido tentará responder às pessoas consultando a uma base de conhecimento e retornando respostas em linguagem natural, para que os usuários tenham a impressão de estar conversando com outra pessoa. Então, o chatbot é avaliado quanto à sua humanização.

Para tanto, este trabalho possui estudos para embasamento teórico, levantamento de requisitos do local de aplicação e definição dos dados conversacionais para o aprendizado da inteligência artificial do chatbot, além de testes em protótipos. Assim, obtendo-se o modelo de chatbot com grafos de conhecimento e o framework de chatbots com *Machine Learnig* e PLN, a ser implementado e, então, avaliado durante o seu próprio desenvolvimento.

CAPÍTULO 2

Fundamentação Teórica

2.1 Machine Learning

Machine Learning (ML) ou Aprendizado de Máquina (AM) é a área da Inteligência Artificial focada na análise de padrões, em uma base de dados, para a criação automatizada de regras de reconhecimento ou replicação destes padrões. Conforme Faceli et al. [16], ML é um processo de indução de hipótese a partir da experiência. Assim, algoritmos de ML deduzem uma solução para um problema a partir de exemplos do problema resolvido. Quanto maior a base de dados de exemplos, melhor a aprendizagem e melhor o modelo de reconhecimento e resposta gerado. Sendo capaz, inclusive, de lidar com situações não apresentadas durante seu aprendizado [16].

Mas o uso do aprendizado de máquina pode e vai além. Para Norvig e Russell [28], aprendendo com uma coleção de entradas e saídas, uma função é criada com capacidade para prever as saídas decorrentes de novas entradas. Nota-se, então, que o ML é uma área importante que envolve não só a IA, como também a Probabilidade e Estatística, bem como a Teoria da Computação, a Teoria da Informação e a Neurociência [16]. Possibilitando, então, a capacidade de criar e organizar novos conhecimentos em sistemas computacionais.

Os algoritmos de aprendizagem de máquina dependem de conjuntos de dados para treinamento. Segundo Coppin [13], na maioria dos problemas de aprendizado, a tarefa é aprender a classificar as entradas de acordo com um conjunto de classificações. Para isso, é fornecido um conjunto de dados de treinamento, previamente classificados, para o sistema de aprendizagem. O sistema, então, tenta aprender a partir destes dados de treinamento como classificar os mesmos dados e também como classificar novos dados que não foram vistos, pressupondo que há alguma relação entre os dados e as classificações.

Na verdade, o aprendizado de máquina é um nome genérico para quando são empregados algoritmos de classificação automática que criam, por exemplo, árvores de decisão, *Support Vector Machines* (SVM), redes neurais e redes neurais profundas, dentre outras técnicas de inteligência artificial.

2.2 Processamento de Linguagens Naturais

De acordo com Coppin [13], linguagens naturais são as usadas por humanos para comunicação entre humanos, dentre outras funções. Então, por linguagem natural, entende-se qualquer linguagem humana que tenha surgido naturalmente. Diferentemente de linguagens formais como, por exemplo, C, Java e Python, as linguagens naturais apresentam ambiguidades e não são estruturadas. Portanto, são extremamente complexas de serem entendidas por um sistema computacional, visto que um sistema computacional precisa de uma sequência de operações lógicas bem definidas para realizar alguma ação.

Ainda conforme Coppin [13], o Processamento de Linguagem Natural (PLN) é um conjunto de técnicas para permitir que um computador possa "entender" a linguagem humana. Neste sentido, o Processamento de Linguagem Natural é a área computacional sobre a interpretação e reprodução de linguagens naturais, tanto em suas formas escritas, como faladas. Para Bird et al. [8] um objetivo importante da pesquisa em PLN tem sido progredir na difícil tarefa de construir tecnologias que entendam a linguagem, cobrindo qualquer tipo de manipulação de linguagem natural por computador.

O processamento de linguagem natural normalmente é implementado com redes neurais. Nestas redes neurais, os modelos de aprendizado profundo não tomam como entrada o texto bruto, pois elas funcionam apenas com vetores numéricos. Os processos de vetorização de texto consistem em aplicar uma tokenização como, por exemplo, separar palavras ou mesmo caracteres e, em seguida, associar vetores numéricos aos tokens. Estes vetores são, então, aplicados no aprendizado de máquina com redes neurais profundas.

Assim, o aprendizado de máquina aplicado ao processamento de linguagem natural significa o reconhecimento de padrões aplicados as palavras, frases e parágrafos. Ainda hoje, nenhum desses modelos de aprendizagem profunda realmente entende o texto semanticamente como uma pessoa entenderia. Na verdade, esses modelos podem mapear uma estrutura estatística da linguagem. Fato que é suficiente para resolver muitas tarefas simples.

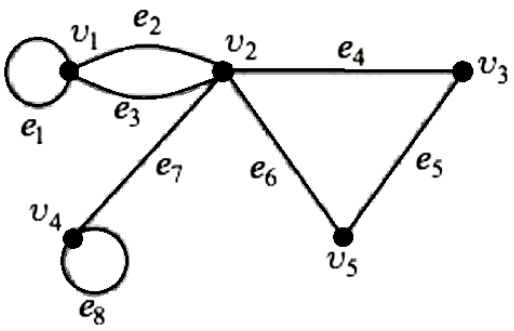
Já os humanos, por conhecerem a semântica de sua linguagem, geralmente podem executar uma nova tarefa de linguagem com apenas alguns exemplos. Porém já existem projetos bancados por grandes empresas, que prometem ao chatbot uma capacidade de diálogo como a de uma pessoa. Para isso, além de uma quantidade massiva de dados de treinamento, também são empregados dados de uma rede semântica de conhecimento para relacionar as palavras. Um exemplo é o Goolgle Meena [3], um modelo de conversação avançado que foi treinado com mais 2,6 bilhões de parâmetros neurais.

2.3 Teoria dos Grafos

> Um grafo G=(V, E) consiste em V, um conjunto não vazio de vértices (ou nós), e E, um conjunto de arestas. Cada aresta tem um ou dois vértices associados a ela, chamados de suas extremidades. Dizemos que uma aresta liga ou conecta suas extremidades. Rosen [34]

Os conceitos e explicações da teoria dos grafos que se seguem são todas baseadas do livro de Rosen [34]. Segundo este autor, os grafos são estruturas discretas que consistem em vértices e arestas que ligam estes vértices. O conjunto de vértices de um grafo pode ser, teoricamente, infinito. Diversos problemas podem ser resolvidos usando modelos de grafos, já que algoritmos computacionais podem percorrer os caminhos formados pelos vértices e arestas.

Dentre suas muitas características, destaca-se que os grafos podem ser classificados como direcionados ou não, quando suas arestas têm um sentido específico ou não. Também podem ser cíclicos ou acíclicos, quando os sentidos das arestas não permitem voltar ao mesmo vértice. Uma forma de se representar um grafo sem arestas múltiplas, que são arestas que possuem os mesmos vértices como extremidade, é usar listas ou matrizes de adjacência. Nelas se especificam os vértices que são adjacentes a outro vértice do grafo. Já as matrizes de incidência também podem ser usadas para representar arestas múltiplas e laços, como pode ser visto na Figura 2.1.



	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
v_1	1	1	1	0	0	0	0	0
v_2	0	1	1	1	0	1	1	0
v_3	0	0	0	1	1	0	0	0
v_4	0	0	0	0	0	0	1	1
v_5	0	0	0	0	1	1	0	0

Fonte: Rosen [34].

Figura 2.1: *Exemplo de grafo e sua matriz de incidência.*

Estas arestas múltiplas são representadas na matriz de incidência usando colunas com elementos idênticos. Representando, assim, que são incidentes ao mesmo par de vértices. Neste caso, os laços usam uma coluna com exatamente um elemento igual a 1, correspondendo ao vértice que é incidente deste laço.

Os grafos também podem ser usados para modelar as relações do mundo real, tais como processos em sistemas físicos, biológicos, sociais e de informações. Neste sentido, os grafos são uma estrutura de dados bastante versátil e vêm sendo empregados com sucesso na área da inteligência artificial. Ainda conforme Rosen [34], a própria internet pode ser modelada como um grafo orientado contendo vários bilhões de vértices e arestas.

Neste modelo dinâmico e em constante crescimento, cada página da Web é um vértice e as arestas são os links entre as páginas.

2.4 Grafos de Conhecimento

A inovação batizada como Grafo de Conhecimento foi lançada pelo Google [37], no final de 2012, utilizando inteligência artificial para fornecer respostas melhores e mais completas nas buscas. De acordo com Paulheim [29], de uma perspectiva mais ampla, qualquer representação baseada em grafo sobre algum conhecimento pode ser considerado um grafo de conhecimento.

Conforme o Google [37], o grafo de conhecimento de sua plataforma de busca fornece resultados de pesquisa mais inteligentes com informações mais relevantes aos usuários, na forma de uma caixa de informações perto dos resultados de busca, como pode ser visto na Figura 2.2(a). Nesta figura, aparece uma caixa de informações, provida pelo grafo de conhecimento do Google, com informações retiradas do site da UFG.

(a) *Busca por "UFG".* (b) *Busca por "teoria dos grafos".*

Fonte: Google [20] [21].

Figura 2.2: *Exemplos de caixas de informações no resultado de busca.*

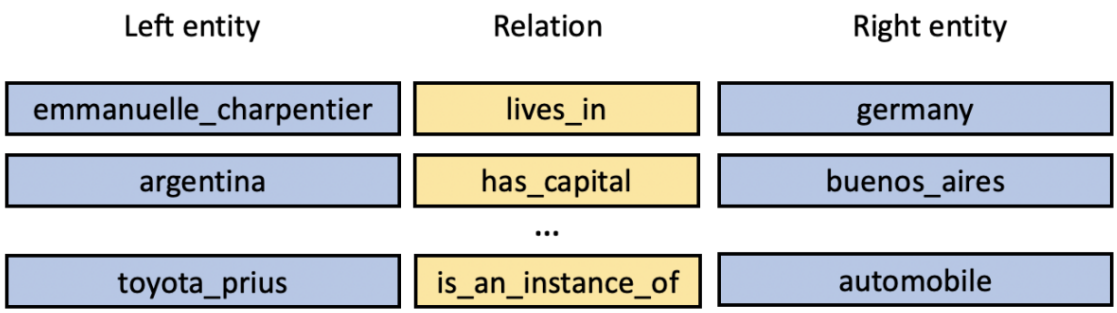
Já na Figura 2.2(b), é possível ver a caixa de informações que aparece ao se pesquisar no Google por "teoria dos grafos". Nesta caixa de informações é possível notar

que além de um resumo, obtido de uma fonte confiável, também são mostrados links para assuntos semelhantes que podem ser úteis ou que também costumam ser pesquisados junto à "teoria dos grafos". Estas informações são fruto de algoritmos de classificação automática, bem como do *feedback* dos próprios usuários.

Conforme informa Paulheim [29], em seu artigo sobre avaliação de grafos de conhecimento, o termo Grafo de Conhecimento foi cunhado pelo Google no lançamento de sua referida inovação. Segundo Monteiro [27], quase nada tem sido escrito sobre Grafos de Conhecimento na literatura científica por ser uma tecnologia privada e por ser uma inovação recente. Supõe-se que grande parte de seu "conhecimento" provém de fontes da web semiestruturada, como a Wikipedia, que contribui para o grafo de conhecimento do Google, bem como marcações estruturadas em páginas da web e conteúdos da rede social do Google.

Mas apesar de pouco se saber sobre o que está por trás dessa tecnologia e que mecanismos são empregados, o termo grafo de conhecimento se tornou popular. Segundo Ehrlinger e Wolfram [15], falta uma definição comum que permita se fazer afirmações precisas sobre a introdução e disseminação dos grafos de conhecimento. Surgiram, então, uma série de propostas de modelos e implementações que uniam grafos e conhecimento, aproveitando o mesmo nome.

Segundo Fisher [18], os sistemas de resposta a perguntas frequentemente dependem de grafos de conhecimento com grandes coleções de fatos sobre entidades do mundo real como pessoas, organizações, países, etc. Ainda conforme este autor, um grafo de conhecimento padrão pode representar os dados usando triplas formadas por dois vértices e uma relação entre eles. Nesta tripla, cada vértice representa uma entidade que possui algum tipo de relação com a entidade do outro vértice, como pode ser visto no exemplo da Figura 2.3. Nesta figura, as entidades "emmanuelle_charpentier" e "germany" estão relacionadas pela relação "lives_in".



Fonte: Fisher [18].

Figura 2.3: *Triplas do grafo de conhecimento consistindo em uma entidade à esquerda, uma relação e uma entidade à direita.*

Sistemas de recomendação também podem ser modelados como grafos conhecimento e são amplamente empregados desde a recomendação de vídeos, notícias e artigos, até a recomendação de pessoas nas redes sociais. No comércio, por exemplo, as entidades são os produtos e a relação entre eles é o fato deles serem comprados juntos.

2.5 Linguagem Python

> Python é uma linguagem de programação poderosa e de fácil aprendizado. Ela já possui nativamente estruturas de dados de alto-nível poderosas e eficientes, como por exemplo o Dicionário, bem como adota uma abordagem de escrita simples e efetiva que evita contratempos ao programador. Sua sintaxe elegante e tipagem dinâmica, em adição à sua natureza interpretada, tornam Python ideal para scripting e para o desenvolvimento rápido de aplicações em diversas áreas e na maioria das plataformas. [10]

A linguagem Python possibilita tanto orientação a objeto, quanto o paradigma procedimental. Então, pode ser usada para programas simples e rápidos, com estruturas de dados complexas nativas como listas e dicionários. Tudo aliado a amplas bibliotecas de código aberto que são constantemente desenvolvidas e aprimoradas.

2.5.1 JSON

Conforme o site json.org [22], JSON (*JavaScript Object Notation* - Notação de Objetos JavaScript), é um formato simples para troca de dados pela internet derivado da linguagem JavaScript, mas é independente da linguagem de programação. Neste sentido, JSON é uma forma de codificar estruturas de dados complexas, como listas e dicionários, em strings de texto simples que podem ser facilmente transferidas pela internet e depois remontadas novamente como as estruturas que eram antes de serem transferidas. O JSON se assemelha com um dicionário do Python, formado por pares de chave e valor.

2.5.2 API REST

Conforme Massé [25], programas clientes usam APIs (*Application Programming Interface* – Interface de programação para aplicativos) para se comunicarem com serviços web. REST (*Representational State Transfer* - Transferência de Estado Representacional) é um modelo de arquitetura comumente utilizado para desenvolver APIs de serviços web modernos, permitindo, então, a troca de informações entre aplicações clientes e serviços web.

Para o uso de serviços via REST, os dados devem ser transferidos no formato JSON. Na linguagem de programação Python, usa-se a biblioteca *requests* para fazer

requisições REST. Conforme Kurose [23], para se realizar uma requisição a um servidor web remoto e recuperar dados, usa-se solicitações HTTP como GET, PUT, PATCH, POST e DELETE. O servidor, então, responde com os dados solicitados no formato JSON e códigos de status HTTP padrões como, por exemplo, 200 para OK e 400 para solicitações inválidas.

2.5.3 YAML

Conforme o site yaml.org [38], a linguagem YAML implementa uma serialização de dados, possibilitando uma melhor legibilidade humana do que outras linguagens. Neste sentido, esta é uma linguagem ideal para implementar uma padronização dos dados de treinamento para o aprendizado de máquina.

2.5.4 TensorFlow

Conforme o site oficial tensorflow.org [2], o TensorFlow é uma interface para expressar algoritmos de aprendizado de máquina e uma implementação para executar esses algoritmos. Neste sentido, o TensorFlow é muito versátil e pode ser usado para expressar uma ampla variedade de algoritmos, tais como algoritmos de treinamento e inferência de rede neural profunda. O TensorFlow tem sido usado para conduzir pesquisas e implantar sistemas de ML em muitas áreas de ciência da computação e outros campos, tais como reconhecimento de fala, visão computacional, robótica, recuperação de informações, processamento de linguagem natural, extração de informações geográficas, dentre outros.

Ainda segundo o site TensorFlow.org [2], um cálculo expresso com o TensorFlow pode ser executado com pouca ou nenhuma alteração em uma ampla variedade de sistemas heterogêneos, desde dispositivos móveis, como telefones e tablets, até sistemas distribuídos em grande escala com centenas de máquinas e milhares de dispositivos computacionais, como placas de GPU.

2.5.5 Sklearn

A API Sklearn [11], disponível para a linguagem Python, integra algoritmos clássicos de aprendizado de máquina. Sendo, então, comumente empregado para PNL e diversos outros campos de pesquisa. O projeto scikit-learn, do qual provem a biblioteca de código aberto Sklearn, é desenvolvido por uma equipe internacional de desenvolvedores centrais, sendo a maioria pesquisadores dos mais variados campos de conhecimento.

2.6 Design de interação

Segundo Preece, Rogers e Sharp [30], quanto mais se observa a maneira como as pessoas interagem e se comportam em relação a dispositivos interativos, mais se percebe o quão estranho pode ser seu comportamento. Isto porque, as pessoas podem não saber o que devem fazer ou os dispositivos podem não funcionar de maneira adequada. Neste sentido, existe a necessidade de se desenvolver produtos adaptados aos usuários ao invés dos usuários terem que se adaptar aos produtos. Assim, o design de interação engloba o desenvolvimento de produtos interativos que são fáceis, eficazes e agradáveis de usar, levando em consideração a perspectiva dos usuários.

Então, conforme Preece, Rogers e Sharp [30], a experiência do usuário deve ser aprimorada com a participação do próprio usuário, realizando testes ou simplesmente ouvindo sua opinião durante o desenvolvimento do produto. Buscando, assim, reduzir os aspectos negativos desta interação como, por exemplo, frustações e aborrecimentos. Além de realçar os aspectos positivos como, por exemplo, satisfação e eficácia.

2.7 Frameworks de Chatbots com IA

Segundo Beltrametti et al. [7], a Inteligência Artificial (IA) oferece a oportunidade de melhorar e multiplicar as possibilidades de ação humana. Neste sentido, um chatbot com tecnologia de IA é um tipo de bot capaz de interpretar o que a pessoa quis dizer e, então, fornecer comunicação em um nível mais natural e próximo ao humano. Assim, é capaz de responder a perguntas mais elaboradas de forma inteligente.

Existem muitas plataformas de criação de chatbots disponíveis, tanto pagas como gratuitas. Algumas plataformas possuem interface gráfica e não precisam nem de codificação. Já os *frameworks* são modelos de fluxos com códigos pra agilizar o desenvolvimento de software. Conforme Allen et al. [5], os *frameworks* e suas bibliotecas podem abstrair as diferenças de plataformas, tornando possível o desenvolvimento de um aplicativo para ser executado em várias plataformas.

O Wit.ai é um projeto de código aberto com ferramentas abertas. Com ele os desenvolvedores podem facilmente criar chatbots com capacidade de entendimento humano. Como foi adquirido pelo Facebook, é fácil de implantar no Facebook Messenger.

Já o ChatterBot é um mecanismo de diálogo de conversação, baseado em aprendizado de máquina, desenvolvido em Python para gerar respostas com base em um banco de conversas existentes em qualquer idioma.

Rasa é um *framework open source* que vem se destacando por suas capacidades de IA e sua flexibilidade de integração com muitas plataformas de comunicação populares como Facebook Messenger, Telegram e Twilio (intermediário do WhatsApp).

CAPÍTULO 3

Framework Rasa Open Source

As informações técnicas e análises sobre o framework Rasa contidas neste projeto são todas baseadas nas informações disponibilizadas pela Rasa Technologies [32]. O Rasa se destaca por ser um modelo de chatbot orientado à conversação. Isso significa que ele não usa um algoritmo de fluxo estático para o usuário seguir, como os modelos já comuns. Seu modelo de IA dá liberdade para o usuário interagir livremente. Para isso, ele utiliza compreensão da linguagem natural (do acrônimo em inglês NLU – *Natural Language Understanding*), gerenciamento de diálogo e interações, permitindo vários ambientes para desenvolvimento, teste e produção. Esse *framework* de chatbot com IA também facilita o desenvolvimento por meio do aprendizado interativo, possibilitando que ele vá aprendendo enquanto interage com as pessoas durante o seu desenvolvimento.

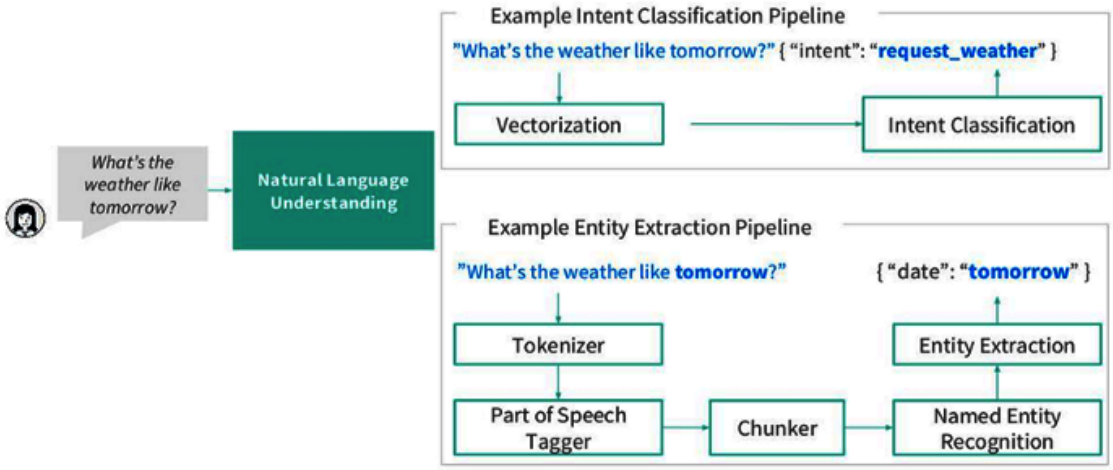
O Rasa Open Source integra as APIs Sklearn[11] e TensorFlow [2] com a linguagem Python para realizar aprendizado de máquina e PLN. Podendo, inclusive, utilizar o poder de processamento de GPUs. O uso da linguagem de programação Python no Rasa permite que o chatbot possa executar tarefas complexas com a agilidade de desenvolvimento que o Python proporciona e permite aos desenvolvedores criar um chatbot com os recursos desejados. Segundo Bocklisch [9], embora o código seja implementado em Python, o Rasa pode usar APIs HTTP para ser facilmente usado por projetos que usam outras linguagens de programação.

3.1 Arquitetura interna do chatbot Rasa

Até as versões 1.x, a arquitetura do framework Rasa era formada por três módulos principais de funcionalidades específicas, pelos quais transita o fluxo de informações. Estes três principais componentes do Rasa são o Rasa NLU *(Natural Language Understanding)*, o Rasa Core e o Rasa NLG *(Natural Language Generation*), que é o módulo responsável pelas mensagens em linguagem natural enviadas ao usuário [32].

O Rasa Core ajuda a criar chatbots conversacionais inteligentes escolhendo a resposta ou ação correta a ser tomada durante um diálogo. Assim, uma ação é chamada quando é necessário consultar um serviço externo para emitir a resposta. Já o módulo NLU

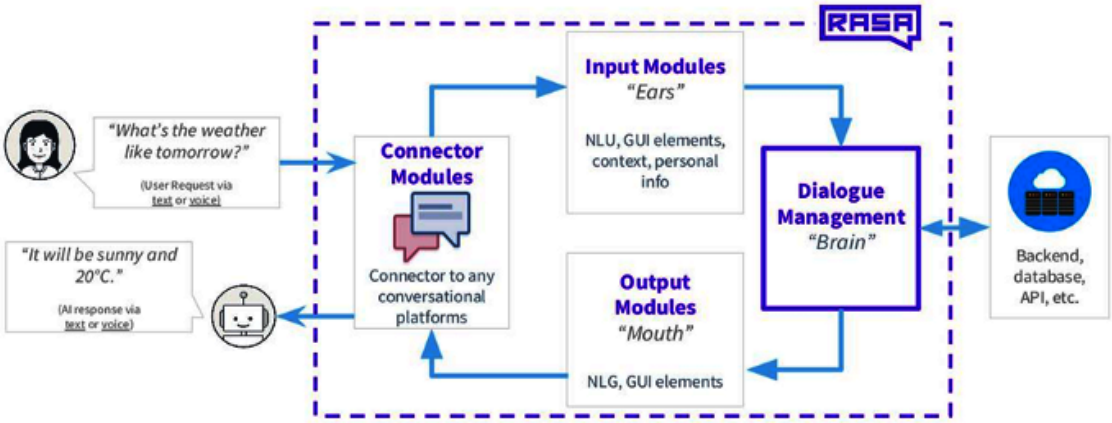
é responsável pela classificação das intenções e pela extração das entidades presentes nas mensagens do usuário. O seu fluxo interno é mostrado na Figura 3.1.



Fonte: Rasa Technologies [32].

Figura 3.1: *Rasa NLU.*

Estas *intents* (intenções) são as classificações que as mensagens do usuário podem receber. Durante a fase de treinamento da rede neural são usados exemplos já classificados com suas devidas intenções, na forma de dados de treinamento para o módulo NLU do chatbot. Assim, o Rasa NLU é o responsável pelo processamento da linguagem natural. Realizando, então, a classificação de *intents* (intenções) e extração de *entities* (entidades). Já estas entidades são os dados estruturados que podem ser obtidos na sentença. Por conseguinte, intenções e entidades são as informações que ajudam o chatbot a entender o que, especificamente, um usuário está querendo.

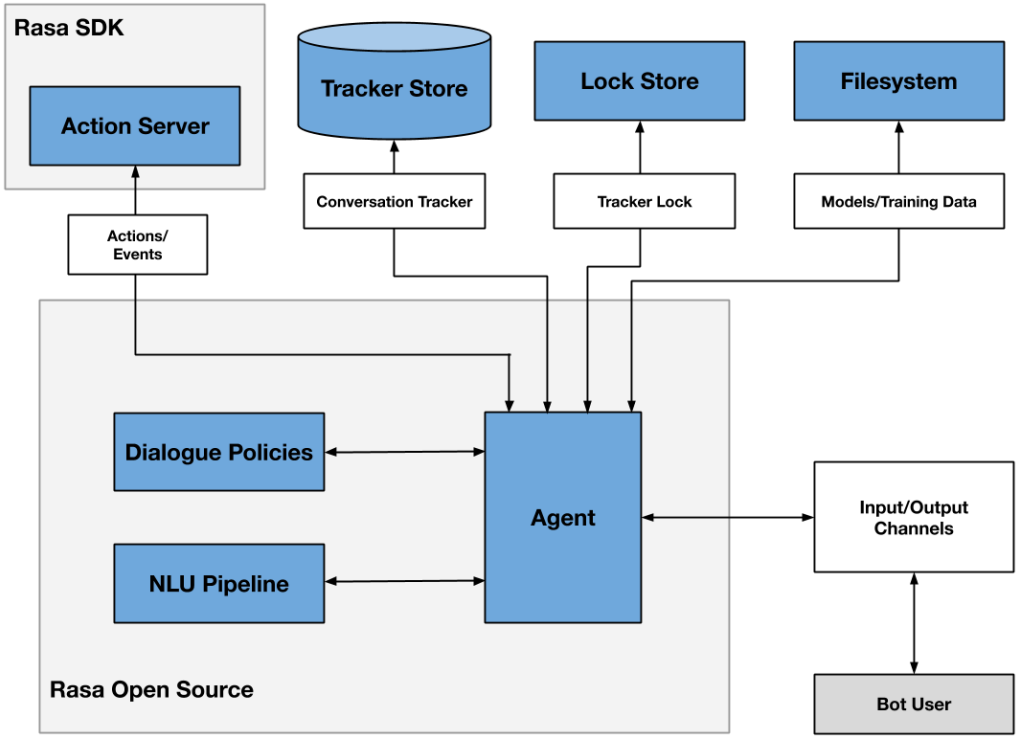


Fonte: Rasa Technologies [32].

Figura 3.2: *Fluxo de construção do diálogo.*

Já o fluxo de construção do diálogo entre o chatbot e o usuário pode ser visto na Figura 3.2. Neste fluxo, depois de passar pelo módulo de conexão com alguma interface de

usuário, a mensagem do usuário passa pelo NLU, que realiza a classificação da intenção e a extração de entidades. Caso esta intenção solicite dados externos ao chatbot, o Rasa Core chama a ação correspondente. Esta ação pode executar qualquer código Python, inclusive chamadas a alguma API para receber informações de fontes externas como, por exemplo, consultar se existe estoque do produto. Então, a resposta passa para o módulo NLG, que se encarrega de emitir uma mensagem em linguagem natural. Esta mensagem, então, vai para o módulo de conexão com a interface de usuário, que se encarregará de apresentar a resposta ao usuário.



Fonte: Rasa Technologies [32].

Figura 3.3: *Arquitetura Rasa 2.1.*

O Rasa Open Source possui uma arquitetura modular e escalável. Isso significa que cada módulo é independente e pode ser substituído ou usado separadamente, além de se poder adicionar outros módulos. Por exemplo, pode-se usar o módulo NLU para realizar PLN em uma outra aplicação ou ainda, criar seu próprio módulo NLG. Agora com a mudança para as versões 2.x, novos módulos foram incorporados e outros modificados.

O diagrama da Figura 3.3 fornece uma visão geral da arquitetura do Rasa 2.1. O Rasa NLU incorporou a recuperação de resposta. Já o Gerenciamento de Diálogo englobou o Rasa Core e é apresentado no diagrama como *Dialogue Policies*. Este Gerenciamento de Diálogo é o módulo que decide a próxima ação em uma conversa com base no contexto e nas políticas de decisão definidas.

O módulo Tracker Store permite escolher gravar os históricos das conversas em um banco de dados tradicional ou ainda criar seu próprio arquivo de conversas. Já o módulo Lock Store permite que vários servidores Rasa possam ser executados em paralelo como serviços replicados.

3.2 Ações

Para especificar as ações executadas pelo chatbot usa-se a palavra-chave *action*, seguida de dois pontos e o nome da ação: *action: nome_da_ação*. O *framework* Rasa possibilita quatro tipos de ações: *responses*, *custom actions*, *forms* e *default actions*.

As *responses* (respostas) somente enviam uma mensagem para o usuário e seu nome sempre começa com *utter_* nas *stories*. As respostas podem conter botões para o usuário clicar e até imagens. Embora sejam recursos que variam com a platarforma utilizada como interface entre o cliente e o chatbot, o framework Rasa permite especificar diferentes respostas de acordo com a interface que o cliente esteja se comunicando.

Os Rasa *forms* (formulários) são um tipo especial de ação, projetada para lidar com a lógica de negócios. São usados nos projetos de conversação em que o chatbot precisa solicitar um conjunto específico de informações, permitindo que o chatbot vá solicitando as informações, preenchendo os slots requeridos até que todos os slots necessários sejam preenchidos.

Já as *custom actions* (ações personalizadas) podem executar qualquer código Python, incluindo suas APIs e consultas a banco de dados, além de poder enviar qualquer número de mensagens. Assim, são utilizadas para responder às requisições do cliente que exijam informações contidas em bases de dados externas e processamentos extras.

Para executar uma *custom action*, o Rasa Core envia uma solicitação ao *Action Server* (servidor de ação) para executar a ação personalizada. Como uma resposta à chamada de ação do Rasa Core, o *Action Server* pode, por exemplo, avaliar os dados fornecidos pelo usuário e enviar, de volta para o usuário, respostas de acordo com estes dados.

O *Action Server* Rasa possui APIs para executar as ações personalizadas escritas na linguagem Python. Assim, o uso do *Action Server*, permite ao desenvolvedor se concentrar apenas na lógica de negócios definida através das *custom actions*. Também é possível usar um módulo servidor de ações desenvolvido por terceiros, provendo suporte a *custom actions* escritas em outras linguagens de programação.

As *default actions* (ações padrão) são respostas incorporadas ao gerenciador de diálogos por padrão. A maioria delas são previstas automaticamente com base em certas situações de conversa como, por exemplo, a *action_listen* que é prevista para sinalizar que o chatbot não deve fazer nada e aguardar a próxima entrada do usuário.

3.3 Extração de entidades e classificação de intenções

O Rasa Open Source possibilita a seleção de integração com diversos componentes para a classificação de intenções e extração de entidades. Desde módulos desenvolvidos pelo próprio Rasa, até componentes de terceiros, cada um tem funcionalidades e características específicas e atende a diferentes demandas.

O DIETClassifier (*Dual Intent and Entity Transformer Classifier*) é um módulo Rasa que realiza a classificação de intenções e o reconhecimento de entidades a partir do que foi aprendido com os dados de treinamento. Também existem módulos extratores de entidades já treinados para determinadas entidades como, por exemplo, o Duckling e o Spacy.

Para a extração de entidades, o DIETClassifier usa uma função de marcação de campo aleatório condicional (do acrônimo em inglês *CRF - Conditional Random Fields*) diretamente na sequência dos tokens de entradas do usuário, para fazer o reconhecimento de entidade nomeada (do acrônimo em inglês *NER - Named Entity Recognition*). Então, no momento da inferência, ele percorre todas as palavras de uma frase e classifica se uma ou mais delas pertencem a uma entidade. Neste processo de classificação o algoritmo analisa, além da palavra sendo avaliada, a palavra que a precede e a palavra que a sucede. Assim, a partir de muitos exemplos de treinamento, o CRF também pode generalizar para dados ainda desconhecidos.

Já para a classificação de intenções, os tokens são vetorizados e os vetores são incorporados em um único espaço vetorial semântico. O DIETClassifier usa uma função de perda do produto escalar para maximizar a semelhança com a intenção alvo e minimizar as semelhanças com amostras negativas. Obtendo, assim, um tipo de grau de certeza quanto ao pertencimento da frase ao conjunto representado por uma intenção.

Também é possível adicionar expressões regulares e tabelas de pesquisa aos dados de treinamento para ajudar o chatbot a identificar *intents* e *entities* corretamente. Estas tabelas de pesquisa são listas de palavras que geram padrões em expressão regular sem diferenciar maiúsculas e minúsculas. Depois a tabela de pesquisa inserida nos dados de treinamento é combinada em uma grande expressão regular. Assim, cada exemplo de treinamento pode ser verificado para ver se contém correspondências nas entradas da tabela de pesquisa.

3.4 Dados de treinamento para a IA Rasa

Os dados de treinamento no *framework* Rasa são salvos na linguagem YAML [38], que possibilita uma forma unificada e extensível de gerenciar todos os dados de treinamento, incluindo dados NLU, histórias e regras. Assim, os dados de treinamento

podem ser divididos em qualquer número de arquivos YAML. Cada arquivo pode conter uma ou mais *keys* (chaves), que indicam o tipo do dado de treinamento, seguida pelos seus dados de treinamento correspondentes. A Figura 3.4 mostra um pequeno exemplo que mantém alguns tipos de dados de treinamento diferentes em um único arquivo. Ressalta-se que um chatbot bem treinado precisa de uma grande quantidade de dados de treinamento.

```
nlu:
- intent: cumprimentar
  examples: |
    - Olá
    - Oi
    - Olá [Luiz](nome)

- intent: faq/idioma
  examples: |
    - Você fala quais idiomas?

stories:
- story: saber local
  steps:
  - intent: cumprimentar
  - action: utter_ola
  - intent: faq/local
  - action: utter_faq/local
```

Fonte: elaborada pelo autor.

Figura 3.4: *Treinamento do chatbot em YAML.*

Os dados de treinamento do Rasa NLU consistem em exemplos de expressões do usuário categorizadas por *intent*. Os dados de treinamento também podem incluir *entities* (entidades), que são informações estruturadas que podem ser extraídas dos diálogos com os usuários. No arquivo que define o domínio dos dados de treinamento, todas as classificações possíveis para as mensagens do usuário são especificadas com a palavra-chave *intent*. Já a palavra-chave *entities* é opcional para o caso de ser preciso extrair algum dado estruturado.

Um exemplo da entidade "nome" também pode ser visto na Figura 3.4. Assim, o chatbot pode entender o que, especificamente, o usuário está perguntando, extraindo dados estruturados da sentença como, por exemplo, um nome, uma cor ou uma quantidade.

3.4.1 Mensagens do usuário

Nos exemplos de mensagens do usuário fornecidos nos dados de treinamento, é preciso especificar para o chatbot como e onde encontrar as entidades. Para fazer isso, as entidades são marcadas e nomeadas nas expressões do usuário fornecidas como exemplos de intenções. Para aumentar a precisão da extração de entidades é necessário adicionar

vários exemplos de frases com essas entidades. O objetivo é dar exemplos com variedade suficiente para que o chatbot possa aprender a generalizar para expressões que não estão em seus dados de treinamento e, assim, comece a entender frases que nunca viu antes.

Para fins de identificação da intenção, é necessário fornecer uma ampla variedade de frases nos exemplos de intenção, e não apenas repetir a mesma frase com uma variação do valor da entidade. Embora os exemplos repetidos possam, em alguns casos, reforçar o grau de certeza atribuído pelo modelo treinado à intenção de uma frase, quanto mais variedade de palavras e sentenças, melhor o chatbot se sairá quando submetido a uma situação de uso real.

Já para a extração das entidades, é importante ensinar o chatbot como recuperá-las em frases diferentes. A partir das frases nos exemplos de treinamento, o modelo a ser treinado deve ser exposto ao conteúdo de suas entidades, não necessariamente todos os valores possíveis, mas o suficiente para começar a generalizar e, então, extrair valores que não foram fornecidos durante o treinamento, mas são semelhantes. Para uma melhor precisão na extração de entidades, o chatbot também deve reconhecer as palavras antes e depois da entidade. Por isso, existe a necessidade de muitos exemplos, com diferentes frases, no seu treinamento.

3.4.2 Representação de diálogos

Os diálogos precisam ser representados por modelos abstratos para que possam ser generalizados. Devido à complexidade natural da linguagem humana, a modelagem dos diálogos exige um projeto cuidadoso dos dados de conversação, com a criação de padrões para que o chatbot possa ser treinado.

No Rasa, *stories* (histórias) e *rules* (regras) representam diálogos entre um usuário e o chatbot. As histórias são usadas no treino do modelo de ML para identificar padrões em conversas e, então, generalizar para outros caminhos que novas interações de diálogo possam tomar. Já as regras são usadas para determinar que certos trechos de diálogos devam sempre seguir o mesmo caminho, sem necessidade de uma tomada de decisão.

As *stories* são compostas por:

- *story*: nome de referência para melhor identificar a história;
- metadata: opcional que serve para informações como o autor, etc;
- lista de *steps* (passos): normalmente formam pares de *intents* e *actions* que compõem a história.

Um exemplo de *stories* pode ser visto na Figura 3.5. As entradas do usuário são representadas pelos nomes das intenções classificadas. Já as respostas do chatbot são representadas pelos nomes das ações que foram previstas.

```
38  stories:
39  - story: história com uma ação personalizada
40    steps:
41    - intent: feedback
42    - action: action_registrar_feedback
```

Fonte: Adaptado de Rasa Technologies [32].

Figura 3.5: *Stories com chamada a uma custom action.*

Nas *stories*, as *custom actions* devem ter seu nome começando com *action_*. Um exemplo de chamada a uma ação personalizada pode ser vista na Figura 3.5.

Ao escrever *stories*, não é preciso lidar com o conteúdo específico das mensagens que os usuários enviam. Em vez disso, aproveita-se a saída do fluxo da NLU, que permite usar a combinação de *intents* e *entities* para se referir a todas as mensagens possíveis de um mesmo significado. Por exemplo, a representação da mensagem do usuário "eu quero saber o saldo do meu cartão de crédito", na qual crédito é uma entidade, pode ser vista na Figura 3.6.

```
26  stories:
27  - story: história com entidade
28    steps:
29    - intent: saldo_conta
30      entities:
31      - account_type: credito
32    - action: action_credito_saldo_conta
```

Fonte: adaptado de Rasa Technologies [32].

Figura 3.6: *Mensagens do usuário com entities.*

A partir das histórias de treinamento, o chatbot Rasa aprende com a mensagem atual do usuário e os estados anteriores das conversas podem influenciar a previsão da próxima resposta. Assim, o desenvolvimento das histórias é todo baseado em uma sequência de *steps* (passos). Então, cada *step* da história pode conter:

- mensagem do usuário, representada por *intents* e suas *entities*;
- ação a ser executada pelo chatbot;
- declaração *OR*, que é uma forma de lidar com várias *intents* da mesma maneira, sem escrever uma *story* separada para cada *intent*;
- o *Slot* da memória que foi preenchido. Um *Slot* funciona como a memória do chatbot. Serve, por exemplo, para guardar o nome do cliente. Então, o nome poderá ser usado em futuras respostas ao próprio cliente. Os slots também podem ser configurados, se necessário, para que seus valores possam influenciar na tomada de decisão do chatbot, controlando, então, o fluxo de um diálogo com o usuário;

- Rasa *form* (formulário) para receber vários dados estruturados do usuário. O formulário é um tipo específico de ação personalizada que percorre um conjunto de slots necessários solicitando essas informações ao usuário até que todas elas sejam recebidas;
- *checkpoint* (ponto de conexão), que conecta uma história a outra história, formando uma história maior. Uma *stories* com um *checkpoint* será conectada a uma outra *stories* com o *checkpoint* de mesmo nome no momento em que o modelo é treinado.

Um pequeno trecho de uma história com o preenchimento de slots pode ser vista na Figura 3.7. O chatbot aprende, em seu treinamento, a decidir qual a próxima ação com base em uma combinação de *intents*, *actions*, *entities* e *slots* preenchidos.

```
stories:
- story: interactive_story_1
  steps:
  - intent: cumprimentar
  - action: utter_cumprimentar
  - intent: dizer_qual_produto
    entities:
    - produto: Pasta
  - slot_was_set:
    - precoR: R$ 0,25
  - slot_was_set:
    - preco: 0.25
  - action: action_consultar_produto
  - slot_was_set:
    - produto: Pasta com aba
  - slot_was_set:
    - corId: None
  - slot_was_set:
    - pid: 53
  - slot_was_set:
    - produto_encontrado: Pasta com aba
  - slot_was_set:
    - pqtd: 100
  - intent: saber_cores
  - action: action_saber_cores
```

Fonte: Dados de treinamento deste projeto.

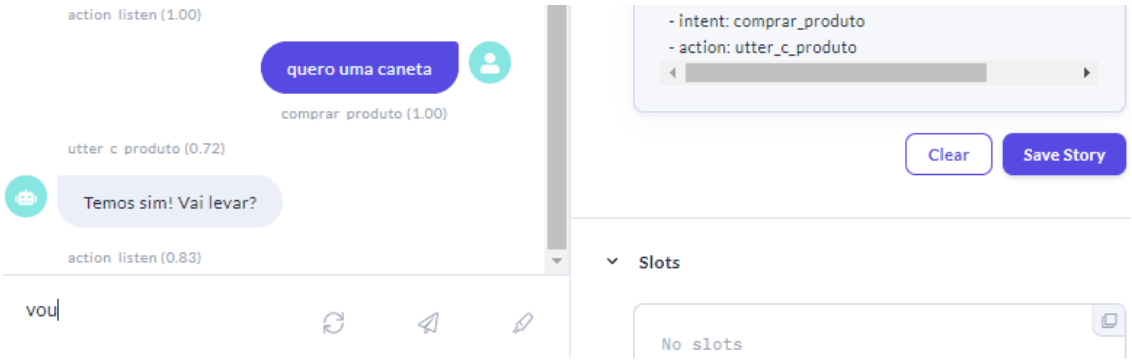
Figura 3.7: *Trecho de uma story com slots.*

3.4.3 Aprendizado interativo e Rasa X

O chatbot deve ser literalmente ensinado através de uma grande quantidade de dados de treinamento. A Rasa Technologies [32] recomenda a execução de treinamentos interativos para o desenvolvimento dos chatbots, principalmente em situações críticas envolvendo tomadas de decisões. Neste tipo de treinamento, no decorrer de um diálogo com o chatbot, todas as classificações e ações executadas pelo chatbot podem ser confirmadas ou corrigidas pelo interlocutor, garantindo que sejam gerados dados de treinamento corretos para o comportamento esperado do chatbot.

Para isso, o Rasa fornece duas alternativas para a realização do treinamento interativo do chatbot. Uma delas é o Rasa *Interactive Learning* (Aprendizado Interativo), feito através da interface de texto do terminal de comandos do sistema operacional e disponível executando-se o comando: *rasa interactive*. Neste aprendizado interativo, o Rasa exibe o grau de certeza da classificação da intenção, entidade ou da ação e o interlocutor sempre tem que confirmar a classificação atual ou informar a classificação correta, inclusive marcando entidades, antes do chatbot proceder. Também é possível acompanhar os slots sendo preenchidos, o histórico do diálogo e até mesmo desfazer o que já foi feito. Ao final são gerados *stories* bem completos, com o preenchimento correto de slots para o fluxo de diálogo desejado, além de exemplos de *intents* e *entities*. Todos estes dados de treinamento já são salvos no formato YAML e ficam prontos para serem usados na próxima execução do treinamento do Rasa NLU e Rasa Core, através do comando: *rasa train*.

Outra opção fornecida para executar um treinamento interativo do chatbot é através do Rasa X. Conforme a Rasa Technologies [32], o Rasa X é uma ferramenta que não é de código aberto, mas possui uma interface gráfica que facilita o acesso dos usuários de teste ao chatbot em desenvolvimento. Assim, esta ferramenta facilita o compartilhamento do chatbot em desenvolvimento com seus usuários de teste para, então, obter um *feedback* imediato, além de mais dados de treinamento.



Fonte: Elaborada pelo autor com Rasa Technologies [32].

Figura 3.8: *Exemplo de treinamento interativo do Rasa X.*

Com o Rasa X é mais fácil revisar e salvar os diálogos com o chatbot, além de verificar como estão os graus certeza sobre as intenções, entidades e ações a serem tomadas, bem como o preenchimento dos slots, como pode ser visto na Figura 3.8. Assim, com esta ferramenta é possível filtrar, sinalizar e corrigir conversas que não funcionaram bem e, então, melhorar continuamente o chatbot. Outra vantagem é que o Rasa X pode ser implementado tanto localmente, quanto usando servidores em nuvem, diversificando, assim, suas formas de disponibilização aos usuários de teste do chatbot.

CAPÍTULO 4

Revisão Bibliográfica

Partindo do princípio de que nenhuma pesquisa começa do zero, realiza-se neste capítulo uma revisão bibliográfica, objetivando um melhor posicionamento acerca do tema. Até o presente momento, muitos dos documentos sobre chatbots encontrados estão relacionados a pesquisas de avaliação ou aceitação do emprego de chatbots nos mais variados locais. Alguns destes trabalhos acadêmicos pareciam verdadeiros tutoriais sobre como fazer um chatbot. Então, limitando-se ao contexto de chatbots relacionados à área de vendas e ao comércio digital, alguns destes trabalhos são analisados a seguir.

Pugliesi, Clementino e Santos [31] publicaram um artigo sobre o desenvolvimento de um chatbot especificamente para troca e devolução de produtos em comércio eletrônico. Conceituaram inteligência artificial, processamento de linguagem natural e chatbots. Abordaram o contexto empreendedor de sua aplicação e o uso das ferramentas de gerenciamento de projetos Canvas e BPMN. Apresentaram o diagrama do empreendimento elaborado com o Canvas e o diagrama de casos de uso do chatbot elaborado com o BPMN. Foi usada a plataforma para chatbots DialogFlow do Google e, como interface para usuários, o React Native desenvolvido pelo Facebook. Então, apresentou-se duas telas do chatbot operando. Por fim, mostraram um trecho do código fonte para conexão com o DialogFlow e um trecho do código que envia mensagens para o React. Sua breve conclusão foi que, para se lançar uma versão oficial de sua aplicação, compensará pesquisar por alternativas ao DialogFlow, já que ele é gratuito apenas para testes.

Aarthi1 et al. [1] escreveu um artigo sobre a criação de um sistema comercial integrando um site de vendas de produtos com um chatbot, usando a linguagem Python como *front-end* e o banco de dados MS SQL como *back-end*. Apesar de realizar uma revisão bibliográfica sobre chatbots em geral, fundamentação teórica e propor o sistema, não mostrou qualquer especificidade da implementação que foi desenvolvida. Então, foram apresentados prints de telas do computador com o chatbot proposto executando localmente em uma simulação de uso. Sua breve conclusão foi que o chatbot pode ajudar a melhorar a comunicação entre o usuário e o site vendedor.

Yoda [39] elaborou sua dissertação de mestrado com uma pesquisa de campo exploratória de caráter qualitativo, analisando, assim, alguns chatbots sob o ponto de vista

do usuário. Nesta dissertação, foi feita uma revisão teórica voltada para a área de administração e marketing. Dentre os chatbots abordados, o chatbot da empresa ShopFácil, pertencente ao Bradesco, realizava a venda de produtos. Trata-se de um chatbot de vendas, acessível através do Facebook Messenger, bastante completo, permitindo a compra, processamento de pagamentos e acompanhamento do status do pedido. Ele permite ao cliente, inclusive, a escolha de ser atendido por algum dos diferentes personagens de chatbot com personalidades diferentes. A autora contextualizou a empresa e realizou uma linha do tempo com o histórico de desenvolvimento do chatbot. Também, foram abordados alguns conceitos de funcionalidades e tecnologias. Então, apresentou o diagrama do fluxo de assuntos do chatbot e em seguida elencou seus comentários acerca de sua experiência ao dialogar com o chatbot.

Asadi e Hemadi [6] apresentaram seu trabalho sobre o design de um chatbot para *e-commerce*. Nesta apresentação, são descritos os conceitos do recebimento de ordens de compras e do sistema de recomendações. Em seguida, são apresentados os diagramas dos fluxos estáticos do recebimento dos pedidos de compras e do sistema de recomendações. Também é feita uma descrição sobre a operacionalidade do chatbot.

Gadelha [19] realizou sua dissertação de mestrado sobre o uso de chatbots no atendimento dos clientes de uma empresa de revenda por catálogo. Apresentou uma fundamentação teórica sobre PLN e chatbots, incluindo um resumo acerca de alguns chatbots históricos. Também foi feita uma revisão bibliográfica de trabalhos relacionados a chatbots, embora nenhum destes seja especificamente sobre a aplicação do chatbot no contexto de vendas. Para o desenvolvimento do chatbot, o autor adotou o uso do então Rasa Stack, versão antiga do Rasa Open Source. Então, descreveu sua arquitetura e todos os componentes de ML disponíveis para seleção no Rasa à época. A pesquisa se concentrou bastante na análise de desempenho do ML, baseada nos testes automatizados do Rasa, sobre os diferentes *pipelines* formados por seus componentes. Depois de identificar as principais demandas, seu chatbot foi projetado para responder às requisições relacionadas à saída de mercadorias, novo revendor, avisos e boletos. Definiu-se, então, que acordos de pagamento ainda deveriam ser realizados por humanos. A implementação do chatbot foi testada localmente por alguns usuários selecionados e, em seguida, foi colocado em produção no Facebook Messenger. Então, sua utilização foi avaliada através de métricas disponibilizadas pelo Facebook Analytics. O texto concluiu destacando que o chatbot proporcionou uma melhora na taxa de retenção de atendimento.

Diante dos textos analisados, nota-se que apenas três deles abordaram o chatbot vendendo produtos. Um destes chatbots foi desenvolvido por uma grande empresa e, então, apenas avaliado pela autora. Assim, nenhum dos trabalhos revisados demonstrou como foi a implementação, de fato, do chatbot realizando vendas. Portanto, a realização de vendas pode representar um grande campo em potencial de pesquisas sobre chatbots.

CAPÍTULO 5

Metodologia

Para desenvolver o modelo do chatbot, isto é, sua representação abstrata a ser implementada, primeiro foi realizado um levantamento teórico sobre IA, ML, PLN, Grafos de Conhecimento e *frameworks open source* para criação de chatbots com IA. Em conjunto com uma revisão bibliográfica de artigos sobre o tema, estes estudos serviram para o embasamento técnico-teórico.

Em seguida, foi definido o setor comercial como a área de implementação deste projeto. Ao ser definida a área de aplicação comercial foi, então, estabelecido uma parceria com uma empresa de site de comércio eletrônico pela internet (*e-commerce*) para a qual o chatbot está sendo desenvolvido. Nesta empresa, foi realizado um levantamento dos requisitos que o chatbot deverá atender. A partir da análise dos requisitos foram, então, definidas junto a empresa todas as especificações que o chatbot deveria atender.

Com base nas especificações definidas com a empresa parceira supracitada foi, então, adotado o framework Rasa de código aberto para que possa atender a todos os requisitos técnicos, operacionais e de implementação definidos pela empresa.

Depois, foi preciso identificar, de acordo com o levantamento anterior, quais funcionalidades são viáveis de serem atendidas pelo chatbot. Isso levando em consideração sua complexidade e as funcionalidades que podem ser atendidas por um chatbot com o Framework Rasa.

Para desenvolver a base de conhecimentos do chatbot foi preciso definir todas as respostas possíveis que deverão ser implementadas. Depois, fornecer dados de conversação para a execução do *Machine Learning*, ou seja, o treinamento da sua rede neural profunda. Para isso, foi necessário usuários de teste para a obtenção de dados reais de conversas com o chatbot para a obtenção dos dados de treinamento.

Por fim, o modelo proposto do chatbot, desenvolvido para atender as necessidades e especificações da empresa e no qual são testadas as características de humanização implementáveis, foi avaliado por utilizadores e clientes em potencial, respondendo a um questionário, logo após contato e diálogo com o chatbot.

5.1 Desenvolvimento Orientado à Conversação

Para o treinamento da inteligência artificial do chatbot é preciso uma grande quantidade de exemplos de conversas reais de usuários. Mesmo assim, existe o problema de não ser possível antecipar tudo o que os usuários podem dizer durante o desenvolvimento do chatbot.

Para amenizar este problema a Rasa Technologies [32] propõe o Desenvolvimento Orientado à Conversação (do acrônimo em inglês CDD - *Conversation-Driven Development*). O CDD institui um processo de ouvir seus usuários e usar suas experiências para melhorar as respostas do chatbot. Assim, o Rasa CDD é uma abordagem centrada no usuário para construir chatbots com IA e, por isso, não é um processo linear, devido aos direcionamentos inesperados que os usuários possam dar à conversa. Portanto, é um processo colaborativo entre produto, design e desenvolvimento que revela o que os usuários estão pedindo. Com o tempo, este processo busca garantir que o chatbot esteja se adaptando ao que o usuário deseja, em vez de esperar que o usuário adapte seu comportamento.

O princípio por trás do CDD é que, em cada conversa, os usuários estão dizendo, em suas próprias palavras, exatamente o que desejam. Ao praticar o CDD em cada estágio do desenvolvimento do chatbot ele é adaptado para a linguagem e o comportamento do usuário real. Por isso, o CDD foi usado já no início da fase de implementação do chatbot na empresa.

Este CDD do Rasa pode ser realizado com a ferramenta Rasa X, que facilita as tarefas de coletar e salvar dados de treinamento do chatbot. Com esta ferramenta é possível visualizar e filtrar as conversas entre usuários e o chatbot e gerenciar modelos, para transformar essas conversas em dados de treinamento. O CDD do Rasa inclui as seguintes ações:

1. Compartilhar: entregar o protótipo aos usuários para testar o mais cedo possível;
2. Revisar: ler as conversas que as pessoas têm com o chatbot. Esta ação é útil em todas as fases de um projeto, do protótipo à produção;
3. Anotar: melhorar o modelo de NLU com base em mensagens de conversas reais. As mensagens registradas são usadas como dados de treinamento para o Rasa NLU. Já as sequências de passos (intenções e ações) são usadas como *stories* para que o Rasa Core possa ser treinado a tomar decisões;
4. Testar: usar conversas inteiras como testes de ponta a ponta do chatbot para identificar falhas ou pontos que podem ser melhorados;
5. Rastrear: identificar conversas bem-sucedidas. Então, usar esses dados para marcar e filtrar conversas para, assim, entender o que está funcionando e o que não está funcionando;

6. Corrigir: conversas mal sucedidas mostram os pontos que precisam de mais dados de treinamento ou se precisa corrigir o código, o fluxo do diálogo ou ainda, acertar a resposta fornecida pelo chatbot, além de permitir identificar a necessidade da criação de novas *intents* para o bot aprender a identificar, bem como suas devidas respostas esperadas. Já as conversas bem-sucedidas podem se tornar dados de treinamento ou de teste imediatamente. Reforçando, então, sua capacidade de acertos nas qualificações de intenções, extrações de entidades e tomadas de decisões.

5.2 Novo método de obtenção dos dados de treinamento

A ferramenta Rasa X, que seria utilizada para a realização do aprendizado interativo do chatbot e obtenção de dados de treinamento dos usuários, não funcionou adequadamente conforme era esperado. Ao serem executadas as *custom actions*, que demandavam mais de uma mensagem de resposta por parte do chatbot, o Rasa X exibia apenas uma das mensagens enviadas pelo chatbot, deixando o usuário sem saber todo o conteúdo da resposta completa do chatbot, impossibilitando, assim, a compreensão e correto prosseguimento dos diálogos fazendo o uso da ferramenta Rasa X. Soma-se a isto, o fato do Rasa X ainda não ter uma versão estável e que apresenta uma série de incompatibilidades entre as diversas APIs que utiliza, sendo muito difícil de se conseguir uma instalação do Rasa X que consiga ser executada. Provavelmente, esta possa ser uma forma da Rasa Technologies [32] dificultar o desenvolvimento de chatbots mais complexos sem ter que contratar a versão paga do Rasa, que oferece suporte técnico, dentre outras funcionalidades.

Como o Rasa X não funcionou adequadamente e na eminência da necessidade do desenvolvimento adequado deste projeto, foi necessário desenvolver um novo método de obtenção dos dados de treinamento. Assim, foram desenvolvidas algumas funções de extração dos dados de treinamento diretamente das conversas dos usuários com o chatbot em desenvolvimento. Dessa forma, foi possível a disponibilização do protótipo do chatbot em desenvolvimento para diversos usuários de teste, colaboradores voluntários, através do aplicativo de mensagens instantâneas Telegram e do *widget* de código aberto para páginas da internet Webchat. Estes dados de treinamento diversificados obtidos são fundamentais, não só para o treinamento do chatbot, como para o seu desenvolvimento como um todo.

CAPÍTULO 6

Levantamento de requisitos

6.1 Visão geral do produto

O projeto é sobre um chatbot de inteligência artificial eficiente que ajude a lidar com inúmeras dúvidas de clientes diariamente. O objetivo do chatbot foi atender os clientes de um *e-commerce*, tendo os clientes em potencial deste *e-commerce* como seu público-alvo.

A necessidade de se implementar o chatbot é para garantir um melhor atendimento aos clientes, mantendo-os satisfeitos e fidelizados, garantindo, mais vendas para a empresa, otimizando o processo de vendas e suporte ao cliente. Então, o chatbot teria como funcionalidades tratar bem os clientes como, por exemplo, dar boas-vindas aos novos clientes e atendê-los, prestando informações sobre os produtos vendidos, esclarecendo dúvidas, vendendo e sugerindo produtos. Também deverá ter funcionalidades de solucionar problemas dos clientes, encaminhando os problemas que não conseguir resolver diretamente à pessoa responsável.

6.1.1 Sobre a empresa parceira

A empresa parceira no desenvolvimento do chatbot é a Multitrem, uma empresa de comércio eletrônico que está passando por uma reestruturação e pretende inovar no atendimento dos seus clientes, usando a qualidade de atendimento como o seu principal diferencial neste competitivo mercado digital.

A Multitrem Comércio Digital Ltda (multitrem.com), teve início de suas atividades em junho de 2017, quando foi registrado o domínio do site. A empresa foi oficialmente registrada no início de 2018, tendo como foco se estabelecer localmente, em Goiânia-GO, como *e-commerce* para a revenda de produtos e também para ser utilizada como *marketplace* de outras empresas. O desafio de conquistar a confiança dos pequenos estabelecimentos comerciais foi pouco a pouco sendo vencido. Contudo, os pequenos comerciantes, principal foco do projeto, demonstraram não terem habilidades tecnológicas e de pessoal suficientes para lidarem com o comércio presencial e virtual simultaneamente.

Então, a Multitrem passou a buscar tecnologias que facilitassem o ingresso destas pequenas empresas no mercado virtual. Em 2020, com o início da pandemia e fechamento dos comércios locais, a empresa suspendeu suas atividades e aproveitou para se dedicar à evolução tecnológica. Atualmente, a Multitrem está passando por uma profunda reformulação e prepara seu retorno ao mercado digital em setembro de 2021.

6.1.2 Descrição dos usuários

Os usuários finais do chatbot são os clientes do *e-commerce*, compradores de produtos pela internet, dos mais variados perfis. Então, o chatbot deverá ter capacidade de interagir com diferentes perfis de clientes para atender às suas demandas.

Até então, estes clientes frequentemente tentam entrar em contato com a empresa para saber mais informações sobre os produtos ou sobre suas compras e não conseguem encontrar informações por conta própria na página do produto, nem procurar direito no site da empresa. Assim, o chatbot deverá preencher essa lacuna dos usuários no acesso às informações que precisam e, até então, não conseguem encontrar facilmente.

6.2 Premissas e restrições

Segue a lista das premissas que são adotadas durante a descrição dos requisitos:

- Premissa 1: parte-se do princípio que toda a implementação do projeto terá um baixo custo.
- Premissa 2: como não foi definido um orçamento prévio, a empresa arcará com todos os custos provenientes da implementação do projeto.
- Premissa 3: o chatbot se restringirá à língua portuguesa falada no Brasil.

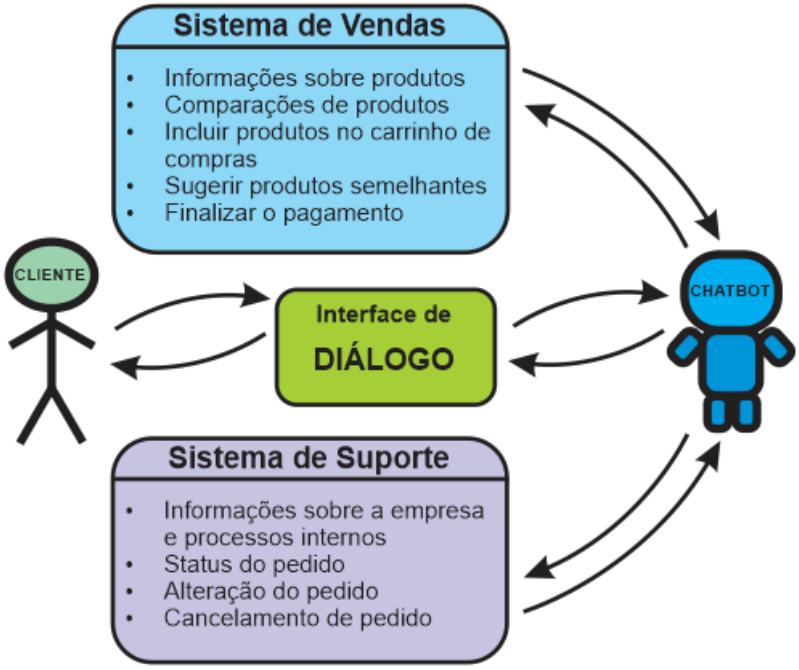
6.3 Requisitos funcionais

A partir da análise das informações coletadas e discutidas na empresa, foram descritos os requisitos funcionais do sistema de chatbot a ser implementado. Estes requisitos refletem as necessidades e expectativas, tanto da empresa, como de seus clientes. Para melhor clareza, as funcionalidades foram agrupadas e descritas a seguir.

- Prestar informações ao usuário sobre os produtos disponíveis no site: estoque, preço, detalhes, características e feedback de compradores;
- Incluir/retirar produtos do carrinho de compras, sugerir produtos e finalizar a compra.

- Prestar informações sobre a empresa e processos internos como prazo de envio, cancelamento de compras, andamento da resolução de problemas ou o status do pedido;
- Proceder alteração e cancelamento de pedido.

6.4 Diagrama de casos de uso



Fonte: Elaborada pelo autor.

Figura 6.1: *Funcionalidades providas pelo chatbot.*

Segundo Preece, Rogers e Sharp [30], um caso de uso é associado a um ator, e é o objetivo do ator ao utilizar o sistema que este caso de uso quer capturar. No diagrama de casos de uso da Figura 6.1, é possível ver o fluxo de informações entre o cliente, o chatbot e o sistema *back-end* da empresa. Este *back-end* trabalha em sincronia com o chatbot, fazendo a ligação entre os dados que vem do chatbot e o banco de dados da empresa, sempre aplicando as devidas regras de negócio, validações e garantias, criando, assim, uma camada de abstração para que o chatbot possa ser desenvolvido independentemente, com flexibilidade e escalabilidade.

O chatbot também não possui interface de usuário própria. Então, ele se comunica com o usuário a partir de outras aplicações como, por exemplo, o WhatsApp, Telegram ou um *widget* de conversação que pode ser adicionado ao site da empresa.

CAPÍTULO 7

Desenvolvimento do projeto

Como o desenvolvimento do chatbot foi proposto para atender a uma empresa, seu projeto ficou muito dependente da cadência de desenvolvimento dos próprios sistemas da empresa. Mesmo assim, com somente uma parte funcionando e apenas utilizando o sistema de vendas da empresa, o chatbot foi sendo desenvolvido visando o tema desta pesquisa aplicada sobre a humanização de chatbots. Dessa forma, o chatbot desenvolvido, com base nas possibilidades disponibilizadas pela empresa, passou pelas várias etapas a seguir.

7.1 Definição da plataforma

Com base nas especificações definidas com a empresa parceira anteriormente citada adotou-se o *framework* de código aberto Rasa, por ser mais versátil e flexível que outros disponíveis até então, para atender a todos os requisitos definidos pela empresa.

7.2 Especificações

Após análise dos requisitos da empresa parceira, foram definidas as seguintes especificações:

- suporte apenas à língua portuguesa falada no Brasil;
- deverá ter o nome de uma pessoa, definido pela empresa;
- personalidade de um vendedor bem dedicado;
- implementação do *framework* Rasa 2.0;
- aprimoramento do chatbot com o uso de dados de treinamento obtidos a partir de usuários reais;
- o chatbot deverá se conectar ao sistema interno da empresa, através de uma API padrão REST/JSON, para atender aos clientes;
- o chatbot terá respostas apenas para assuntos relacionados à empresa e a seus produtos.

7.3 Humanização do chatbot

Alan Turing, conhecido como o pai da computação, propôs o mais famoso teste sobre a humanização de uma máquina. No Teste de Turing, um interrogador humano teria que descobrir se as respostas escritas vêm de uma pessoa ou de um computador.

Para passar no Teste de Turing, segundo Norvig e Russell [28], a máquina precisaria ter as seguintes capacidades:

- processamento de linguagem natural para permitir se comunicar com sucesso no idioma da pessoa;
- representação de conhecimento para armazenar o que sabe ou aprende;
- raciocínio automatizado para usar os conhecimentos armazenados para responder as perguntas e até tirar novas conclusões;
- aprendizado de máquina para se adaptar a novas circunstâncias, detectar e extrapolar padrões.

Quando se trata da humanização de chatbots, muitas empresas acabam concluindo que o chatbot não precisa ser humanizado, mas sim ser eficiente e eficaz na execução das tarefas que lhe forem atribuídas. Por isso, comumente são encontrados atendimentos "mecanizados" nos chatbots já disponíveis no mercado.

Mas além de aumentar o poder de sua inteligência artificial, pouca coisa a mais pode ser feita para diminuir a impressão de se estar conversando com uma máquina. Segundo pesquisa realizada por Ciechanowski et al. [12], os participantes relataram menos efeitos estranhos e menos efeitos negativos em cooperação com um chatbot estático, de texto mais simples, do que com um chatbot de avatar animado mais complexo. Então, animações do personagem chatbot não foram aplicadas neste modelo desenvolvido.

Já Cross et al. [14] desenvolveu um estudo, bastante exploratório, projetado para ver como os chatbots podem parecer mais humanos. Seu intuito foi o de examinar como a presença de palavras em maiúsculas e erros de digitação se relacionam com percepções do usuário sobre a humanidade do chatbot. Porém, segundo o estudo, tiveram um efeito negativo sobre a percepção de humanidade do *bot* pelo entrevistado. Assim, a característica chatbot escrever corretamente, também, é boa para uma melhor aceitação do chatbot pelos clientes da empresa.

Portanto, o chatbot pode ter algumas características humanas como o nome próprio de uma pessoa, aparentar ter preferências por determinadas coisas, indicar outros produtos relacionados, sempre ser gentil nas respostas aos clientes, dentre outras. Mas a principal característica humana que o chatbot pode ter é a capacidade de entender a linguagem natural dos clientes e conseguir manter um diálogo fluido com eles. Também é muito importante a capacidade de tomar decisões, escolhendo a ação correta de acordo com o contexto da conversa.

7.3.1 Entendendo o cliente

O computador trabalha com operações lógicas e precisas. Entretanto, a linguagem humana é, geralmente, ambígua e sua semântica depende do contexto. Portanto, é difícil desenvolver uma aplicação que se aproxime de uma compreensão da linguagem no nível humano. Assim, apesar de não entender totalmente, a aplicação do PLN possibilita que os chatbot entendam, pelo menos, a intenção da linguagem natural e dêem respostas razoáveis.

Como o chatbot deve entender o que os clientes querem dizer em sua língua nativa, suas frases precisam ser analisadas para se extrair as informações relevantes. Porém, a linguagem natural possibilita todos os tipos de reações inesperadas do usuário. Portanto, é necessário um modelo de rede neural de PLN bem treinado. Para isso, se faz necessário uma ampla variedade de exemplos de conversas com clientes reais comprando os produtos da empresa. Então, durante o desenvolvimento é necessário compartilhar o chatbot com testadores reais, coletar as conversas que eles têm e, assim, gerar novos dados de treinamento.

7.3.2 Tratamento dedicado ao cliente

O tratamento dedicado ao cliente é um dos pontos chaves da humanização do chatbot, pois as respostas providas pelo chatbot podem tanto passar a impressão de se estar falando com uma máquina, devido a respostas sistemáticas iguais, como podem passar a impressão de se estar sendo bem tratado por uma pessoa real.

A personalidade do chatbot terá o perfil de um vendedor. Assim, o chatbot reagirá diante das interações dos usuários como se fosse uma pessoa que trabalhasse na loja, um funcionário dedicado, que tem o comportamento desenhado com base nos preceitos da identidade da empresa. Para isso, deve estabelecer um fluxo de diálogo, no qual suas "falas" estão de acordo com a estratégia do negócio.

Segundo o SEBRAE [35], o cumprimento é uma forma de demonstrar receptividade e comentários agradáveis valorizam a relação vendedor-cliente. Já as expressões cordiais como, por exemplo, "por favor, me informe..." costumam agregar valor à percepção do cliente. Então, o chatbot deve responder utilizando-se de construções linguísticas que transmitam empatia ao cliente. Nas informações prestadas ao cliente, a verdade é extremamente importante. Por fim, qualquer reclamação, sugestão ou elogio por parte do cliente deve ser registrada para acompanhamento por parte da direção da empresa.

Durante o atendimento ao cliente, o chatbot deve oferecer, sempre que solicitado, outras formas de atendimento que sejam realizadas por pessoas reais. Como, por exemplo, telefone e até presencial, para deixá-los sempre satisfeitos.

7.3.3 Áreas de conhecimento do chatbot

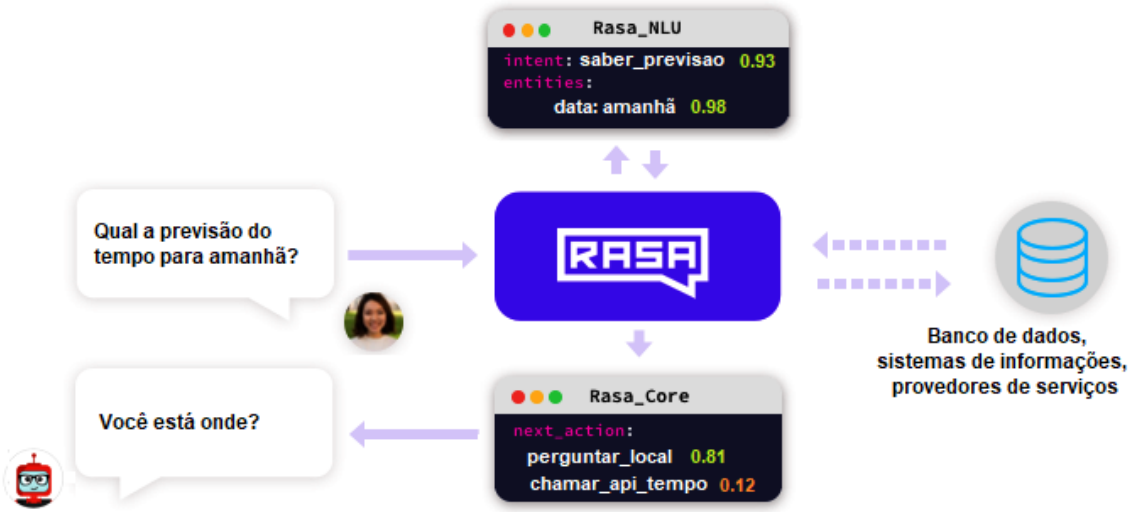
As áreas de conhecimento do chatbot podem ser divididas em duas. A primeira é a área especializada em vendas e informações sobre os produtos. Já a segunda é a área especializada em suporte ao cliente. As informações necessárias são fornecidas por requisições via API HTTP REST/JSON aos respectivos sistemas da empresa.

Para realizar uma venda, o chatbot poderá fornecer aos clientes diversas informações sobre os produtos, além de sua disponibilidade em estoque, visto que um mesmo produto possui diversas variações como, por exemplo, diferentes modelos, tamanhos e cores. Assim, estes conhecimentos permitirão a identificação precisa dos produtos que os clientes desejam comprar. Para isso, será necessário manter esta base de conhecimentos sobre os produtos sempre atualizada.

Já na área de suporte ao cliente, o conhecimento do chatbot se resume aos problemas comuns que podem ser enfrentados pelos clientes como, por exemplo, produto com defeito ou saber o prazo de envio. Na solução dos problemas deverão ser empregadas resoluções rápidas e simples para os problemas do cliente como, por exemplo, registrar a reclamação e já fornecer uma resposta como “Pronto! Nossa equipe entrará em contato para efetuar a troca. Algo mais?”, sempre usando expressões bastante cordiais como, por exemplo, “Por favor, me informe o problema do produto”.

7.4 Conexões com os sistemas da empresa

O chatbot Rasa roda em um servidor e pode executar requisições HTTP para outros sistemas usando APIs. A estrutura de fluxo de informações entre os sistemas da empresa, clientes e os componentes internos do Rasa pode ser vista na Figura 7.1.



Fonte: Adaptado de Rasa Technologies [32].

Figura 7.1: *Conexões do chatbot.*

A conexão com o sistema de informações dos produtos e vendas ocorre automaticamente durante o diálogo com um cliente. Para vender um produto ou prestar informações sobre produtos ou ainda fazer comparações, o chatbot se conecta ao módulo do sistema da empresa através de requisições de API HTTP padrão REST/JSON.

O chatbot consegue resolver os problemas dos clientes se conectando ao sistema responsável por informações de suporte técnico ao cliente via API HTTP REST/JSON. Todas as interações devem ficar registradas com um número de protocolo, para que seja possível para a empresa gerenciar as interações com os clientes e os clientes retornarem facilmente ao atendimento que fora prestado para maiores esclarecimentos.

7.5 Conexões com interfaces de usuários

Geralmente os frameworks de chatbots não fornecem uma interface gráfica de usuário (do acrônimo em inglês *Graphical User Interface* - GUI). Então, é necessário implementar uma interface de usuário ou optar pela integração com aplicativos de mensagens existentes.

Dentro do site de vendas da empresa, foi implementado um *widget* responsável pela interface entre o chatbot e o cliente. Os *widgets* são pequenos aplicativos, considerados como complementos de outros aplicativos. Assim, no site da empresa foi implementado o *widget* de código aberto Webchat, que pode ser incorporado à página da Web simplesmente adicionando-o ao código HTML da página.

Para se conectar a interface do aplicativo de mensagens instantâneas WhatsApp, o chatbot deve se conectar a uma conta do Twilio. Twilio é uma empresa líder mundial em serviços de telefonia e comunicação em nuvem, que provê serviços e APIs para serviços de mensagens, voz e vídeo. Com uma conta no Twilio ativa, então, já é possível utilizar a API do WhatsApp fornecida pelo Twilio por um período de testes. Depois, o pagamento deste serviço é cobrado por cada mensagem enviada.

Mas além do Twilio demorar vários dias para aprovar a conta da empresa, ainda é exigido um novo cadastro exclusivo do WhatsApp e bem mais rigoroso de ser aprovado para, enfim, permitir a utilização da WhatsApp Business API. Até o momento da conclusão deste trabalho escrito, a Multitrem ainda não havia conseguido a aprovação do seu cadastro para a utilização do WhatsApp. Porém, com o seu cadastro inicial já foi possível a realização de alguns testes de troca de mensagens pelo WhatsApp através de uma interface limitada e controlada chamada Twilio Sandbox.

O Telegram é um aplicativo de mensagens instantâneas gratuito e bastante popular, que está disponível para ser usado em vários dispositivos móveis como *smartphones* e *tablets*. Para o chatbot se conectar ao aplicativo Telegram, ele precisa ter um endereço na internet com o protocolo de segurança HTTPS. Depois é preciso se cadastrar no aplicativo

para obter as credenciais do aplicativo Telegram e, assim, se conectar a sua interface. Este cadastro é realizado de forma automatizada conversando com o *@BotFather*, o chatbot do Telegram encarregado de intermediar a configuração de todos os chatbots no aplicativo.

7.6 Obtenção de dados de treinamento de usuários

Segundo Preece, Rogers e Sharp [30], a melhor maneira de se assegurar que o desenvolvimento esteja levando as atividades dos usuários em conta é envolver usuários reais durante o desenvolvimento. Neste sentido, submeter o chatbot ainda em desenvolvimento para que usuários de teste possam experimentá-lo é um processo fundamental na obtenção de um chatbot mais complexo, pois pessoas diferentes perguntam coisas diferentes com as mais variadas construções textuais. Então, o chatbot precisa ser treinado também com tudo isto para, assim, conseguir reconhecer estas frases, além de conseguir generalizar para o reconhecimento de frases semelhantes.

7.6.1 Criação de um novo método de compartilhamento do protótipo e extração dos dados de treinamento reais

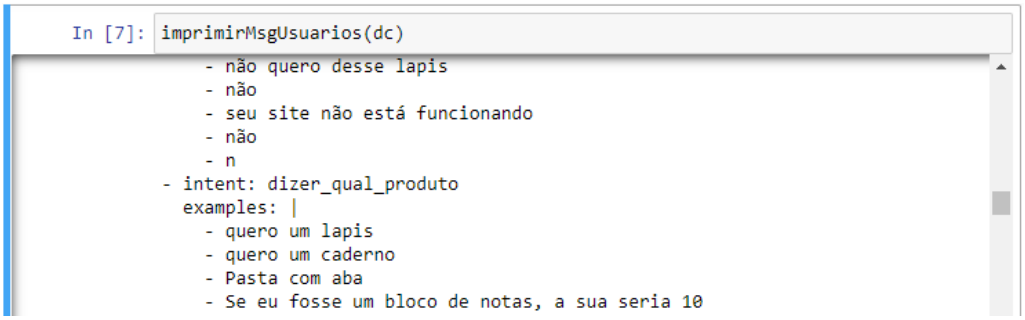
Foram desenvolvidas funções de extração dos dados de treinamento diretamente das conversas dos usuários de teste do chatbot, como pode ser visto na Figura 7.2.

(a) *Utilização do /log10.* (b) *Dados de todos os usuários.*

Fonte: Elaborada pelo autor.

Figura 7.2: *Obtenção de dados de treinamento.*

A Figura 7.2(a), mostra a função */log10* sendo aplicada no aplicativo Telegram. Esta função permite entender o que está acontecendo na hora da conversa. Já a Figura 7.2(b), mostra o resultado da função que registra todos os dados já no formato YAML.



Fonte: Elaborada pelo autor.

Figura 7.3: *Mensagens agrupadas por intenção.*

Na Figura 7.3, é possível ver a função de mostrar todas as mensagens agrupadas por intenções. Mas antes de se tornarem dados de treinamento, estes dados coletados ainda precisam passar por uma triagem manual, para verificar se estão classificados na intenção correta ou se existe a necessidade da criação de uma nova intenção. Neste caso, de se criar uma nova intenção, é necessário se criar algumas dezenas de variações de frases que normalmente seriam ditas com aquela intenção. Já para a criação de novas *stories*, é necessário revisar e corrigir estes passos das conversas, garantindo que o chatbot terá uma sequência correta de passos como exemplo.

Por fim, depois de todos estes dados de treinamento obtidos serem selecionados, retrabalhados ou replicados com reformulações, organizados, revisados, corrigidos e reunidos, é possível se executar o *Machine Learning*, que é realizado sobre todo este conjunto de dados para, então, se obter o modelo treinado da IA deste chatbot.

7.7 Implementação dos grafos conhecimento do chatbot

Normalmente as pessoas ficam mais satisfeitas ao comprar produtos em lojas em que recebem o máximo de apoio ao cliente, com um atendimento excelente. Neste sentido, uma das formas de se prestar apoio ao cliente é com o uso de um mecanismo de recomendação de produtos. Conforme Silveira e do Prado [36], sistemas de recomendação realizam uma seleção de informações e amenizam a sobrecarga de informações que os usuários normalmente lidariam ao buscar um produto. Assim, um grafo de conhecimento pode ser usado como mecanismo de recomendação. Embora os grafos de conhecimento normalmente sejam estruturas bastante complexas, o que se propõe neste projeto é a implementação de uma estrutura simples, mas que possa agregar qualidade ao chatbot como, por exemplo, sugestões de produtos baseadas no conhecimento do que outros clientes, geralmente, também compraram junto com o produto em questão. Outra importante forma de recomendação é pelos produtos mais bem avaliados. Por último, também, pode-se recomendar produtos em promoção.

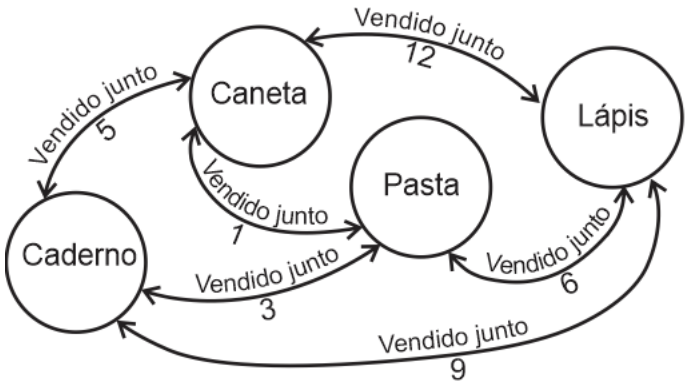
A implementação dos grafos de conhecimento deste projeto foi realizada através de estruturas de dados chamadas Dicionários, nativas da linguagem Python. O primeiro grafo de conhecimento é uma base de conhecimento para o chatbot ser capaz de oferecer sugestões de compras aos clientes, baseado na relação dos produtos que foram comprados juntos, por exemplo. Já os demais grafos são: um para o chatbot entender números escritos por extenso, inclusive escritos errados; outro para entender cores, também escritas erradas ou com variações como aumentativos, diminutivos e gêneros; e outro para o chatbot reconhecer alguns tipos de produtos para, por exemplo, o chatbot conseguir responder ao cliente que a loja não trabalha com aquele tipo de produto.

Na Figura 7.4(a), pode ser vista uma representação matricial do grafo de conhecimento implementado para o oferecimento de produtos. Já na Figura 7.4(b), pode ser observado como são feitas estas relações entre os produtos.

Número de vezes que os produtos foram vendidos juntos

	Caderno	Caneta	Pasta	Lápis
Lápis	9	12	6	
Pasta	3	1		6
Caneta	5		1	12
Caderno		5	3	9

(a) *Representação matricial.*



(b) *Relação de venda conjunta.*

Fonte: Elaborada pelo autor.

Figura 7.4: *(a) e (b) representam o grafo de recomendações de produtos.*

Já na Figura 7.5, podem ser vistas algumas triplas dos grafos de conhecimento implementados para o chatbot saber sobre os tipos de produtos, as quantidades por extenso e as cores não padronizadas. Os números e cores servem para o chatbot conseguir realizar uma consulta no sistema comercial da empresa utilizando os valores padronizados que estão cadastrados lá e, assim, obter sucesso na operação.

Entidade	Relação	Entidade
Cerveja	tipo de produto	Bebidas
Treis	número	3
meia duzia	número	6
azulao	cor	azul

Fonte: Elaborada pelo autor.

Figura 7.5: *Triplas do grafo de conhecimento.*

7.8 Realização de vendas

A principal função realizada pelo chatbot é consultar sobre produtos no sistema on-line da empresa. Para realizar esta tarefa, que é trivial para um humano, o chatbot precisou de uma complexa *Custom Action* escrita em código Python. Além de acessar o sistema da empresa, o chatbot precisa entender os detalhes do produto procurado, fazer uma triagem destes detalhes e saber qual requisição, no padrão REST, que cada detalhe pode ser consultado no sistema comercial da empresa. Por conseguinte, o chatbot ainda precisa saber retirar as informações solicitadas pelo usuário da resposta recebida em formato JSON da respectiva requisição padrão REST.

Nesta ação desenvolvida para consultar os produtos, o usuário pode informar, tanto todos os detalhes na mesma frase, quanto em frases separadas, com o chatbot solicitando pelos demais que forem necessários. Assim, o produto informado pelo usuário é consultado no grafo de conhecimento para saber o seu tipo e, então, o chatbot poder responder se vende aquele tipo de produto. Após consultar o sistema comercial com os dados no formato correto, o chatbot recebe, como resultado desta consulta, uma detalhada lista de produtos no formato JSON. O chatbot, então, consegue filtrar e mostrar apenas o que o usuário pediu. Quando existe mais de um produto que atende ao que o usuário pediu, o chatbot passa uma lista de opções, enumeradas por letras, para o usuário informar qual daqueles produtos seria o desejado. Neste caso, o usuário poderia escolher da forma mais conveniente que desejar, tanto informando a letra correspondente ao produto nesta lista, quanto informando algum detalhe do produto descrito na lista.

A qualquer momento o cliente pode solicitar mais detalhes, cores ou, ainda, qualquer pergunta desconexa com o produto. Caso o cliente informe uma quantidade por extenso, o chatbot consulta grafo de conhecimento, que é capaz de reconhecer algumas quantidades como, por exemplo, dúzias e resmas, e até mesmo alguns números escritos errados. Para o caso do cliente informar uma cor, o grafo do conhecimento também é capaz de reconhecer algumas cores escritas erradas, com aumentativos e diminutivos, gêneros e plurais, permitindo, então, que o chatbot consiga pesquisar no sistema da empresa por alguma cor padrão que normalmente é cadastrada no sistema, obtendo um maior índice de sucesso nas consultas ao sistema.

Ao adicionar um produto ao carrinho, o chatbot também consulta o grafo de conhecimento para saber qual produto também costuma ser vendido junto e, assim, oferecê-lo ao cliente, no caso do produto ainda estar disponível em estoque.

Já para realizar a retirada de produtos do carrinho, o chatbot usa uma função bastante semelhante à função de consultar produtos, mas, neste casso, o chatbot buscará pelos produtos que já estão dentro do carrinho de compras do cliente.

CAPÍTULO 8

Resultados

Os resultados obtidos, tanto em termos de conhecimentos adquiridos quanto em termos de produto desenvolvido neste projeto, são apresentados e discutidos a seguir.

8.1 Alguns números deste projeto

Até então, o modelo treinado deste chatbot já possui 266 blocos de história, 9.524 exemplos das 31 intenções distintas já classificadas e 6.363 exemplos de quatro entidades: nome, produto, quantidade e cor. Embora, alguns destes exemplos possam ser repetidos, estes números representam apenas os dados de treinamento. O projeto ainda conta com as 1.597 triplas que formam os grafos de conhecimento escritos na linguagem Python, sem contar as triplas do grafo de conhecimento de oferecimento de produtos, que é auto preenchido, além de 1.104 linhas de códigos das *custom actions* e de funções do grafo de conhecimento, também escritas em Python.

A maioria das intenções que precisam de respostas simples do chatbot foram agrupadas em dois subgrupos. Então, as perguntas frequentes resultaram em 34 subintenções faq (*Frequently Asked Questions*) e as frases de bate-papo cotidiano em 20 subintenções *chitchat*. Cada um destes subgrupos de intenções possuem um treinamento com um classificador extra para melhorar a resposta do chatbot.

8.2 Extração de entidades

Os extratores de entidades de terceiros, que já vêm treinados, não são bastante desenvolvidos para a nossa língua portuguesa e, por isso, não apresentaram resultados satisfatórios nos testes iniciais. Portanto, a extração de entidades foi toda desenvolvida a partir dos dados de treinamento. Com isso, foi possível um maior controle e, também, maior eficiência para a extração das entidades necessárias. Assim, este chatbot foi treinado para extrair as características dos produtos, números escritos errados, cores com variações na escrita e nomes de pessoas.

8.3 Problemas enfrentados

Alguns dos problemas de desenvolvimento enfrentados e contornados, durante o decorrer deste trabalho, são discutidos a seguir.

8.3.1 Problemas de generalizações

Um dos problemas enfrentados ao se fazer o próprio treinamento de extração de entidades é que o chatbot pode generalizar para coisas que não têm relação alguma com o que realmente era para ser extraído. Por exemplo, o chatbot pode generalizar que uma mensagem contendo uma única palavra iniciada em maiúscula como se a pessoa estivesse dizendo o seu nome. A solução para este problema é fornecer exemplos mais variados de frases com a pessoa dizendo o seu nome para o chatbot aprender como, geralmente, as pessoas informam o próprio nome.

A generalização dos exemplos de treinamento possibilita que o chatbot decida qual a melhor ação a ser executada baseado em todas as histórias exemplos que ele já aprendeu. Mas se não houver exemplos para uma determinada situação, pode ser tomada uma decisão de escolha de uma ação que não tem relação alguma com a situação. Ou pior ainda, uma ação que prejudique a conversa, como, por exemplo, finalizar uma compra ou cancelar uma compra, sendo que a intenção na verdade era outra. Então, se nos exemplos de treinamento tiver muitas finalizações de compras após o usuário responder "sim", quando ocorrer uma situação, ou seja, uma sequência de passos, que não havia nos exemplos de treinamento, na qual o chatbot identifique a mensagem do usuário com a intenção de afirmar, existe o risco do chatbot ter um alto grau de certeza de que a melhor ação a ser tomada seria finalizar a compra, fato que prejudicaria e muito a situação da conversa.

Para evitar este tipo de problema, é necessário fornecer uma maior variedade de situações para o chatbot durante o seu treinamento, para ele não ter que tomar uma decisão de uma situação sem saber o procedimento correto. Curiosamente, o fato do chatbot tomar uma decisão sem saber qual a correta se assemelha muito, por exemplo, a um funcionário humano tomando uma decisão errada, por nunca ter passado pela situação antes e nunca ter sido instruído do procedimento correto a ser feito. Ou seja, a falta de treinamento atrapalha tanto pessoas, quanto inteligências artificiais.

Percebeu-se que o Rasa, a partir da versão 2.4, implementou uma nova função para determinar o grau de certeza, tanto para classificação de intenções, quanto para a tomada de decisões, que veio a diminuir consideravelmente o grau de certeza para uma situação desconhecida. Provavelmente, a intenção desse grau de certeza menor seria para evitar que decisões erradas sejam tomadas. Mas, apesar de diminuir as chances de executar ações erradas, outra consequência notada foi o aumento da frequência das

respostas de falha do chatbot afirmando que não entendeu a mensagem do usuário. Isso pode ser cansativo para o usuário.

8.3.2 Problemas de interpretações e ambiguidades

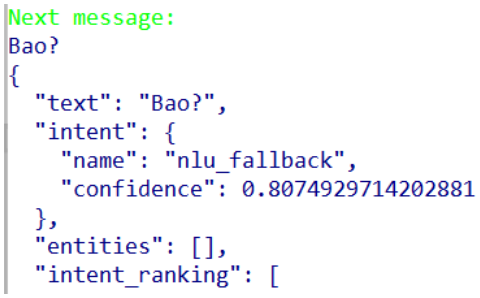
O problema da interpretação é comum com o uso das linguagens naturais, pois qualquer pessoa é passível de apresentar dificuldades ao interpretar mensagens, principalmente as mensagens mal elaboradas. Além de, muitas vezes, o próprio emissor da mensagem pode não conseguir expressar a mensagem que realmente queria transmitir. Então, para o caso dos chatbots os problemas se multiplicam, uma vez que se aliam o não entendimento da linguagem natural pelo computador com os problemas de expressão dos usuários, com as ambiguidades naturais da linguagem e ainda as sentenças semelhantes com significados bem diferentes como, por exemplo a negação de uma frase ou, ainda, uma pergunta retórica.

Uma forma de contorno para alguns destes problemas seria, além de tentar fornecer exemplos variados na execução do *Machine Learnig*, agrupar as sentenças de intenções semelhantes como se fossem uma única intenção mais geral, providenciando uma resposta mais ampla que satisfaça a todas as intenções semelhantes. É possível, ainda, fornecer exemplos de negação de uma frase. Mas, neste caso, seriam necessários exemplos de negação de todas as frases aprendidas, para, então, se obter um resultado com altas taxas de acerto. Mas o problema de muitas frases com as mesmas palavras e na mesma ordem é que pode desencadear em baixos graus de certezas na classificação das intenções, além de graus de certezas com valores muito próximos, fato que já geraria outro problema para o chatbot.

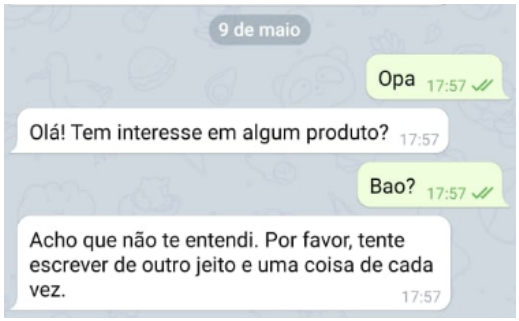
8.3.3 Problemas com grau de certeza baixo ou próximo

O problema do baixo grau de certeza, para algumas intenções, é crônico, principalmente para intenções que, em sua maioria, possuem exemplos com apenas uma palavra e, ainda, palavras curtas, como, por exemplo, as frases “Oi”, “Não” e “Bão”. O caso se agrava se, além de pequena, a palavra ainda não pertença aos dados de treinamento, como pode ser visto na Figura 8.1(a). Quando o Rasa não consegue um grau de certeza adequado, ele já classifica como uma falha e, então, retira este passo do contexto em execução, para não interferir nas tomadas de decisões futuras.

Já a Figura 8.1(b), mostra a resposta emitida pelo chatbot em razão do baixo grau de certeza. Então, os graus de certeza abaixo de limite estabelecido como aceitável já levam a uma resposta de falha no entendimento pelo chatbot. Assim como é possível se estabelecer um limite mínimo para o grau de certeza ser aceito, também é possível se estabelecer um intervalo limite entre estes valores para que possam ser aproveitados.



(a) *Baixo grau de certeza.*



(b) *Resposta ao baixo grau de certeza.*

Fonte: Elaborada pelo autor.

Figura 8.1: *(a) e (b) representam dois exemplos de falhas no reconhecimento da intenção.*

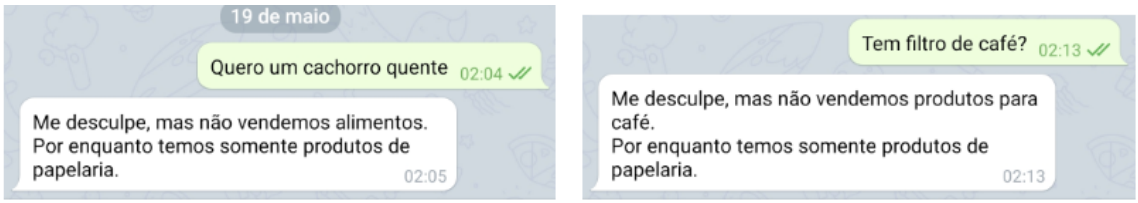
A partir da versão 2.4, o Rasa implementou uma nova fórmula para a determinação do grau de certeza, que tende a aumentar o valor dos graus mais altos e diminuir o valor dos graus mais baixos, levando, assim, a muitas intenções e decisões por ações, que antes eram aceitas, a não serem reconhecidas. Então, para possibilitar que situações de baixos graus de certeza, que mesmo com bastante dados de treinamento continuam com baixos graus de certeza, continuassem a ser aceitas, foi preciso se estabelecer empiricamente um novo valor de limite, diminuindo o limite anterior de 0.7 para apenas 0.28. Mesmo assim, dependo dos novos dados que são adicionados ao treinamento, pode ocorrer que algum exemplo novo venha a interferir no grau de certeza de uma intenção crítica, que já tinha o seu valor próximo deste limite mínimo, tornando-se não aceitável, levando à resposta de falha do chatbot.

Mensagens pequenas como, por exemplo, “oi” e “olá”, possuem pouca informação para o chatbot aprender, além de já estarem contidas em frases de outras intenções, necessitando, então, de um retrabalho na base da tentativa e erro, sobre os dados de treinamento. Este método da tentativa e erro é uma prática comum de ajuste na área da inteligência artificial, principalmente para se determinar os hiper parâmetros de treinamento. Assim, para o caso do chatbot Rasa, não adianta repetir milhares de vezes somente as palavras “oi” e “olá”, pois não apresenta melhora em seu grau de certeza. Então, este retrabalho consiste em procurar se existe algum exemplo de outra intenção que possa estar atrapalhando com o grau da intenção crítica em questão ou, então, tentar fornecer mais exemplos de frases maiores para que o *Machine Learning* obtenha mais informações contrastantes para a diferenciação da palavra “oi” contida na intenção de cumprimentar, com a palavra “oi” das outras intenções.

8.4 Grafo de conhecimento e a humanização

Até o presente momento, muitos chatbots são limitados a improvisar os menus de opções disponíveis, no site ou outro sistema, através dos aplicativos de mensagens instantâneas. Ao se propor um chatbot mais humanizado o objetivo é permitir um diálogo mais fluido e menos restritivo ao usuário. Portanto, como uma possibilidade de proporcionar mais qualidade de atendimento, este chatbot permite que o usuário possa dizer o que quiser, ao invés de se restringir em apenas clicar nos botões exibidos ou escolher dentre opções estáticas.

Ao realizar uma busca por um produto em um sistema computacional, normalmente, ele só consegue emitir uma resposta padrão como, por exemplo, “produto não encontrado”. Para o chatbot se parecer mais com um vendedor humano, o grafo de conhecimento permite, por exemplo, ao chatbot demonstrar que ele já conhece alguns tipos de produtos, através da capacidade de afirmar que a empresa não trabalha com aquele tipo de produto, como pode ser visto na Figura 8.2, ao invés de simplesmente emitir uma resposta computacional padrão à moda antiga, permitindo uma melhor condução do diálogo pelo chatbot.



Fonte: Elaborada pelo autor.

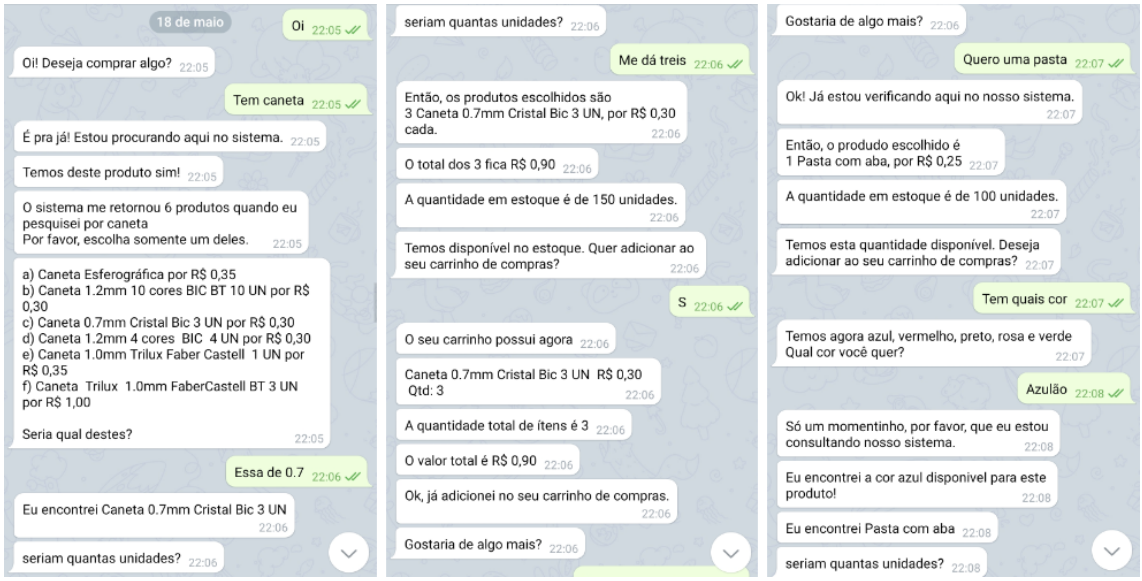
Figura 8.2: *Reconhecimento dos tipos de produtos.*

A nossa língua portuguesa possibilita inúmeras variações para uma mesma palavra. Além de plural, aumentativos, diminutivos e gêneros, durante os diálogos, as pessoas, também, costumam escrever errado e abreviar. Assim, este chat também consegue entender alguns números por extenso escritos errados e algumas cores com aumentativos, diminutivos e plurais, para conseguir consultar os valores corretos pela forma como estão cadastrados no sistema da empresa.

Até então, todos estes conhecimentos devem ser inseridos pelo desenvolvedor nos seus respectivos grafos de conhecimento e, assim, muitos ainda precisarão ser inseridos para aumentar a capacidade de reconhecimento deste chatbot. Mas a facilidade de aprender e utilizar o conhecimento aprendido é uma das principais características humanas. Assim, com o emprego do grafo de conhecimentos, este chatbot consegue literalmente aprender sozinho os produtos que possam ser de interesse dos clientes da empresa e, assim, oferecer estes produtos aos clientes durante o seu atendimento, permitindo um possível aumento nas vendas da empresa.

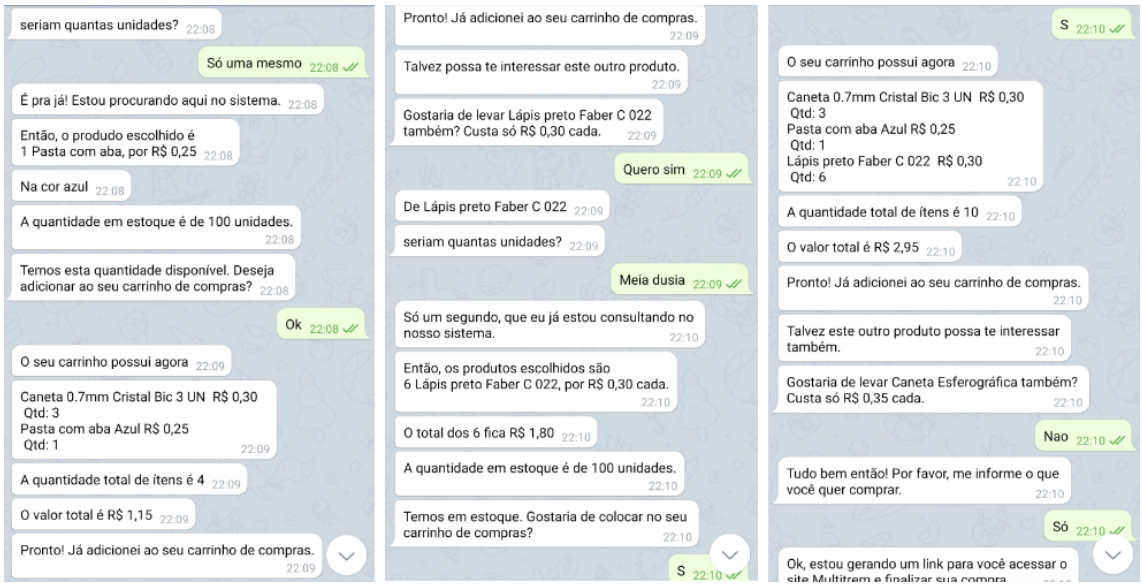
8.5 Exemplos da atuação do chatbot como vendedor

A seguir, apresenta-se um primeiro exemplo de diálogo completo. Neste exemplo, o chatbot deste projeto realiza uma venda com a utilização de algumas das funcionalidades providas pelo seu grafo de conhecimento.



Fonte: Elaborada pelo autor.

Figura 8.3: *Início do exemplo de uma venda pelo app Telegram.*



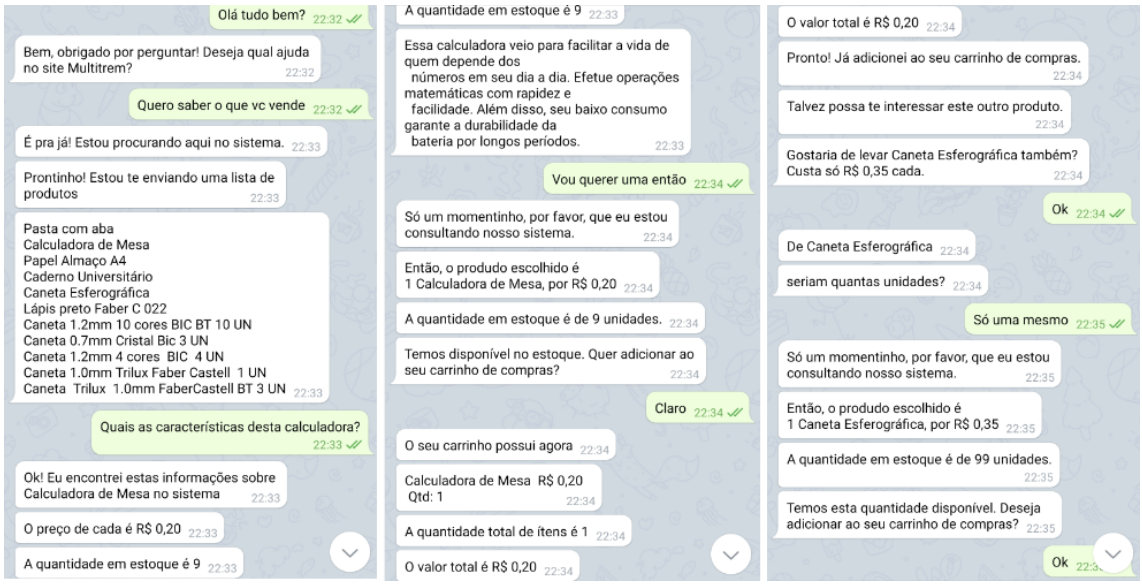
Fonte: Elaborada pelo autor.

Figura 8.4: *Final do exemplo de uma venda pelo app Telegram.*

Neste exemplo, mostrado nas Figuras 8.3 e 8.4, o grafo de conhecimento reconhece quantidades escritas erradas e uma cor no aumentativo, além de oferecer um

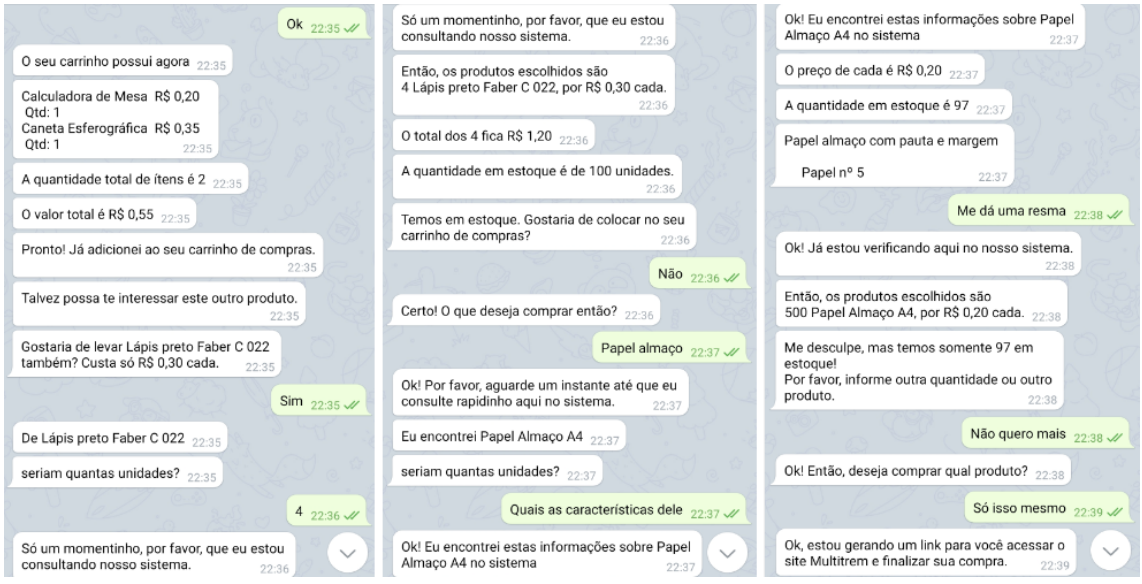
produto que aprendeu, anteriormente, que outros clientes, também, costumam comprar junto com o produto que está sendo comprado.

Já neste segundo exemplo de diálogo completo, apresentado a seguir, o chatbot deste projeto informa os produtos disponíveis e as características de alguns produtos.



Fonte: Elaborada pelo autor.

Figura 8.5: *Início do segundo exemplo de uma venda.*



Fonte: Elaborada pelo autor.

Figura 8.6: *Final do segundo exemplo de uma venda.*

Conforme este segundo exemplo, mostrado nas Figuras 8.5 e 8.6, o grafo de conhecimento reconhece quantidades como “uma resma” e, então, o chatbot consegue verificar no sistema da empresa se existe esta quantidade disponível em estoque.

8.6 Pesquisa avaliativa com usuários

Seguindo o design de interação [30], realizou-se uma pesquisa quantitativa para avaliar a percepção dos usuários durante o desenvolvimento do chatbot, comparando-o com outros chatbots que os participantes já tenham interagido ou opinando quanto à humanização do chatbot e também sobre a sua atuação na função de vendedor.

Esta pesquisa é baseada em um questionário disponibilizado através da plataforma Google Forms. Para Malhotra [24], o questionário ou o formulário (sua forma eletrônica) é um conjunto formal de perguntas cujo objetivo é obter informações, normalmente sem contato direto com os entrevistados. Neste sentido, o preenchimento de formulários pela internet é uma prática bastante desejável devido as recomendações de isolamento social vigentes em decorrência da pandemia Covid-19.

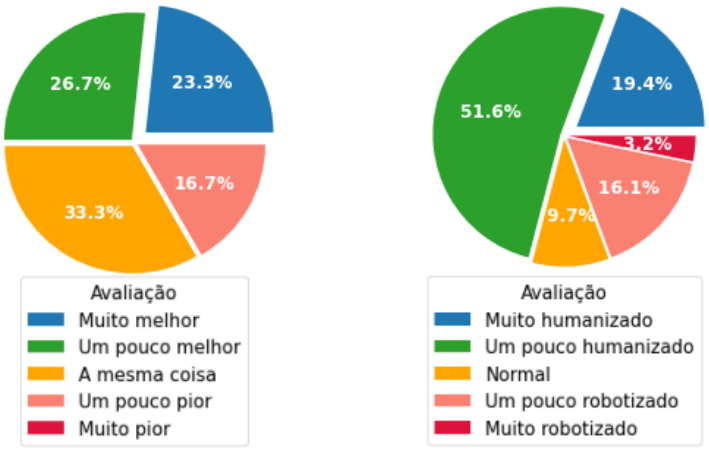
Conforme Preece, Rogers e Sharp [30], questionários bem projetados são eficientes para a obtenção de respostas à questões específicas. Assim, estas questões foram elaboradas para elucidar o quanto este chatbot pode ser considerado humanizado, na opinião dos usuários, no desempenho do seu personagem de vendedor. Além de quantificar o seu grau de aceitação para o desempenho desta função.

O questionário a ser respondido pelos usuários, após conversarem com o chatbot, teve as perguntas elaboradas conforme a escala de Likert. Segundo Mattar [26], na escala de Likert os respondentes são solicitados, não só a concordarem ou discordarem das afirmações, mas também a informarem qual o seu grau de concordância/discordância. Neste sentido, a amplitude de respostas permitida representa uma informação mais precisa sobre a opinião do respondente em relação a cada afirmação.

A pesquisa envolveu o envio de solicitações de participação para alguns membros da comunidade UFG e para alguns clientes em potencial da empresa que não haviam participado dos testes de desenvolvimento. Assim, antes de responder ao formulário, os participantes deveriam conversar com o chatbot, acessível através de links tanto para o seu Telegram, quanto para o Webchat do site. Não foi informado aos participantes o que poderiam falar com o chatbot e nem sobre o que ele vende, obrigando, assim, os participantes a conversarem e se informarem com o próprio chatbot e proporcionando uma melhor verificação de desempenho do chatbot ao conduzir um diálogo e realizar suas vendas.

Para Preece, Rogers e Sharp [30], quando se desenvolve um produto para o mercado aberto, a probabilidade de se conseguir um usuário para colaborar com a equipe de desenvolvimento é menor. Isso foi exatamente o que ocorreu neste trabalho, com poucos usuários de testes e poucas pessoas participando desta pesquisa avaliativa, que foi respondida por 31 voluntários que, em sua maioria, trocaram poucas palavras com o chatbot.

As duas primeiras perguntas foram apenas informativas e para controle, com "sim" ou "não", garantindo, assim, que os participantes entendiam o contexto para responder as demais perguntas. Todos os participantes afirmaram já saber o que é um chatbot e apenas um afirmou que nunca havia conversado com um chatbot e, por isso, não pôde responder à pergunta sobre o que achou deste chatbot em relação a outros que já havia conversado.

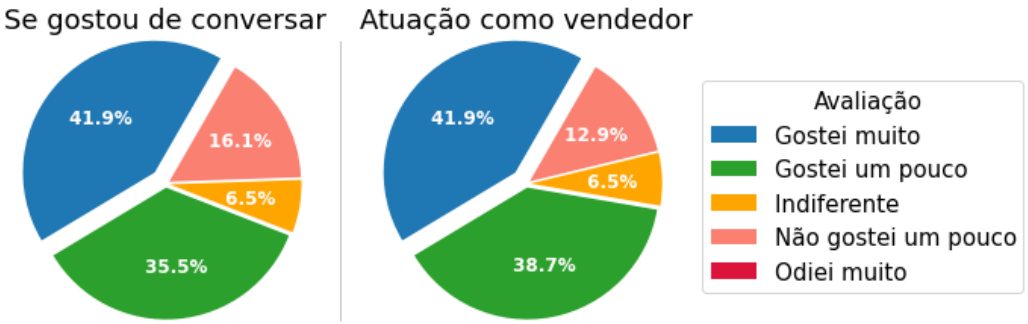


(a) *Comparação com outros.* (b) *Quanto a humanização.*

Fonte: Elaborada pelo autor.

Figura 8.7: *Resultados das duas primeiras questões da avaliação.*

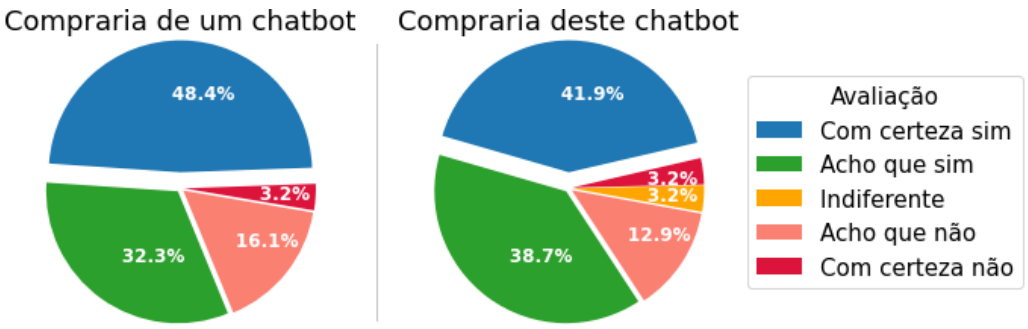
Na Figura 8.7, nota-se que a metade dos respondentes acharam este chatbot melhor do que outro(s) chatbot(s) que eles já haviam conversado e apenas 16,7% o acharam pior. Já a grande maioria achou este chatbot, pelo menos, um pouco humanizado.



Fonte: Elaborada pelo autor.

Figura 8.8: *Resultados sobre o quanto gostaram deste chatbot.*

Pela Figura 8.8, constata-se que a imensa maioria dos participantes gostou de conversar com este chatbot e, também, gostou da sua atuação como vendedor. Ressalta-se que, nenhum dos participantes respondeu com o grau mais baixo de insatisfação: "Odiei muito".



Fonte: Elaborada pelo autor.

Figura 8.9: *Resultados da aceitação de chatbot vendedor.*

Já na Figura 8.9, destaca-se que a maior parte dos participantes aceitaria ser atendido por um vendedor chatbot. Apenas um dos participantes foi totalmente contrário à ideia de ter um chatbot atuando como vendedor, afirmando ter a certeza de que não compraria deste vendedor chatbot ou de qualquer outro chatbot.

Pelo horário de envio do formulário foi possível a identificação da maioria dos diálogos que resultaram nas opiniões emitidas, com exceção de alguns casos de diálogos simultâneos. Como todos os formulários e seus respectivos diálogos foram conferidos, foi possível se determinar o porquê de algumas destas respostas na avaliação. Assim, constatou-se que a maioria dos diálogos se limitou entre quatro a oito mensagens enviadas ao chatbot. As opiniões ruins, geralmente, foram de pessoas que não conseguiram obter respostas satisfatórias do chatbot. Já as melhores avaliações foram feitas por pessoas que obtiveram respostas provindas dos grafos de conhecimento. Isto nos mostra que o grafo de conhecimento pode ser um diferencial de qualidade.

Muitas das avaliações ruins e diálogos curtíssimos, poderiam ser decorrentes do sistema comercial da empresa, que contava com poucos produtos cadastrados e poucas opções disponíveis. Assim, ao receber, logo no início do diálogo, respostas negativas, estas pessoas simplesmente já desistiam do teste. Aparentemente, as pessoas acostumadas a chatbots tradicionais, com opções estáticas, podem ter estranhado este chatbot, chegando a reclamar que ele não conseguiu responder todas as perguntas feitas sobre os produtos ou ainda, que ele deveria enviar uma lista com todos os produtos vendidos sempre que não encontrasse um produto. Convém ressaltar que, este chatbot foi projetado para uma quantidade indefinida de produtos e enviar uma lista muito grande de produtos também poderia desagradar. Já o treinamento de todas as perguntas possíveis, para todos os produtos possíveis, além de ser impossível poderia levar a um baixo grau de certeza ou a um grau de certeza bem próximo entre as perguntas.

Estas avaliações para ajudar no desenvolvimento permitiram, também, a constatação de várias falhas, como, por exemplo, o chatbot falhando ao tomar uma decisão logo após o usuário adicionar algo ao carrinho e negar a perguntas do tipo “Algo mais?”.

CAPÍTULO 9

Considerações Finais

Esta pesquisa aplicada sobre a humanização de chatbots com PLN, *Machine Learning* e grafos de conhecimento consistiu em um trabalho realizado em parceria com uma empresa privada. Esta empresa parceira, até então, ainda está desenvolvendo seus sistemas. Assim, muitas das funcionalidades esperadas pela empresa, para serem atendidas pelo chatbot, ainda não possuem sequer um sistema da empresa no qual o chatbot possa se comunicar. Logo, o chatbot foi desenvolvido apenas com os recursos de sistema da empresa já disponíveis. Mesmo assim, possibilitou-se o desenvolvimento de um chatbot com foco voltado principalmente na humanização e no uso de grafos de conhecimento, realizando a simulação da venda de alguns produtos de teste.

O chatbot apresentado neste trabalho ainda é um protótipo em desenvolvimento. Assim, ainda existem muitos padrões que ele ainda precisa aprender a reconhecer para, então, se adaptar aos mais variados usuários. Por fim, levando-se em conta as experiências obtidas até agora neste projeto e deduzindo-se com base nos resultados levantados ao final deste projeto, uma das conclusões que convém destacar é a de que um chatbot com o nível de desenvolvimento exigido para atender aos requisitos estabelecidos pela empresa em questão, levaria alguns anos para desenvolvimento e treinamento, contando com a participação de uma boa equipe para atingir uma capacidade razoável de discernimento e assertividade das respostas ao amplo conjunto domínio no qual suas funções de realização de atendimentos diversos, destinadas ao público em geral, estão inseridas.

A pesquisa avaliativa com os usuários mostrou que faltou ao chatbot a capacidade de apresentar melhor quais as opções que os usuários poderiam seguir, demonstrando que o chatbot possuía vícios que estavam adequados aos usuários de testes, mas se mostraram ineficientes aos demais. Após estas avaliações iniciais foi possível se considerar quais as melhores formas de um chatbot de vendas ser implementado. O chatbot poderia ser implementado para ser usado por aplicativos de mensagens instantâneas para receber pedidos rapidamente usando apenas a identificação do usuário no aplicativo de mensagens e o pagamento sendo realizado no ato da entrega ou via pix por exemplo. Já o seu uso dentro do site ficaria um pouco restrito ao esclarecimento de dúvidas, visto que acessando o site, muitos já prefeririam comprar direto sem ter que falar com o chatbot.

A empresa parceira espera usar o chatbot para aumentar a satisfação dos clientes, mas a adição do uso do chatbot pela empresa pode trazer, também, problemas, reclamações, irritações, insatisfações e frustrações dos clientes ocasionadas por falha do chatbot. Se já possível a ocorrência de problemas quando o atendimento é realizado pessoas reais, quando este atendimento é feito por um chatbot a ocorrência de problemas pode ser muito maior, visto que o chatbot não consegue entender totalmente a linguagem natural das pessoas. Portanto, fazer prevalecer as vantagens sobre as desvantagens do emprego de chatbots pela empresa será um dos desafios a serem enfrentados no futuro.

Já o grafo de conhecimento permite uma espécie de transição entre valores entendidos por humanos e os entendidos por sistemas computacionais, pois um sistema computacional exige valores normalizados ou padronizados. Já as pessoas possuem uma ampla variedade de formas não normalizadas ou regulares de informarem o que desejam. Assim, o grafo de conhecimento permite ao chatbot seguir um princípio fundamental do design de interação, que afirma que o produto deve ser adaptado ao usuário ao invés do usuário ter que se adaptar ao sistema. Então, será importante que as pesquisas futuras, sobre grafos de conhecimento, busquem por novas formas do grafo de conhecimento ser auto preenchido, ou seja, aprender sozinho como neste caso da recomendação de produtos, sem a necessidade destes conhecimentos serem inseridos pelo seu desenvolvedor.

Podem existir desafios para o mercado de chatbots no futuro, como, por exemplo, leis que regulamentam inteligências artificiais que tomam empregos de pessoas reais ou, ainda, uma grande empresa lançar, por exemplo, um vendedor virtual que converse com o cliente ouvindo sua voz, vendo e reconhecendo as imagens que o cliente mostre do produto desejado pela câmera e, também, responda claramente ao cliente por meio de voz. Por isso, as pesquisas na área de vendedores virtuais também poderiam evoluir com a ajuda de outros campos da inteligência artificial como, por exemplo, a visão computacional, além do reconhecimento e a sintetização de respostas audíveis.

Atualmente, os frameworks de chatbot, como todas as plataformas de chatbots, ainda são voltados para tarefas simples como perguntas frequentes ou a realização de alguma triagem inicial de atendimento. E o Rasa não é uma exceção. Assim, normalmente, os chatbots trabalham melhor em função de um conjunto domínio de atuação bem restrito e específico. À medida que se expande o seu conjunto domínio de atuação a sua complexidade também se expande. Só que ela se expande exponencialmente, pois existem infinitas possibilidades de perguntas possíveis quando o seu escopo de atuação é global. Então, geralmente, os chatbots são desenvolvidos com um pequeno escopo de atuação bem definido e as pessoas, normalmente, por não conhecerem quais são estas limitações, podem facilmente, e até frequentemente, obter a reposta de falha do entendimento por parte do chatbot. Portanto, a utilização das atuais plataformas de chatbots para a realização de tarefas como vendas em geral requer muita pesquisa e um extenso desenvolvimento de

funções para tentar suprir este déficit das plataformas de chatbot atuais.

Mas o desenvolvimento de novas funções, para que o chatbot possa tentar realizar tarefas um pouco mais complexas, não é uma solução ideal e muito menos trivial, visto que estas plataformas possuem e impõem uma série de limitações ao projeto. O Rasa, por exemplo, só suporta uma intenção por mensagem do usuário. Então, se a mensagem conter duas perguntas ou afirmações, por exemplo, ele já falha. Ele também suporta somente uma entidade de mesmo nome por mensagem. Assim, no caso da entidade produto, por exemplo, se o usuário pedir dois produtos diferentes na mesma frase, ele também falha. O Rasa também não consegue gerenciar extratores de entidades diferentes para uma mesma entidade. Por exemplo, poderia ser bastante útil combinar extratores diferentes para se aumentar a amplitude de reconhecimento.

Mas o fator mais limitante, para a execução de tarefas complexas, é não permitir que uma ação assuma o controle para solicitar e receber uma resposta do usuário e, assim, continuar de onde parou. Como, a cada mensagem do usuário, o Rasa pode decidir por executar quaisquer ações, não há garantias de que ele irá executar a ação que estava aguardando pela resposta. As decisões sobre as ações a serem executadas são baseadas no aprendizado sobre as sequências de passos constantes nas histórias dos exemplos de treinamento, a quantidade de combinações de passos possíveis aumenta exponencialmente com o número de intenções e ações. Além do mais, as entidades presentes nas intenções e, se os slots foram preenchidos ou não, também influenciam nas decisões e, por isso, são exponenciais combinatórios também.

Então, para grandes projetos, é impossível se fornecer exemplos de todas as situações que possam surgir. Como a falta de exemplos pode provocar decisões erradas ou, principalmente, levar a mensagens de falha do entendimento, por não se conseguir um grau de certeza aceitável de decisão por nenhuma ação, são, então, necessários muitos artifícios de desenvolvimento que somente são descobertos ou inventados através muita experiência prática por parte do desenvolvedor. Como, por exemplo, usar slots como *flag* para sinalizar que as próximas respostas podem ser para uma ação anterior que havia solicitado. Ou ainda, de dentro da ação errada que o Rasa encaminhou uma mensagem, já reencaminhar esta mensagem do usuário diretamente para a ação que a esperava.

Por fim, fazendo uma analogia sobre a utilização de sistemas limitados, o desenvolvimento de funcionalidades mais complexas do que as suportadas pelas plataformas de chatbots seria como ter que desenhar alguma figura em uma interface de texto usando-se apenas caracteres: além de ser bem notável a improvisação, a figura criada pode até se parecer com a pretendida, mas as possibilidades do desenho não sair como o pretendido são grandes. Então, considerando que a engenharia é uma arte, este trabalho foi uma arte de desenvolver soluções mais complexas em uma plataforma limitante em possibilidades e limitada de capacidades.

Referências Bibliográficas

[1] AARTHI1, N.; G.KEERTHANA2.; A.PAVITHRA3.; K.PAVITHRA4. **Chatbot for retail shop evaluation**. *International Journal of Computer Science and Mobile Computing*, 9:69–77, 2020.

[2] ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANÉ, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIÉGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y.; ZHENG, X. **TensorFlow: Large-scale machine learning on heterogeneous systems**, 2015. Software available from tensorflow.org.

[3] ADIWARDANA, D. **Towards a conversational agent that can chat about anything**. <https://ai.googleblog.com/2020/01/towards-conversational-agent-that-can.html>. Acesso: 02 mai. 2021, 2020.

[4] AINLEY, M.; FRYER, L. K.; THOMPSON, A.; GIBSON, A.; SHERLOCK, Z. **Stimulating and sustaining interest in a language course**. *ELSEVIER: Computers in Human Behavior*, 75:461–468, 2017.

[5] ALLEN, S.; GRAUPERA, V.; LUNDRIGAN, L. **Desenvolvimento Profissional Multiplataforma para Smartphone**. Alta Books Editora, Rua Viuva Claudio 291, Bairro Industrial do Jacare, CEP 20970-031 - Rio de Janeiro, 1a edition, 2012.

[6] ASADI, A. R.; HEMADI, R. **Design and implementation of a chatbot for e-commerce**. 02 2018.

[7] BELTRAMETTI, M.; COWLS, J.; CHATILA, R.; CHAZERAND, P.; DIGNUM, V.; LUETGE, C.; MADELIN, R.; PAGALLO, U.; ROSSI, F.; SCHAFER, B.; VALCKE, P.; VAYENA, E. **Ai4people: An ethical framework for a good ai society**. *Minds & Machines*, 28(4):689–707, 2018.

[8] BIRD, S.; KLEIN, E.; LOPER, E. **Natural language processing with python**. <http://www.nltk.org/book/>. Acesso: 13 set. 2020, 2019.

[9] BOCKLISCH, T.; FAULKNER, J.; PAWLOWSKI, N.; NICHOL, A. **Rasa: Open source language understanding and dialogue management**, 2017.

[10] BRASIL, P. **Python tutorial**. <https://wiki.python.org.br/PythonBrasil>. Acesso: 02 out. 2020, 2020.

[11] BUITINCK, L.; LOUPPE, G.; BLONDEL, M.; PEDREGOSA, F.; MUELLER, A.; GRISEL, O.; NICULAE, V.; PRETTENHOFER, P.; GRAMFORT, A.; GROBLER, J.; LAYTON, R.; VANDERPLAS, J.; JOLY, A.; HOLT, B.; VAROQUAUX, G. **API design for machine learning software: experiences from the scikit-learn project**. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, p. 108–122, 2013.

[12] CIECHANOWSKI, L.; PRZEGALINSKA, A.; MAGNUSKI, M.; GLOOR, P. **In the shades of the uncanny valley: An experimental study of human chatbot interaction**. *Future Generation Computer Systems*, 92:539–548, 2019.

[13] COPPIN, B. **Artificial Intelligence Illuminated**. Jones and Bartlett Publishers, 40 Tall Pine Drive, Sudbury, MA 01776, 1a edition, 2004.

[14] CROSS, A. C.; WESTERMAN, D.; LINDMARK, P. G. **I believe in a thing called bot: Perceptions of the humanness of chatbots**. *Communication Studies*, 70(3):295–312, 2019.

[15] EHRLINGER, L.; WOLFRAM, W. **Towards a definition of knowledge graphs**. <http://ceur-ws.org/Vol-1695/paper4.pdf>. Acesso: 11 set. 2020, 2016.

[16] FACELI, K.; LORENA, A. C.; GAMA, J. **Inteligência Artificial**. LTC, Travessa do Ouvidor, 11, CEP 20040-040, Rio de Janeiro, Brasil, 1a edition, 2011.

[17] FARRERAS, I. G.; FORD, W. R.; HILL, J. **Real conversations with artificial intelligence**. *ELSEVIER: Computers in Human Behavior*, 49:245–250, 2015.

[18] FISHER, J. **Mitigating social bias in knowledge graph embeddings**. <https://www.amazon.science/blog/mitigating-social-bias-in-knowledge-graph-embeddings>. Acesso: 25 nov. 2020, 2020.

[19] GADELHA, I. B. L. **O uso de chatbots no atendimento de clientes de revenda por catálogo**. Master's thesis, Universidade Federal do Pará, Programa Pós Graduação em Computação Aplicada, Tucuruí, 2019.

[20] GOOGLE. **Resultados da busca por ufg**. <https://www.google.com/search?q=ufg>. Acesso: 19 nov. 2020, 2016.

[21] GOOGLE. **Resultados da busca por ufg**. <https://www.google.com/search?q=teoria+dos+grafos>. Acesso: 20 nov. 2020, 2016.

[22] JSON.ORG. **Introducing json**. <http://json.org/json-en.html>. Acesso: 10 mai. 2021, 2021.

[23] KUROSE, J. F.; ROSS, K. W. **Computer Networking: A Top-Down Approach**. Addison-Wesley Publishing Company, USA, 8th edition, 2021.

[24] MALHOTRA, N. **Pesquisa de Marketing: Uma Orientação aplicada**. Bookman, 6a edition, 2012.

[25] MASSÉ, M. **REST API Design Rulebook**. O'Reilly Media Inc, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 1a edition, 2012.

[26] MATTAR, F. N. **Pesquisa de Marketing**. Elsevier Editora Ltda, 5a edition, 2012.

[27] MONTEIRO, S. D.; MOURA, M. A. **Knowledge graph and semantization in cyberspace**. *Knowledge Organization*, 41(6):429–439, 2014.

[28] NORVIG, P.; RUSSELL, S. **Inteligência Artificial**. Elsevier Editora Ltda., Rua Sete de Setembro, n 111 , 20050-006, Centro , Rio de Janeiro, Brasil, 3a edition, 2013.

[29] PAULHEIM, H. **Knowledge graph refinement: A survey of approaches and evaluation methods**. *Semantic web*, 8(3):489–508, 2017.

[30] PREECE, J.; ROGERS, Y.; SHARP, H. **Interaction Design: Beyond Human-Computer Interaction**. Wiley, Hoboken, NJ, 5 edition, 2019.

[31] PUGLIESI, J. B.; CLEMENTINO, J. S. Q.; DE OLIVEIRA SANTOS, J. P. **Mike: um chatbot para troca e devolução de produtos**. *Revista Eletrônica de Computação Aplicada*, 1(1):111–130, 2020.

[32] RASA. **Rasa open source documentation**. <https://rasa.com/docs/rasa/>. Acesso: 21 nov. 2020, 2020.

[33] ROBBINS, S. P. **Comportamento organizacional**. São Paulo: Pearson Prentice Hall, 2005.

[34] ROSEN, K. **Elementary Number Theory and Its Applications**. Addison-Wesley, 2011.

[35] SEBRAE. **Atendimento de qualidade**. <https://www.sebrae.com.br/sites/PortalSebrae/artigos/artigoshome/15-dicas-para-atender-bem>. Acesso: 26 nov. 2020, 2020.

[36] SILVEIRA, S. R.; DO PRADO, K. S. **Framework genérico de recomendação para lojas virtuais**. *RCT: Revista de Ciência e Tecnologia*, 2015.

[37] SINGHAL, A. **Introducing the knowledge graph**. <https://blog.google/products/search/introducing-knowledge-graph-things-not/>. Acesso: 02 mai. 2021, 2012.

[38] YAML. **The official yaml web site**. <https://yaml.org/>. Acesso: 12 out. 2020, 2020.

[39] YODA, F. S. **Atividades de chatbot no marketing de relacionamento em negócios digitais**. Master's thesis, Universidade de São Paulo, Departamento de Administração, Programa de Pós Graduação em Administração, São Paulo, 2019.

APÊNDICE A

Plano de Trabalho

Em anexo.

PLANO DE TRABALHO DO PROJETO FINAL DE CURSO 2

Título: Vendedor virtual

Dados	Discente
Matrícula/Nome	201602428 – Luiz Yokoyama Felix de Souza
Telefone	(62)98407-8114
E-mail	luizfelix@discente.ufg.br
Orientador(a):	Elisângela Silva Dias - Instituto de Informática - UFG
Curso:	Eng. de Computação
Certif. De Estudos	Não
Tipo de Projeto (Art. 13, Inciso V):	**pesquisa aplicada**

Resumo

O projeto consiste em uma pesquisa aplicada com o seguinte tema: **Humanizando chatbots com PLN, Machine Learning e Grafos de Conhecimento**. Provendo, então, continuidade ao trabalho desenvolvido na disciplina Projeto de Final de Curso 1 (PFC1). O emprego de chatbots (programa robô de conversação), apesar de estar em plena ascensão, ainda possui capacidades bastante limitadas, sendo normalmente relegado a triagens iniciais e respostas a dúvidas frequentes. Para o chatbot parecer mais humano e, com isso, melhor atender ao usuário, aplica-se machine learning (aprendizado de máquina) e o processamento de linguagem natural (PLN) para inferir a intenção da frase do usuário. Com a adição de um Grafo de Conhecimento (GC), um chatbot pode aprender a partir da própria interação com os usuários, sendo então, capaz de dar respostas mais elaboradas como, por exemplo, sugerir também coisas que outros usuários costumam buscar.

I. Objetivos.

Objetivo Geral:

O objetivo deste trabalho é analisar o modelo proposto de chatbot, com PLN, Machine Learning e grafos de conhecimento, aplicado na área de vendas.

Os objetivos específicos deste projeto são:

- Implementar o modelo do chatbot vendedor virtual projetado no PFC1.
- Reforçar o embasamento teórico e prático já levantado sobre o tema no PFC1;
- Expandir a base de dados conversacional do modelo supracitado;
- Avaliar o referido modelo.

II. Metodologia (atividades a serem desenvolvidas).

A metodologia utilizada neste projeto será:

- Pesquisa bibliográfica para reforço do embasamento teórico sobre o tema;
- Desenvolver o chatbot usando o framework adotado no projeto;
- No local de aplicação, implementar o chatbot projetado de acordo com a empresa;
- Disponibilizar esta implementação para usuários experimenta-la e avalia-la por meio de formulário;
- Avaliar as características e o desempenho das funcionalidades do modelo implementado.

III. Resultados Esperados

Os resultados esperados neste projeto são:

- Síntese do estágio de desenvolvimento dos conhecimentos, teóricos e empíricos, sobre o tema;
- Avaliação geral da implementação do modelo desenvolvido.

IV. Cronograma de Atividades

Na Tabela 1 é mostrado o Cronograma de Atividades.

Tabela 1 – Segundo semestre letivo de 2020.

Etapas do Projeto	**MAR 2021**	**ABR 2021**	**MAI 2021**	**JUN 2021**
1. Pesquisa bibliográfica e teórica	X	X		
2. Desenvolvimento/implementação	X	X		
3. Pesquisa avaliativa		X		
4. Análise dos resultados		X	X	
5. Elaboração da Monografia		X	X	
6. Apresentação do Projeto Final				X

Goiânia, 4 de março de 2021.

Luiz Yokoyama Felix de Souza

Assinatura do aluno
Matrícula: 201602428

Evangelo S. Dias

Assinatura do(a) Prof.(a) Orientador(a):