UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

LIA: um *chatbot* inteligente para o domínio de imóveis

Trabalho de Conclusão de Curso

Diogo de Lima Silva

São Cristóvão – Sergipe

2019

UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO

Diogo de Lima Silva

LIA: um *chatbot* inteligente para o domínio de imóveis

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Handrik Teixeira Macedo
Coorientador(a): Gilton José Ferreira

São Cristóvão – Sergipe

2019

Dedico este projeto a toda minha família, amigos e professores que me deram o apoio necessário para chegar até aqui.

Agradecimentos

Primeiramente a Deus que permitiu que tudo isso acontecesse ao longo de minha vida e não somente nestes anos como universitário, mas que em todos os momentos é o maior mestre que alguém pode conhecer. A esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

Aos meus pais Josefa de Lima e Agnaldo Lino, pelo amor, incentivo e apoio incondicional. A minha noiva Júlia Marília por todo o suporte e minha tia Marinalva por todo apoio durante todos esses anos de formação. A toda minha família que nunca desistiram de me apoiar nos momentos mais difíceis desta jornada.

Agradeço ao meu orientador e coorientador Hendrik Macedo e Gilton Ferreira, respectivamente, por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. A todos os professores e membros do grupo de pesquisa Ludii.co por participarem efetivamente deste projeto, em especial, aos discentes Tiago Conceição e Izaquel Alves; aos docentes Bruno Prado, Leonardo Matos e Kalil Bispo e ao parceiro de mercado Breno Peixoto. A todos que direta ou indiretamente fizeram parte da minha formação, meu muito obrigado.

Resumo

Atualmente o uso de agentes conversacionais inteligentes para auxiliar no desenvolvimento e execução de tarefas administrativas e estratégicas de empresas e organizações tem se tornado cada vez mais frequente. Atualmente existem dois tipos principais de *chatbot*: os que são baseados em regras e os que usam algoritmos e técnicas de inteligência artificial (IA) e aprendizagem de máquina. Esses agentes ou *chatbots* baseados em IA vão um nível além dos *chatbots* baseados em regras pré-definidas que já têm um fluxo de conversa pré-determinado e, com isso, a depender da resposta do usuário ele segue determinado caminho. Neste trabalho, inicialmente foi desenvolvido uma revisão sistemática acerca de quais ferramentas, métodos e técnicas existem na literatura sobre o tema. Posteriormente, também foi realizada uma pesquisa de mercado para determinar quais a principais *frameworks* existentes que utilizam estas ferramentas e técnicas no desenvolvimento de *chatbots* inteligentes. Assim, Logo após essa revisão sistemática e pesquisa de mercado as *frameworks* Rasa e *Botkit* foram selecionadas, pois juntas atenderam a todos os requisitos pré-estabelecidos. Por fim, neste projeto foi desenvolvido o agente inteligente LIA para o domínio de imóveis.

Palavras-chave: *chatbot*, agente inteligente, processamento de linguagem natural, aprendizagem de máquina, imóveis.

Abstract

Nowadays the use of intelligent conversational agents to assist in the development and execution of administrative and strategic tasks of companies and organizations has become more and more frequent. There are currently two main types of chatbots: rules-based chatbots and those that use artificial intelligence and machine learning algorithms and techniques. These AI-based agents or chatbots go one step further than predefined rule-based chatbots that already have a predetermined chat flow, and depending on the user's response, they follow a certain path. In this work, initially a systematic review was developed about which tools, methods and techniques exist in the literature on the subject. Subsequently, market research was also conducted to determine which major existing frameworks use these tools and techniques in the development of smart chatbots. Thus, Shortly after this systematic review and market research, the Rasa and Botkit frameworks were selected, as together they met all pre-established requirements. Finally, in this project the intelligent real estate agent LIA was developed.

Keywords: chatbot, natural language processing, smart agent, machine learning, properties.

Lista de ilustrações

Lista de Quadros

Lista de tabelas

Lista de códigos

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
abnTeX	ABsurdas Normas para TeX
AFM	Alphabet Frequency Matrix
API	Application Programming Interface
ARIMA	Auto-Regressive Integrated Moving Average
BRN	Bug Report Network
BTS	Bug Triage System
CAS	Context-Aware Systems
CCB	Change Control Board
CR	Change Request
CVS	Concurrent Version System
ES	Expert System
FLOSS	Free/Libre Open Source Software
GLR	Generalized Linear Regression
GQM	Goal Question Metric
HTML	HyperText Markup Language
IR	Information Retrieval
IRT	Recôncavo Institute of Technology
JDT	Jazz Duplicate Finder
LDA	Latent Dirichlet Allocation
LOC	Lines of Code
LSI	Latent Semantic Indexing
MS	Mapping Study
MSR	Mining Software Repositories

NLP	Natural Language Processing
PROMISE	Predictive Models in Software Engineering
RBES	Rule-Based Expert System
RHEL	RedHat Enterprise Linux
SaaS	Software as a Service
SCM	Software Configuration Management
SERPRO	Brazilian Federal Organization for Data Processing
SLR	Stepwise Linear Regression
SLR	Systematic Literature Review
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SVN	Subversion
TF-IDF	Term Frequency-Inverse Document Frequency
VSM	Vector Space Model
XP	Extreming Programming

Lista de símbolos

Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
$\in$	Pertence

Sumário

1 Introdução

Chatbots são sistemas de software que interagem com seus usuários por meio de uma conversação baseada em linguagem natural. Sua base de conhecimento consiste em uma coleção de regras, cujo acionamento depende dos padrões de texto reconhecidos na entrada do usuário. *Chatbots* ou agentes inteligentes baseados em aprendizado de máquina geram resultados mais práticos, pois fornece respostas com base no contexto da conversa e, com isso, tendem a ser mais fáceis de usar.

Agentes inteligentes representam uma mudança potencial na forma como as pessoas interagem com os dados e serviços online (BRANDTZAEG; FØLSTAD, 2017).

Van Baker, Pesquisador e vice-presidente do *Gartner's Application Innovation department* afirma que até 2020 devido a habilidade de *chatbots* de entender linguaguem natural e otimizar o tempo gasto em tarefas repetitivas, aproximadamente 50% das grandes empresas em média terão produtos em forma de *chatbots*.

Grandes empresas de tecnologia como *Google, Facebook* e *Microsoft* enxergam *chatbots* como a próxima tecnologia que se tornará mais popular. Em 2017, aproximadamente 30 mil *chatbots* foram lançados no *Facebook Messenger* (BRANDTZAEG; FØLSTAD, 2017).

Os *chatbots* são um meio para o engajamento direto do usuário por meio de mensagens de texto para fins de atendimento ao cliente ou *marketing*, evitando a necessidade de aplicativos ou páginas *Web* para propósitos especiais (XU et al., 2017).

O termo *chatbot* vem do inglês onde *chat* significa conversador e *bot* é uma abreviação para *robot* que significa robô. (SGANDERLA; FERRARI; GEYER, 2003) definem *chatbots* como sistemas computacionais que simulam o comportamento humano em conversas e que são capazes de analisar, interpretar e responder perguntas.

Logo, neste trabalho foi desenvolvido um *chatbot* chamado lia pra realizar conversas e tirar dúvidas de usuários sobre o domínio de imóveis, isto é, o lia manterá um diálogo flexível

e contextualizado sempre sobre a área de imóveis, quer seja para assuntos relacionados a compra e venda ou simplesmente só parar tirar e curiosidades de como fazer para adquirir um imóvel. Portanto, utilizou-se a *framework* Rasa integrada com o banco de dados *MongoDB* para confecção do Lia e, por fim, foi realizado testes de conversação para verificar o diálogo no domínio estabelecido.

1.1 Motivação

O mercado imobiliário é um ramo que movimenta grandes valores e possui uma enorme volubilidade. O fluxo de caixa de uma construtora pode variar bastante, de acordo com o ritmo das vendas ou a necessidade de investir em novos empreendimentos. Um dos grandes empecilhos para um resultado ainda melhor do setor está no fato de que um cliente muitas vezes precisa se desfazer do seu imóvel ou bens, para conseguir capital e então conseguir negociar e comprar um imóvel novo.

Assim, o objetivo desta solução está em facilitar as negociações de ambos os envolvidos e dirimir demais dúvidas tanto a construtora que está vendendo um imóvel novo quanto do cliente que está tentando vender seu imóvel atual para conseguir negociar um novo. A medida que a construtora pode assumir um papel de facilitadora da negociação e ajudar na efetivação da transação.

Portanto, a motivação para o desenvolvimento deste trabalho é a falta de aplicações de *chabots* inteligentes (*chatbots* que utilizam técnicas, métodos e algoritmos de inteligência artificial) que auxiliem corretamente as construturas a negociarem e vender seus imóveis de forma automatizada permitindo, assim, maior flexibilidade nas vendas e redução de custos com material humano.

1.2 Objetivos

Esta seção visa estabelecer os objetivos geral e específicos deste trabalho de pesquisa técnico-científico para que, assim, conduza-se todo o resto do trabalho pautados nos mesmos.

1.2.1 Objetivo Geral

Desenvolver o *chatbot* inteligente Lia para conversar com usuários sobre o ramo de imóveis sem perder o contexto da conversa.

1.2.2 Objetivos específicos

- Identificar arquiteturas, métodos e ferramentas para construção de *chatbots* baseados em IA;

- Identificar as *frameworks* existentes para construção de *chatbots* baseados em IA;
- Integrar o *chatbot* Lia com o MongoDB para persistência dos dados;
- Fazer testes de funcionalidade do *chatbot* Lia.

1.3 Método

Este trabalho se caracteriza como de mercado, bibliográco e prático devido ao levantamento dos estudos realizado na revisão sistemática do estado da arte, da análise de mercado e da aplicação prática dos mesmos. Os passos seguidos neste trabalho foram:

1. **Determinação do tema - problema**: nesta fase verificou-se a necessidade de exploração do contexto tanto mercadológico quanto bibliográfico para início das atividades e determinação da real necessidade de resolução do problema;
2. **Refletir sobre o uso de *framework* vs implementação própria**: nesta fase verificou-se que o uso de *frameworks* é a abordagem mais utilizada atualmente, pois diminui o tempo de desenvolvimento permitindo assim mais competitividade, reuso e maior foco na criação de diálogos e base de conhecimento dos *chatbots*.
3. **Levantamento bibliográfico**: nesta etapa foi realizada uma Revisão Sistemática da literatura seguindo o procedimento proposto por (KITCHENHAM; CHARTERS, 2007) para identificar e analisar trabalhos que proponham algum método, técnica, arquitetura ou algoritmo para auxiliar no desenvolvimento de *chatbots*. Assim, 2 estudos de suma importância foram encontrados como descritos no capítulo 3. No entanto, uma busca de mercado ainda se fez de suma importância para encontrar as ferramentas de desenvolvimento atuais como as *frameworks*, por exemplo;
4. **Pesquisa de mercado**: nesta etapa verificou-se a necessidade de exploração das *frameworks* utilizadas no mercado devido a falta de ferramentas para desenvolvimento prático na etapa de levantamento bibliográfico. Nesta etapa, também foi determinada a *framework* para desenvolvimento de *chatbots* mais adequada para este trabalho a partir de uma análise comparativa de características e restrições de projeto;
5. **Análise dos requisitos**: foi utilizado a técnica de estória de usuário para levantar os requisitos deste trabalho de maneira ágil e intuitiva. A partir disso, atribuiu-se pontos de acordo com a dificuldade das estórias;
6. **Desenvolvimento**: nesta etapa traz-se a arquitetura, detalhamento dos módulos, ferramentas e tecnologias utilizadas, implementação e testes do *chatbot* inteligente LIA;

7. **Considerações finais e trabalhos futuros**: nesta etapa apresenta-se os resultados obtidos do trabalho e possíveis próximos passos.

O presente trabalho foi conduzido para construção de um Mínimo Produto Viável (MVP) por meio do método Ludus. Este método foi criado com o objetivo de viabilizar a construção de um MVP de um produto utilizando-se como base o objeto de pesquisa tratado no Trabalho de Conclusão de Curso (TCC) do aluno. A justificativa para utilização desse método e desenvolvimento do MVP é explorar ao máximo o esforço despendido na realização da pesquisa e, com isso, impactar a sociedade, academia e pesquisador com um produto inovador, oriundo da pesquisa científica e rentável.

O método Ludus trata-se de uma abordagem que mescla gestão de projetos e engenharia de software construída para atender as especificidades da realidade acadêmica brasileira, mais especificamente o grupo de pesquisa Ludii.co[1], situado no Departamento de Computação da Universidade Federal de Sergipe. Os valores visados nó método são planejamento, objetividade e flexibilidade na condução. O método evolui em seu processo da seguinte forma:

1. Fase de definição - nesta etapa o aluno trata de definir o objetivo do MVP e como se dará sua associação com o TCC;
2. Fase de Planejamento - nesta etapa é realizado o trabalho mais pertinente a engenharia de software: a partir do objetivo criado o escopo do projeto é definido e quebrado em tarefas de desenvolvimento;
3. Fase de criação do *Backlog* - nesta etapa as tarefas recebem datas de entrega tendo em vista o prazo disponível para finalização do projeto;
4. Fase cíclica de gestão do projeto - nesta fase as tarefas vão sendo conduzidas semanalmente;

A fase é cíclica porque quando uma tarefa não é validada ela é alocada para uma outra semana após reavaliação. As tarefas possuem as seguintes situações:

- Tarefas da semana ou *sprint backlog*;
- Tarefas em andamento;
- Validação e teste;

[1] Laboratório para Universalização do Desenvolvimento, Inovação e Inteligência Computacional

1.4 Estrutura do Documento

Para facilitar a navegação e melhor entendimento, este documento está estruturado em capítulos que são:

- Capítulo 1 - Introdução: esta seção contextualiza acerca da problemática abordada, assim como os objetivos e o método utilizado;
- Capítulo 2- Fundamentação Teórica : esta seção define conceitos relevantes à assimilação do trabalho proposto;
- Capítulo 3- Trabalhos Relacionados: esta seção descreve o levantamento de informações obtidas por meio da revisão sistemática de literatura e revisão de mercado realizadas;
- Capítulo 4 - Desenvolvimento: esta seção apresenta o desenvolvimento do projeto proposto, descrevendo ferramentas, técnicas, métodos, algoritmos e abordagem utilizada;
- Capítulo 5 - Considerações Finais: está seção apresenta os objetivos alcançados, a relevância do trabalho proposto e a expectativa para os trabalhos futuros.

2

Fundamentação Teórica

Este capítulo descreve os principais conceitos e estudos que fundamentam este trabalho. Desta forma, definições de agentes conversacionais inteligentes ou *chatbots*, processamento de linguagem natural (PLN) e aprendizagem de máquina são apresentadas para embasar cientificamente este trabalho. Assim, este capítulo fica dividido da seguinte forma: seção 2.1 - conceitos de *chatbots*; seção 2.2 - processamento de linguagem natural e seção 2.3 - aprendizagem de máquina.

2.1 Conceitos de *Chatbot*

Um robô de conversação, também chamado de *chatbot*[1], é a tecnologia que visa interação de sistemas de software com humanos por meio da linguagem. *VanBaker*, pesquisador e vice-presidente do *Gartner's Application Innovation department*, afirma que até 2020, devido a habilidade de *chatbots* entender linguagem natural e otimizar o tempo gasto em tarefas repetitivas, aproximadamente 50% das grandes empresas em média terão produtos em forma de *chatbots*.

(JACOB, 2003; MARTINEZ, 2011) mostram que a interação conversacional humano computador ressurge e recupera popularidade com um novo conjunto de usuários aproximadamente a cada 10 anos. Nos anos 80, Linux e DOS empregaram linhas de comando para controlar os sistemas em quais programadores tiveram conversa com computadores por entrada de comandos, e depois de receber os comandos, computadores executaram exatamente as ações referidas nos comandos ao mostrar os resultados na tela para o usuários.

(SGANDERLA; FERRARI; GEYER, 2003) definem *chatbots* como sistemas computacionais que simulam o comportamento humano em conversas capazes de analisar, interpretar e responder perguntas.

[1] *Chabot* também pode ser conhecido como *smartbot, talkbot, chatterbot, bot*, agente interativo, interface de conversação e etc. Esses termos são constantemente usados como sinônimos.

(PAIKARI; HOEK, 2018) destaca que *bots*, abreviação para *robot* que signica robô, podem ser denidos como ferramentas de software capazes de automatizar tarefas repetitivas como um *web crawling*[2]. Nesse sentido, *chatbots* são um tipo de *bot* que interagem com usuários por meio de diálogos de texto ou voz.

Alguns *chatbots* podem, por exemplo, monitorar o que os usuários digitam em canais como *Facebook* ou *Telegram* e acionar determinadas ações quando certos padrões estão presentes (PAIKARI; HOEK, 2018).

Ainda segundo (PAIKARI; HOEK, 2018), eles podem se engajar em conversas mais signicativas e contextuais com os usuários utilizando processamento de linguagem natural (PLN) e Inteligência Articial (IA).

A assistente virtual da *Apple*, *Siri* [3], e a assistente da *Microsoft*, *Cortana* [4], podem se engajar em conversas e desempenhar funções como responder sobre condições climáticas ou ativando o alarme do celular quando solicitado, por exemplo. Contudo, esses assistentes virtuais conversam e entendem assuntos gerais, isto é, contextos que pertencem a vários domínios enquanto que *chatbots* tendem a ser mais específicos abordando áreas específicas como, por exemplo, o domínio de imóveis.

2.1.1 Tipos de *chatbot*

Chabots geralmente são classicados em dois tipos:

1. Baseados em regras: *chatbot* que funciona com um conjunto de regras predefinidas. Esse tipo de *chatbot* é limitado, já que seu comportamento é todo programado antecipadamente (KAR R.; HALDAR, 2016).

2. Baseados em IA: *chatbot* que dá a impressão de ser mais inteligente por usar processamento de linguagem natural, não somente casamento de padrões e regras predefinidas, e ficar mais inteligente a medida que interage e mantém informações sobre os estados de conversação (KAR R.; HALDAR, 2016).

Além do processamento de linguagem natural, *chatbots* baseado em IA podem utilizar algoritmos de reconhecimento de voz e aprendizado de máquina.

2.1.2 Canais de interação

Canais de interação são aplicações que executam em dispositivos *desktop* ou *mobile* e fornecem um meio de comunicação aos usuários com o *chatbot*.

[2] Um rastreador da rede, em inglês web crawler, é um programa de computador que navega pela rede mundial de uma forma metódica e automatizada. Disponível em: <https://pt.wikipedia.org/wiki/Rastreador$_web$ >

[3] Disponível em: <https://www.apple.com/br/siri/>

[4] Disponível em: <https://support.microsoft.com/pt-br/help/17214/windows-10-what-is>

(BRANDTZAEG; FØLSTAD, 2017) mostram que desde 2016 muitos serviços de conversação incluindo serviços como o *Facebook Messenger*, *Slack*, *Telegram*, *Skype*, *LINE* e *WeChat* lançaram *Chatbots*. Fornecendo, assim, *APIs* aos desenvolvedores para que eles pudessem construir novos tipos de aplicativos para interação e serviços de informação.

É importante destacar que segundo (BRANDTZAEG; FØLSTAD, 2017) *chatbots* também são utilizados em páginas *web* e aplicativos *mobile* como um meio comum de interação com dados e serviços.

De acordo com (KAR R.; HALDAR, 2016) em algumas abordagens de implementação os canais de interação possuem uma interface intermediária, também chamado de conector, que pode utilizar *Webhooks*[5] para realizar a comunicação.

Em suma, os principais canais de comunicação para *chabots* são:

- Serviços mensageiros
- Páginas e sistemas *web*
- Aplicativos *mobile*

2.1.3 *Engine Chatbot*

A *engine* é responsável por transformar linguagem natural em uma ação entendível por máquinas e segundo (KAR R.; HALDAR, 2016) é o componente mais importante de um *chatbot*.

De acordo com (KAR R.; HALDAR, 2016) as *engines* de *chatbots* geralmente são desenvolvidas utilizando-se vários modelos de PLN e aprendizado de máquina para prover níveis aceitáveis de precisão.

Com o objetivo de facilitar o trabalho de desenvolvedores de *chatbots* algumas empresas oferecem essa *engine* como um serviço[6]. Exemplos desse tipo de serviço são: *Wit.ai*[7] e *IBM Watson Assistant*[8].

Os principais conceitos em uma *engine* de *chatbot* são: entidades, intenções, contexto e diálogo.

- **Entidades**: as entidades são informações específicas de um domínio que são extraídas de uma expressão na qual mapeiam as frases de linguagem natural para as suas frases canônicas com o objetivo de entender a intenção (KAR R.; HALDAR, 2016). Além disso, ajudam a identicar os parâmetros necessários para tomar ações específicas (KAR

[5] *Webhooks* são *callbacks HTTP* que são denidos pelo utilizador do serviço.
[6] Esse tipo de abordagem é conhecida como *Software-as-a Service(SaaS)*, em português, Software como um serviço ou *AI-as-a-service*, em português, IA como um serviço.
[7] Disponível em : https://wit.ai
[8] Disponível em : https://www.ibm.com/cloud/watson-assistant/

R.; HALDAR, 2016). O *email* de um usuário, por exemplo, seria uma entidade que o *chatbot* poderia utilizar para realizar a ação de enviar documentos ou noticações.

- **Intenções:** as intenções são cruciais em uma aplicação de *chatbot.* As intenções representam o que os usuários estão buscando realizar ou saber dada uma mensagem (KAR R.; HALDAR, 2016).

- **Contexto:** determinar o contexto de uma expressão criada pelo usuário é uma funcionalidade considerada importante em *chatbots* modernos. O contexto pode ser usado para lidar com situações onde a entrada do usuário seja muito vaga ou possui múltiplos significados baseado no histórico de conversação. Contextos representam a habilidade dos agentes em manter o estado da conversa para utilizar como forma de identicar a intenção do usuário (KAR R.; HALDAR, 2016).

- **Diálogo**: o diálogo utiliza as intenções, as entidades e o contexto da aplicação para retornar uma resposta baseada na entrada do usuário.

2.2 Processamento de Linguagem Natural

O processamento da linguagem natural (PLN) trata computacionalmente os diversos aspectos da comunicação humana, como sons, palavras, sentenças e discursos, considerando formatos e referências, estruturas e significados, contextos e usos. Em sentido bem amplo, podemos dizer que o PLN visa fazer o computador se comunicar em linguagem humana, nem sempre necessariamente em todos os níveis de entendimento e/ou geração de sons, palavras, sentenças e discursos. Estes níveis são:

- fonético e fonológico: do relacionamento das palavras com os sons que elas produzem;

- morfológico: da construção das palavras a partir de unidades de significado primitivas e de como classificá-las em categorias morfológicas;

- sintático: do relacionamento das palavras entre si, cada uma assumindo seu papel estrutural nas frases, e de como as frases podem ser partes de outras, constituindo sentenças;

- semântico: do relacionamento das palavras com seus significados e de como elas são combinadas para formar os significados das sentenças;

- pragmático: do uso de frases e sentenças em diferentes contextos afetando o significado.

A representação do significado de uma sentença, independente de contexto, é obtida através de sua forma lógica (ALLEN, 1995; FRANCONI, 2001). A forma lógica codifica os possíveis sentidos de cada palavra e identifica os relacionamentos semânticos entre palavras e

frases. Uma vez que os relacionamentos semânticos são determinados, alguns sentidos para as palavras tornam-se inviáveis e, assim, podem ser desconsiderados.

A estrutura sintática de uma sentença é obtida através do processamento morfo sintático, sendo que a representação dessa estrutura é regida por leis gramaticais definidas em uma gramática. Outras informações necessárias a esta etapa são como as categorias morfológicas das palavras encontram-se em um léxico.

O mapeamento da estrutura sintática da sentença em sua forma lógica é realizado pelo processamento semântico e, nele, o léxico também exerce papel fundamental com informações sobre o significado dos itens lexicais. desta forma, vimos que a gramática e o léxico são recursos indispensáveis para a transformação da sentença em sua forma lógica. Vamos examiná-los um pouco mais nas subseções seguintes.

2.2.1 Gramática

Uma gramática é constituída por um conjunto de regras de boa formação das palavras e das sentenças de uma língua. Essas regras permitem dupla função para as gramáticas (BOUILLON, 1998) : a função normativa, que define regras de combinação das palavras gerando sentenças corretas; e a função representativa, que associa a uma ou mais frases suas representações sintáticas.

Uma boa gramática deve ser segundo (ALLEN, 1995) (a) suficientemente genérica, aceitando o maior número possível de sentenças válidas; (b) seletiva, reconhecendo os casos identificados como problemáticos e (c) inteligível, favorecendo o entendimento de suas regras, principalmente, pela simplicidade das mesmas.

Uma gramática é dita gerativa quando consegue traduzir os fatos linguísticos (inclusive os aspectos criativos) da linguagem por meio de regras e processos explícitos, precisos e de aplicação automática, obedecendo a condições específicas (LOBATO, 1986).

Diversos formalismos de representação computacional podem ser usados para representar uma gramática (NUNES et al., 1999). Um destes formalismos é o da gramática de constituintes imediatos *(phrase-structure grammar – PSG)*, que é definida como uma quádrupla <T,N,P,S>, onde: T representa o conjunto das palavras da língua, N representa o conjunto das categorias funcionais e das categorias lexicais, P representa o conjunto de regras de produção, e S representa o símbolo inicial pertencente a N.

Não há um formalismo eleito como o melhor. Os modelos que se situam entre as gramáticas livres de contexto e aquelas sensíveis ao contexto têm sido propostos pelos pesquisadores como os mais indicados (VIEIRA; LIMA, 2001). De qualquer forma, quanto ao PLN, é indispensável o uso de critérios formais para a construção das regras gramaticais. Esses vão se aliar a outro recurso do PLN que é o léxico.

2.2.2 Léxico

De forma genérica o termo "léxico" significa uma relação de palavras com suas categorias gramaticais e seus significados. Em relação a uma determinada língua, léxico é o universo de todos os seus itens lexicais que seus falantes utilizam, já utilizaram ou poderão vir a utilizar (SCAPINI, 1995).

Alguns autores argumentam que o termo "dicionário" carrega tipicamente impresso o significado de vocabulário *(wordbook)* para leitores humanos (GUTHRIE et al., 1996). Em alguns casos, utiliza-se o termo "léxico" para identificar o componente de um sistema de PLN com informações semânticas e gramaticais sobre itens lexicais. Também, usa-se a expressão "base de dados lexical" como sendo uma coleção de informações lexicais, apresentadas em formato estruturado e acessível a sistemas de PLN.

De qualquer forma, o propósito dos dicionários (ou léxicos) é prover uma grande gama de informações sobre palavras, como etimologia, pronúncia, morfologia, sintaxe, entre outras. Eles fornecem definições de seus sentidos e, em decorrência disso, produzem conhecimento não apenas sobre a linguagem, mas sobre o próprio mundo (GUTHRIE et al., 1996).

2.2.3 Sintático

Fazem parte do processamento morfo-sintático, a análise morfológica e a análise sintática. A morfologia e a sintaxe tratam da constituição das palavras e dos grupos de palavras que formam os elementos de expressão de uma língua. Enquanto o analisador léxico-morfológico lida com a estrutura das palavras e com a classificação das mesmas em diferentes categorias, o analisador sintático trabalha em nível de agrupamento de palavras, analisando a constituição das frases.

A análise sintática *(parsing)* é o procedimento que avalia os vários modos de como combinar regras gramaticais, com a finalidade de gerar uma estrutura de árvore que represente a estrutura sintática da sentença analisada. Se a sentença for ambígua, o analisador sintático (parser) irá obter todas as possíveis estruturas sintáticas que a representam.

O papel do processamento sintático varia em importância (NUNES et al., 1999). Ele tradicionalmente ocupa posição de destaque, com a semântica sendo considerada uma interpretação da sintaxe. Mas, também, pode ser considerado em posição secundária, de acordo com os pesquisadores denominados semântico-gerativistas. Neste último caso, a sintaxe é uma projeção da semântica. Entretanto, qualquer que seja a visão adotada, o processamento sintático é uma etapa indispensável para viabilizar o processamento semântico, que passamos a discutir.

2.2.4 Semântico

Enquanto a sintaxe corresponde ao estudo de como as palavras agrupam-se para formar estruturas em nível de sentença, a semântica está relacionada ao sentido, não só de cada palavra,

mas também do conjunto resultante delas. O processamento semântico é considerado um dos maiores desafios do PLN, pois se vincula, de um lado, com a morfologia e a estrutura sintática e, de outro lado em alguns casos, com informações da pragmática (SAINT-DIZIER, 1999).

Segundo o princípio da composicionalidade, o significado de qualquer construção em uma linguagem depende do significado de cada um dos seus componentes (ALLEN, 1995). Assim, o significado de uma frase, por exemplo, origina-se do significado de cada palavra. Este princípio revela a importância das relações que ocorrem entre os itens lexicais. Quando essas conexões ligam elementos de domínios semânticos, tem sido usual denominá-las "relações semânticas", enquanto as ligações entre itens lexicais são tratadas como "relações lexicais". Entretanto, quando não é possível ou é desnecessária a distinção, é adotado o termo "relações semânticas lexicais" (EVENS, 1992).

As palavras podem se associar através de dois tipos de relações: paradigmáticas e sintagmáticas. Entre as relações paradigmáticas estão: sinonímia, antonímia, hiponímia, hiperonímia (em sentido contrário da hiponímia), meronímia (relação entre um holônimo, que representa o todo, e um merônimo, que representa a parte), holonímia (em sentido contrário da meronímia), implicatura e pressuposição. A implicatura é a relação entre A e B, quando B só é verdadeiro se A também for. A pressuposição é a relação entre A e B, quando B é verdadeiro se A ou a negação de A forem verdadeiros.

As relações paradigmáticas associam palavras através do significado, como "nadar" e "água". As relações sintagmáticas conectam palavras que são frequentemente encontradas no mesmo discurso, como "água" e "poça".

As associações de termos (RUSSEL; NORVIG, 1995) englobam diferentes tipos de relações semânticas lexicais, como a sinonímia (exemplo: "recipiente" e "receptáculo"), a hiponímia (exemplo: "reservatório" e "tanque"), a meronímia (exemplo: "carro" e "tanque"), a antonímia (exemplo: "aceleração" e "desaceleração") e a compatibilidade (exemplo: "carro" e "dirigir"), entre outras. Estão incluídos nestas classes de relacionamentos tanto os paradigmáticos quanto os sintagmáticos.

O processo composicional é um dos principais problemas do processamento semântico. Assim, como o significado de um constituinte de uma sentença depende dos significados dos seus sub-constituintes, os significados destes podem, por sua vez, serem determinados por regras gramaticais.

2.3 Aprendizagem de Máquina

Há décadas os seres humanos perderam o monopólio sobre as decisões que afetam sua vida cotidiana, pois cada vez mais as compartilham com algoritmos capazes de encontrar padrões escondidos em dados ruidosos, aprender modelos genéricos a partir de instâncias de treinamento

e tomar decisões ótimas e sub-ótimas considerando benefícios e custos parcialmente conhecidos.

Aprendizado de máquina é uma área da Inteligencia Articial (IA) que engloba o estudo e a construção de sistemas inteligentes a partir de dados(MOHRI; ROSTAMIZADEH; TALWALKAR, 2013). Após efetuado o aprendizado, também denominado treinamento, um sistema pode ser utilizado para classicar ou estimar saídas para instâncias desconhecidas. (SIMON, 2013) deniu o aprendizado de máquina como "o campo de estudos que fornece a computadores a habilidade de aprenderem sem serem explicitamente programados".

> Na maioria dos problemas de aprendizado, a tarefa é aprender a classificar entradas de acordo com um conjunto finito (ou, às vezes, infinito) de classificações. Tipicamente, um sistema de aprendizado é dotado de um conjunto de dados de treinamento, que foram classificados manualmente. O sistema, então, tenta aprender, a parir desses dados de treinamento, a como classificar estes mesmos dados (geralmente, uma tarefa relativamente fácil) e também como classificar novos dados ainda não observados. (COPPIN, 2017).

Com a finalidade de obter um sistema de aprendizado capaz de representar computacionalmente um determinado problema, bem como a sua solução, é necessário descrever objetos, processos e situações que fazem parte do seu domínio (MORNARD; BARANAUKAS, 2000).

(WITTEN; FRANK, 1999) descreveram quatro conceitos caracterizando os vários algoritmos de aprendizado de máquina: aprendizado por classicação, onde um conjunto de exemplos pertencentes à classes são utilizados para construir modelos; aprendizado por associação, onde deseja-se identicar grupos de um ou mais atributos que determinam o valor de classe de uma instância; aprendizado por agrupamento, no qual exemplos semelhantes de acordo com um critério estabelecido são agrupados e aprendizado por regressão, que possui como objetivo desenvolver um modelo matemático correlacionando atributos com o valor de classe.

Dentre as diversas técnicas desenvolvidas na área de aprendizado de máquina, foram consideradas para este estudo as técnicas mais recorrentes em pesquisas correlatas para tarefas relacionadas ao problema.

3

Trabalhos Relacionados

Este trabalho explorou dois diferentes pontos de vista. Inicialmente foi realizada uma investigação do estado da arte por meio de uma pesquisa em bases de trabalhos científicos, posteriormente foram exploradas as ferramentas e soluções existentes no mercado por meio de uma revisão de mercado.

3.1 Revisão sistemática dos estudos do estado da arte

Uma revisão literária sistemática é um meio de identicar, avaliar e interpretar todas as pesquisas disponíveis relevantes a uma determinada questão de pesquisa, ou área de um tópico, ou fenômeno de interesse. Estudos individuais que contribuem para uma revisão sistemática são chamados estudos primários; uma revisão sistemática é uma forma de estudo secundário (KITCHENHAM; CHARTERS, 2007).

Um estudo de revisão sistemática provê uma estrutura do tipo de relatórios de pesquisa e resultados que foram publicados através de sua categorização e, geralmente, isso é fornecido por um sumário visual de seus resultados. (KITCHENHAM; CHARTERS, 2007) define revisão sistemática como um método para construir esquemas de classificação e estruturar uma determinada área de interesse. Desta forma, a metodologia do presente trabalho é baseada no modelo exposto por (KITCHENHAM; CHARTERS, 2007) que é apresentado em 3 etapas principais: planejamento da revisão e busca de estudos primários[1]; seleção de estudos relevantes[2] e, extração e análise dos resultados.

[1] São todos os estudos encontrados como resultado à aplicação da *string* de busca nas bases de dados (PETERSEN et al., 2008).

[2] São os estudos resultantes da aplicação dos critérios de inclusão e exclusão que são relevantes para responder as questões de pesquisa (PETERSEN et al., 2008).

3.1.1 Planejamento da Revisão

Nesta primeira fase de planejamento, será informado o propósito e os objetivos do trabalho; as fontes, estratégias e questões de pesquisa; os critérios e procedimentos de seleção dos resultados; e a forma de extração dos dados resultantes deste processo.

3.1.1.1 Questões de Pesquisa

As questões de pesquisa em uma revisão sistemática visam descobrir tendências de pesquisa (p.ex., tendência de publicação ao longo do tempo, tópicos cobertos na literatura etc.) (KITCHENHAM; CHARTERS, 2007). Desta forma, as questões de pesquisa podem ser definidas como mecanismos utilizados para auxílio na estruturação de uma determinada área de pesquisa. Portanto, visando atender os objetivos definidos nesta pesquisa e baseado no conjunto de palavras-chave foram estabelecidas as seguintes questões de pesquisa conforme apresentadas no Quadro 1.

Quadro 1: Questões de pesquisa definidas para o estudo

Q1) Quais estudos sobre *chatbots*, armazenamento e recuperação de dados existem atualmente e suas principais características?
Q2) Como *chatbots* acessam dados ou diálogos a partir de uma base de dados?
Q3) Como *chatbots* armazenam dados e fluxos de conversa a partir de uma base de dados preexistente?
Q4) Qual(s) algoritmos, ferramentas, técnicas e/ou métodos de aprendizagem de máquina e/ou processamento de linguagem natural são utilizados por *chatbots*?

Fonte: esta pesquisa

3.1.1.2 Estratégias de busca

Para encontrar o maior número possível de fontes, uma revisão sistemática utiliza uma estratégia de busca bem definida e clara, incluindo *strings* de busca compostas por palavras-chave, relacionadas com as questões de pesquisa e por lógica booleana (KITCHENHAM; CHARTERS, 2007). Para tal, a estratégia de busca deve ser menos restritiva de modo a permitir recuperar mais estudos (KITCHENHAM; BRERETON; BUDGEN, 2011).

Para esta revisão foram utilizadas as seguintes bases de dados: *SciVerse Scopus*, *IEEE Xplorer*, *ACM Digital Library* , *Science Direct*, *Web Of science* e *Compendex*. A *Scopus* é um indexador para as principais bibliotecas digitais de editoras. As bibliotecas digitais de editoras contêm basicamente os artigos publicados apenas por elas, enquanto os indexadores indexam artigos de várias bibliotecas digitais.

Como essas base de dados são proprietárias, foi utilizado o portal de periódicos da CAPES [3] para acesso remoto via instituição Fundação Universidade Federal de Sergipe (FUFS)

[3] Disponível em: <http://www.periodicos.capes.gov.br>. Acesso em: 28 de janeiro de 2019

de forma que não houvesse restrições de download.

Assim, neste trabalho, foi definida a seguinte *string* de busca: (*chatbot OR chatterbot OR virtual assistant OR chat-agent OR intelligent conversational agent) AND (natural language OR processing OR ontology OR machine learning OR data recovery*).

A *string* de busca selecionada foi executada em 28 de janeiro de 2019 nas bases de dados predefinidas.

3.1.1.3 Critérios de Seleção

De acordo com as questões de pesquisa e com o objetivo da revisão foram denidos critérios de inclusão e exclusão relevantes para o tema da pesquisa com o objetivo de nortear a seleção dos artigos na fase de seleção dos estudos. Logo, foram elaborados 4 critérios de inclusão (I) e 4 critérios de exclusão (E) como descritos abaixo.

Os critérios de inclusão considerados na seleção dos estudos:

I1. O estudo apresenta algum mecanismo de armazenamento e/ou recuperação de dados estruturados.

I2. O estudo apresenta alguma arquitetura, técnica ou modelo para recuperação de dados estruturados por *chatbots*.

I3. O estudo fornece a descrição de algum mecanismo de armazenamento de dados estruturados a partir de uma base de dados preexistentes.

I4. O estudo apresenta algum algoritmo genérico e/ou método de aprendizagem de máquina e/ou processamento de linguagem natural para recuperação de dados por *chatbots*.

Os critérios de exclusão considerados na seleção dos estudos foram:

E1. O estudo não possui nenhum tipo de informação pertinentes ao escopo deste trabalho.

E2. O estudo não apresenta um modelo claro e completo de armazenamento de dados estruturados.

E3. O estudo não dispõe uma arquitetura para recuperação de dados estruturados legível ou completa.

E4. No estudo a temática armazenamento de dados ou fluxos de conversa por *chatbots* não é abordada.

3.1.2 Seleção dos Estudos

O Quadro 2 traz a *String* de busca aplicada a cada base de dados e seus respectivos filtros. Determinou-se um período limite nas bases para refinar os resultados de forma que fossem retornados os artigos mais recentes, para isso foi denido um ltro de data padrão de 2000 - 2019.

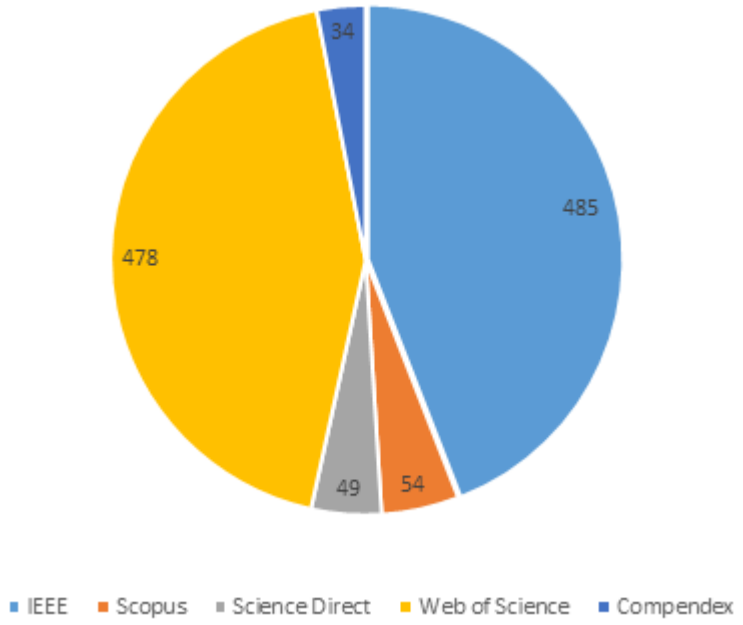
Quadro 2: *String* de busca aplicada a cada base.

IEEE Xplorer	(*(chatbot OR chatterbot OR virtual assistant OR chat-agent OR intelligent conversational agent) AND (natural language OR processing OR ontology OR machine learning OR data recovery) Filters Applied*: 2000-2019)
Scopus	(*(chatbot OR chatterbot OR virtual assistant OR chat-agent OR intelligent conversational agent) AND (natural language OR processing OR ontology OR machine learning OR data recovery) Filters Applied*: 2000-2019)
Science Direct	(*(chatbot OR chatterbot OR virtual assistant OR chat-agent OR intelligent conversational agent) AND (natural language OR processing OR ontology OR machine learning OR data recovery) Year*: 2000-2019)
Web of science	*TOPIC:(((chatbot OR chatterbot OR virtual assistant OR chat-agent OR intelligent conversational agent) AND (natural language OR processing OR ontology OR machine learning OR data recovery)))Timespan:2000-2019. Indexes: SCI-EXPANDED, SSCI, AHCI, CPCI-S, CPCI-SSH, ESCI.*
Compendex	*((chatbot OR chatterbot OR virtual assistant OR chat-agent OR intelligent conversational agent) AND (natural language OR processing OR ontology OR machine learning OR data recovery))*

Fonte: esta pesquisa

Dessa forma, a distribuição de estudos por base ficou como apresentado na figura 1, sendo a *IEEE explorer* com 485 artigos a base com o maior número de resultados, seguido da *Web of Science* com 478, *Scopus* com 54, *Science direct* com 49 e, por fim, *Compendex* com 34 estudos.

Figura 1 – Estudos primários encontrados por base de dados.



Fonte: esta pesquisa

O processo de seleção dos estudos relevantes se caracterizou em etapas, isto é, neste processo foram lidos os títulos, palavras-chave e resumos, respectivamente, pautado pelos critérios de seleção definidos na subseção 3.1.1.3. A Quadro 3 mostra este processo de seleção de forma detalhada.

Quadro 3: Etapas de seleção dos estudos relevantes

Total	1.100
Análise de títulos	57
Análise de palavras-chave	55
Análise de resumo	21

Fonte: esta pesquisa

Portanto, após a etapa de seleção dos estudos relevantes, 21 no total, os mesmos foram encaminhados para análise aprofundada e exposição sistemática dos resultados.

3.1.3 Extração e Análise dos Resultados

Na etapa de extração todos os 21 estudos selecionados foram examinados integralmente e classificados de acordo com os critérios de inclusão, definidos na subseção 3.1.1.3, conforme apresentado no Quadro 4.

É importante notar que existem estudos apresentados no Quadro 4 que podem está atrelado a mais de um critério de inclusão, visto que, este pode satisfazer um ou mais critérios.

Quadro 4: Classificação dos estudos selecionados por critérios de inclusão

I1	E1, E4, E9, E10, E13, E19, E20
I2	E1, E2, E3, E4, E7, E12, E15, E16,
I3	E11, E13, E17, E18
I4	E3, E4, E5, E6, E7, E8, E9, E10, E11, E13, E14, E19, E21

Fonte: esta pesquisa

Para responder a primeira questão de pesquisa Q1 os estudos foram identificados e apresentado uma breve síntese de pontos principais de cada estudo conforme as tabelas 1 e 2, respectivamente.

Tabela 1 – Identificação dos Estudos

E1	CONTEXT SENSITIVE CONVERSATIONAL AGENT USING DNN	(KARVE et al., 2018)
E2	Interactive Intelligent Agent Architecture	(AMEUR; HEUDIN, 2006)
E3	Generic natural language command interpretation in ontology-based dialogue systems	(MAZUEL; SABOURET; CHOI, 2006)
E4	The ChatBot Feels You – A Counseling Service Using Emotional Response Generation	(LEE; OH, 2017)
E5	Machine-Learned Ranking based Non-task-oriented Dialogue Agent using Twitter Data	(KOSHINDA; INABA; TAKAHASHI, 2015)
E6	Can Proactive Behavior turn Chatterbots into Conversational Agents?	(L'ABBATE; THIE; KAMPS, 2005)
E7	Generic command interpretation algorithms for conversational agents	(MAZUEL; SABOURET, 2006)
E8	MAKING IT REAL: LOEBNERWINNING CHATBOT DESIGN	(WILCOX; WILCOX, 2013)
E9	Conversation-Based Natural Language Interface to Relational Databases	(OWDA; BANDAR; CROCKETT, 2007)
E10	Chatbot using TensorFlow for small Businesses	(SINGH et al., 2018)
E11	A fuzzy logic system for classifying the contents of a database and searching consultations in natural language	(GóMEZ; ROPERO; LEóN, 2006)
E12	A Personal Agents Hybrid Architecture for Question Answering featuring Social Dialog	(CORONADO; IGLESIA; MARDOMINGO, 2015)
E13	Problem Solving Chatbot for Data Structures	(SHAH et al., 2018)
E14	Intelligent Chatbot for Easy Web-Analytics Insights	(RAVI, 2018)
E15	A Modular Framework for Versatile Conversational Agent Building	(AUGELLO et al., 2011)
E16	Exploring Ontologies to Improve the Empathy of Interactive Bots	(JUSTO et al., 2018)
E17	Intelligent Web Interface using Flexible Conversational Agent with Semantic Bayesian Networks	(KIM; HONG; CHO, 2005)
E18	OntBot : Ontology based ChatBot	(AL-ZUBAIDE; ISSA, 2016)
E19	A STANDALONE GENERATIVE CONVERSATIONAL INTERFACE USING DEEP LEARNING	(VARGHESE; PILLAI, 2018)
E20	An Efficient Search for Context-Based Chatbots	(ATIYAH; JUSOH; ALMAJALI, 2018)
E21	Automatically Extracting Dialog Models from Conversation Transcripts	(NEGI et al., 2009)

Fonte: este trabalho

Tabela 2 – Síntese dos estudos

Estudos	**Descrição**
E1	Neste trabalho, (KARVE et al., 2018) investigam um método para construir um agente conversacional inteligente de domínio fechado usando redes neurais profunda. Este agente de conversação usa um modelo baseado em recuperação que identifica a intenção da consulta do usuário de entrada e a mapeia para uma base de conhecimento para retornar resultados apropriados. Assim, O agente de conversação gera respostas de acordo com o contexto atual de conversação, permitindo conversas mais humanas.
E2	O estudo de (AMEUR; HEUDIN, 2006) propõe um modelo de agente conversacional que se baseia na arquitetura de subsunção de vários agentes. Os agentes de comportamento são representados por classificadores. Neste estudo, utilizou-se uma hierarquia etológica de agentes do agente reflexivo ao agente cognitivo. Um agente estratégico também foi desenvolvido para controlar todo o diálogo, em uma arquitetura proativa. Para fazer esse agente conversacional evoluir, usa-se um sistema classificador baseado em expressões tipo *lisp*.
E3	Neste estudo, (MAZUEL; SABOURET; CHOI, 2006) apresentam uma arquitetura geral para uma abordagem mais genérica dos agentes conversacionais. Essa arquitetura contém módulos de linguagem natural genéricos (independentes do aplicativo) que são baseados em ontologias para interpretação de comandos. Focou-se na apresentação dos módulos geradores de eventos e gerenciadores de diálogo, que contam com uma abordagem *bottom-up* para combinar o comando do usuário com o conjunto de ações possíveis no momento.
E4	Neste estudo, (LEE; OH, 2017) sugerem a introdução de um novo sistema de *chatbot* para o serviço de aconselhamento psiquiátrico. Esse sistema entende o conteúdo da conversação com base em métodos recentes de processamento de linguagem natural (NLP) com reconhecimento de emoções. Ele percebe o fluxo emocional através da observação contínua da conversa. Além disso, gera-se respostas de aconselhamento personalizadas a partir da entrada do usuário, para isso, usa-se restrições adicionais ao modelo de geração para a geração de respostas adequadas que podem ser detectadas no contexto de conversação, na emoção do usuário e na reação esperada.

Continua na próxima página

Tabela 2 – Continuação da página anterior

Estudos	**Descrição**
E5	Neste trabalho, (KOSHINDA; INABA; TAKAHASHI, 2015) descrevem um método para desenvolver um agente de diálogo não orientado à tarefa (também chamado de agentes de diálogo conversacional ou orientado a bate-papo) que pode abranger uma ampla gama de tópicos. Esse método extrai um tópico do enunciado de um usuário e adquire elocuções de candidatos que contêm o tópico do Twitter. Assim, este agente seleciona um enunciado adequado para contexto de diálogo de candidatos usando o método de classificação aprendido por máquina. Os resultados de um experimento demonstram que um agente de diálogo baseado no método proposto pode conduzir uma conversa mais natural e prazerosa em comparação com outros agentes de diálogo.
E6	*Chatterbots* são sistemas de software que interagem com seus usuários por meio de uma conversação baseada em linguagem natural. Neste estudo, (L'ABBATE; THIE; KAMPS, 2005) sugerem uma melhoria da tecnologia do *chatterbot*, baseada na implementação de um comportamento de diálogo mais pró-ativo. Esse *chatterbot* aprimorado com a proatividade pode ser considerado como um agente de conversação inteligente que geralmente é caracterizado por uma abordagem de implementação mais complexa, mas fornece um controle de diálogo mais eficiente por meio de estratégias de iniciativa mista.
E7	A comunicação homem-máquina no âmbito de agentes inteligentes torna-se cada vez mais frequente. Neste trabalho, (MAZUEL; SABOURET, 2006) Propõem uma arquitetura genérica fornecida com um algoritmo de linguagem natural (LN) para interpretação de comandos que pode ser adaptada para diferentes domínios de agentes. Essa arquitetura de LN normalmente depende de uma área problemática e de uma grande área deontológica. Consideramos as abordagens clássicas para a interpretação do comando do LN: a abordagem *top-down*, que se baseia nos constrangimentos sintáticos do modelo do agente, e a abordagem *bottom-up* que se baseia no conjunto de possíveis ações do agente. Proprõe-se combinar as duas abordagens em um algoritmo baseado em *bottomup* que faz uso das restrições do agente.
E8	Neste estudo, (WILCOX; WILCOX, 2013) discutem brevemente o *ChatScript*, uma linguagem de *script* de linguagem natural de código aberto e o mecanismo que executa os *bots* Suzette, Rosette, Angela. Em seguida, analisa como foram construídos tais *chatbots* e o que aprendeu-se.

Continua na próxima página

Tabela 2 – Continuação da página anterior

Estudos	**Descrição**
E9	Neste artigo, (OWDA; BANDAR; CROCKETT, 2007) propõem uma nova abordagem para criar interfaces de linguagem natural baseadas em conversação para bancos de dados relacionais, combinando agentes conversacionais orientados a objetivos e árvores de conhecimento. Os agentes de conversação orientados por objetivos provaram sua capacidade de desambiguar as necessidades do usuário e conversar dentro de um contexto (ou seja, domínio específico). Árvores de conhecimento usadas para superar a falta de conectividade entre o agente de conversação e o banco de dados relacional, através da organização do conhecimento de domínio em árvores de conhecimento. As árvores de conhecimento também funcionam como um roteiro para o fluxo de diálogo do agente conversacional. O *framework* proposto torna mais fácil para engenheiros de conhecimento desenvolver um NLI-RDB baseado em conversação confiável. O sistema protótipo desenvolvido mostra excelente desempenho em consultas comuns (ou seja, consultas extraídas do especialista por um engenheiro de conhecimento). O usuário terá uma interface amigável que pode conversar com o banco de dados relacional.
E10	o estudo de (SINGH et al., 2018) demonstra um método de desenvolvimento de *chatbots* que podem seguir o contexto da conversa. Este método usa o *TensorFlow* para desenvolver o modelo de rede neural dos *chatbots* e usa as técnicas de processamento de linguagem natural (NLP) para manter o contexto da conversação.
E11	Neste estudo, (GóMEZ; ROPERO; LEóN, 2006) apresentam um método para a classificação de conteúdos em um banco de dados, a fim de responder às consultas do usuário usando a linguagem natural. A inteligência artificial (IA) é usada para relacionar essas consultas ao conteúdo do banco de dados. O sistema é baseado em um mecanismo de lógica difusa para aproveitar suas propriedades tão adequadas a essa aplicação e é ideal para conjuntos de conhecimento acumulado que podem ser construídos em níveis hierárquicos por uma estrutura de árvore. Por fim, O objetivo final deste sistema é a implementação de um assistente virtual da *Web* para um portal da Internet.
A12	Neste artigo, (CORONADO; IGLESIA; MARDOMINGO, 2015) propõem uma arquitetura híbrida para um sistema de perguntas que apresenta diálogo social. Afirmamos que incluir diálogo social em sistemas de controle de qualidade aumenta a satisfação dos usuários e faz com que eles se envolvam facilmente com o sistema. Por fim, apresentamos uma avaliação que suporta essas hipóteses.

Continua na próxima página

Tabela 2 – Continuação da página anterior

Estudos	**Descrição**
A13	Neste trabalho, (SHAH et al., 2018) apresentam um sistema que pode resolver problemas de estrutura de dados usando rede neural profunda (DNN). Com um determinado conjunto de dados, o sistema pode fornecer serviços para acessar dados em formato, como matrizes, pilhas, filas e árvores. Com base nessas estruturas de dados, podemos resolver problemas como cruzar listas, inverter números e traduzir a linguagem de divergências sintáticas. O serviço de aprendizado não é um programa algorítmico e sim um modelo treinado usando o DNN. Com a implementação do *chatbot* de solução de problemas, ele entenderá como organizar e recuperar dados com base na escolha da estrutura de dados do usuário. Usa-se também o *Neural Stack Machine* (NSM) com *Recurrent Neural Network* (RNN) como o controlador.
E14	Neste estudo, (RAVI, 2018) compara duas ferramentas de análise amplamente usadas com base em sua facilidade de uso. À luz do mesmo, propõe um *chatbot* acionado pelo *AIML (Artificial Intelligence Machine Learning)*, alimentado com dados brutos de análise, que permitirá aos usuários de *bot* obter *insights* de negócios apenas digitando a consulta.
E15	Neste artigo, (AUGELLO et al., 2011) ilustram uma infraestrutura baseada na Web de uma arquitetura para agentes de conversação equipados com uma base de conhecimento modular. Essa solução tem a vantagem de permitir a construção de módulos específicos que lidam com características particulares de uma conversa (variando de seu tópico à maneira de raciocinar do *chatbot)*. Isso aprimora os recursos de interação do agente. A abordagem simplifica o processo de design da base de conhecimento do *chatbot*: estender, generalizar ou mesmo restringir a base de conhecimento do *chatbot,* a fim de adequá-lo ao gerenciamento de tarefas de diálogo especícas, tanto quanto possível.
E16	Neste artigo, (JUSTO et al., 2018) propõem uma arquitetura de software que permite a interpretação baseada em ontologias de vários tipos de dados (áudio, vídeo e texto) do ambiente do *bot*. Definimos regras formais baseadas em conceitos para expressar o comportamento afetivo com o objetivo de melhorar a empatia dos *bots*. A técnica proposta baseia-se em tecnologias semânticas, como as linguagens OWL e SWRL.

Continua na próxima página

Tabela 2 – Continuação da página anterior

Estudos	**Descrição**
E17	Neste trabalho, (KIM; HONG; CHO, 2005) propõem um modelo de redes *bayesianas* semânticas que inferem a intenção do usuário com base em redes *bayesianas* e suas informações semânticas representando a relação entre nós. Como a conversa geralmente contém expressões ambíguas, o gerenciamento do contexto ou a incerteza deve ser necessário para oferecer suporte a agentes de conversação flexíveis. O método proposto impulsiona a interação de iniciativa mista (MII), que solicita a falta de conceitos e esclarece conceitos espúrios para entender corretamente a intenção do usuário. Assim, implementando um guia de informações da Web com o método proposto, confirmou-se a utilidade do método proposto.
E18	Neste estudo, (AL-ZUBAIDE; ISSA, 2016) propõem uma nova abordagem baseada em ontologia para modelar e operar os *chatbots (OntBot)*. O *OntBot* usa a técnica de mapeamento apropriada para transformar ontologias e conhecimento em banco de dados relacional e, em seguida, usar esse conhecimento para conduzir seus bate-papos. A abordagem proposta supera uma série de desvantagens tradicionais dos *chatbots,* incluindo: a necessidade de aprender e usar a linguagem específica do *chatbot*, como AIML, alta interferência de *botmaster* e o uso de tecnologia não amadurecida. O *OntBot* tem o poder adicional de interações fáceis dos usuários usando sua linguagem natural e o suporte contínuo de diferentes domínios de aplicativos. Isso dá à abordagem proposta várias propriedades exclusivas de escalabilidade e interoperabilidade.
E19	Hoje, geralmente todas as interfaces conversacionais geradoras desenvolvidas estão utilizando o aprendizado profundo. Neste artigo, (VARGHESE; PILLAI, 2018) têm como objetivo principal atualizar uma interface de conversação gerativa usando o aprendizado profundo. Além disso, a estrutura sugerida surge na formação de respostas baseadas na base de conhecimento dinâmico e na entrada atual em um domínio fechado.
E20	Neste estudo, (ATIYAH; JUSOH; ALMAJALI, 2018) objetivam introduzir um novo método chamado *B-Point Tree* para aumentar a eficiência da busca por respostas precisas, adicionando uma estrutura de dados adicional ao algoritmo BST tradicional. O desempenho da Árvore B-Point foi comparado ao *BST* tradicional, resultados de experimentos sugerem que o *B-Point Tree* supera o *BST t*radicional. Os resultados sugerem que a Árvore *B-Point* é capaz de aumentar a eficiência de um *chatbot.*

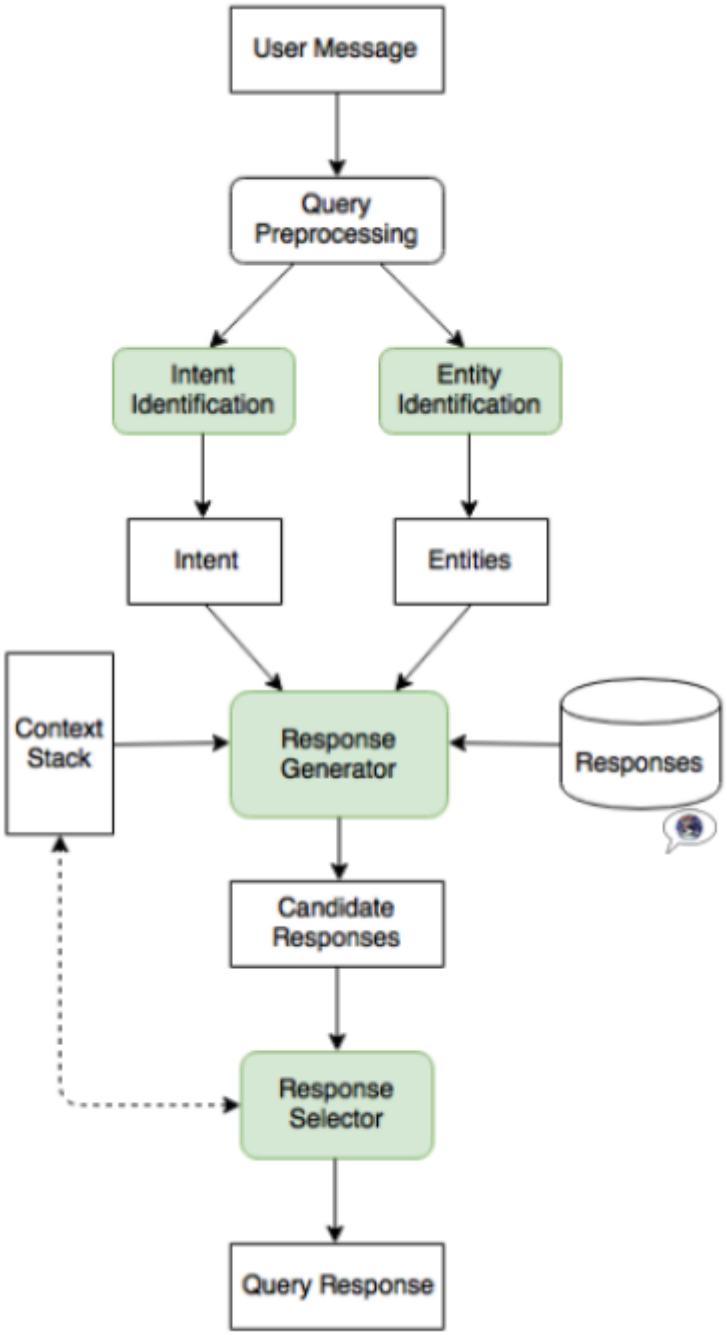
Continua na próxima página

Tabela 2 – Continuação da página anterior

Estudos	**Descrição**
E21	Neste trabalho, (NEGI et al., 2009) buscam adquirir conhecimento específico de uma tarefa, definindo a sub-tarefa como a unidade-chave de uma conversa orientada por tarefas. Propomos um algoritmo não supervisionado, como a priori, que extrai as sub-tarefas e seus ordenamentos válidos de conversas humanas anônimas não anotadas. A modelagem de diálogos como uma combinação de sub-tarefas e suas ordenações válidas captura facilmente a variabilidade nas conversas. Ele também nos fornece a capacidade de mapear nosso modelo de diálogo para construções *AIML* e, portanto, usar intérpretes *AIML* padrão para construir *chatbots* orientados a tarefas.

Em sequência, para responder as questões de pesquisa Q2 e Q3 foi selecionado o estudo E1, visto que dentre os 21 estudos selecionados, somente este apresentou uma arquitetura completa, atual e detalhada sobre armazenamento e recuperação de dados para o contexto de *chatbots* sobre um domínio fechado. A seguir detalhamos essa arquitetura:

Figura 2 – Arquitetura



Fonte: (KARVE et al., 2018)

Inicialmente o usuário fornece uma consulta a este sistema que a interpreta e produz uma resposta adequada. Todo o sistema pode ser dividido em três diferentes subsistemas:

1. Pré-processamento

   O pré-processamento de consulta refere-se ao conjunto de ações que são executadas na consulta para operações adicionais e compreende as seguintes operações:

a) Tokenização: tokenização é o processo de quebra de sentenças ou consultas de usuário em um conjunto de palavras chamado 'token';

b) Normalização: normalização é o processo de correção de erros ortográficos e tipográficos comuns em um determinada consulta do usuário;

c) Eliminação de palavras irrelevantes: stop-words são palavras que não têm efeito no processo de classificação. stop-words incluem artigos como 'a', 'e', pronomes como 'meu', 'seu' etc. As stop-words também podem ser chamadas de 'ruído' em uma determinada consulta de usuário. Essas palavras não contribuem para o significado geral da sentença e, portanto são eliminadas;

d) Identificação das partes do discurso e identificação da entidade: A marcação de partes da fala é o processo de marcar cada token com seu papel na frase de parte do discurso, como substantivo, verbos etc. Identificação de entidade é o processo de identificação ou entidades especiais em uma determinada frase, por exemplo, um organização, local, nome, datas etc. Isso nos ajuda a identificar palavras significativas que fornecem informações suplementares para a intenção da consulta;

e) stemização: é o processo de reduzir as palavras às suas raízes. Por exemplo, correndo torna-se correr, planejado se torna plano etc. Isso ajuda na identificação de grande número de palavras sem ter que manter uma lista de todos os morfemas de cada palavra;

f) Incorporação de vetores: a lista de *token* derivado é então convertido em vetores de 1 e 0. Isso é feito comparando-se as palavras de entrada com a lista de palavras derivadas únicas obtidas do conjunto de dados de treinamento. Esse vetor de 1s e 0s é usado como entrada para a rede neural.

2. Identificação de intenção:

   Identificação de intenção é o processo de reconhecer ou classificar o motivo ou a intenção do usuário por trás da consulta. Isso é realizado usando uma rede neural profunda de várias camadas. O vetor de consulta é fornecido como entrada para esta rede neural. O modelo de rede neural produz uma distribuição de probabilidade de possíveis intenções de consulta. Aquela com a maior probabilidade é considerado o objetivo de uma determinada consulta de usuário.

   a) Rede Neural Artificial de Treinamento: o multi-camadas rede neural é treinado com um conjunto de dados de amostra consistindo de conjuntos de diálogos comuns e consultas gerais que os usuários conversam com agente de conversação. Esses conjuntos de diálogos são classificados em diferentes categorias. A categoria identificada é usada para representar o rótulo dessa frase. O conjunto de dados é dividido em duas partes chamadas recursos e etiquetas; ambos traduzidos em formato de vetor (incorporação de vetores). Esse conjunto de dados está instalado na rede neural. A

rede neural resultante atua como modelo classificador no sistema proposto e isso identifica a intenção da consulta do usuário;

3. Tratamento de Contexto e Geração de Respostas

   A manipulação de contexto é implementada usando um simples mecanismo de pilha. Toda resposta está associada a um certo contexto. Assim, quando uma resposta é gerada, o contexto correspondente é empurrado para pilha. A qualquer momento, a parte superior da pilha indicará o contexto atual do bate-papo. Esse contexto corrente é usado como um filtro para selecionar a resposta correspondente à intenção da consulta. Se nenhuma resposta satisfazer o filtro de contexto, a pilha de contexto é exibida para recuperar o contexto anterior. Agora, as respostas são pesquisadas com o contexto pop-up como filtro. Esse processo é repetido até que uma resposta apropriada seja encontrada.

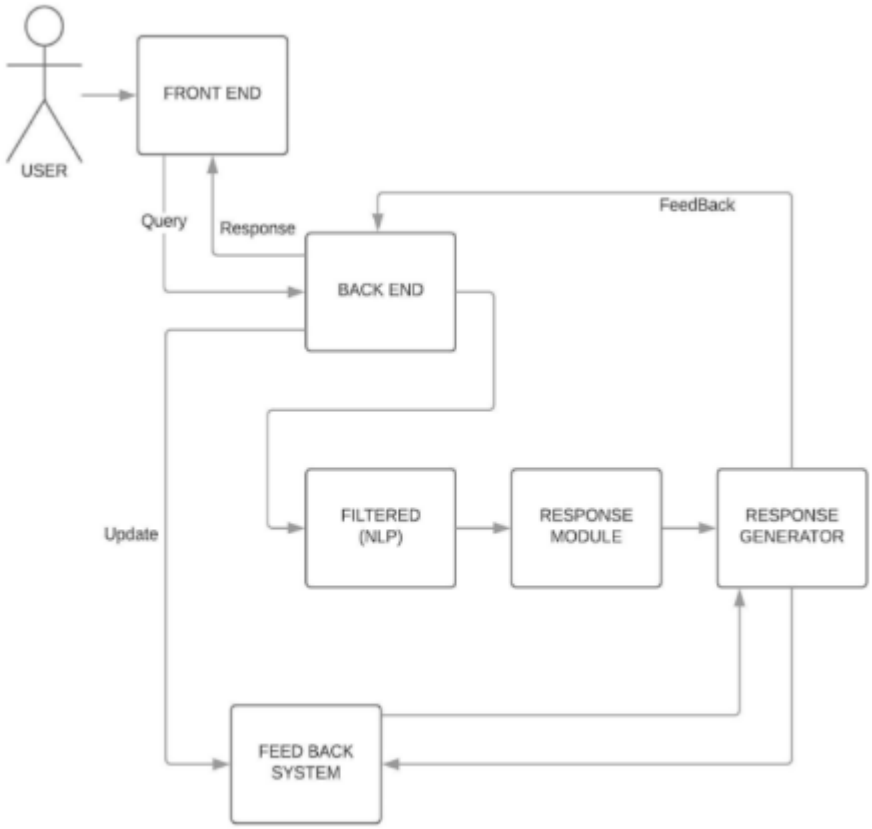
(KARVE et al., 2018) também trazem um algoritmo em alto nível para a arquitetura proposta, no entanto , o mesmo não foi descrito neste trabalho, visto que o objetivo das questões de pesquisa Q2 e Q3 é identificar uma arquitetura.

Por fim, para responder a última questão de pesquisa Q4 utilizamos o estudo E10. Nesse estudo (SINGH et al., 2018) demonstram um método de desenvolvimento de *chatbot* que pode seguir o contexto da conversa. Este método usa *TensorFlow*[4] para o desenvolvimento do modelo de rede neural do *chatbot* e usa as técnicas de PLN para manter o contexto da conversação. Tal método divide-se em 3 partes principais:

1. Interface de usuário
2. Modelo de rede neural e unidade de PLN
3. *Feedback* do sistema

[4] Disponível em: https://www.tensorflow.org/

Figura 3 – Arquitetura



Fonte: (SINGH et al., 2018)

Por simplicidade, neste estudo, analisaremos somente o item 2, visto que, esse módulo é o núcleo principal de todo sistema, isto é, a parte que gera a resposta real para a consulta do usuário. Primeiro o modelo é definido usando o *TensorFlow*, que é treinado usando o arquivo de intenção criado. O arquivo de intenção está no formato de arquivo *json* como se segue:

Figura 4 – código *Json*

```
{"tag": "admission_enquiry",
 "patterns": ["i have some doubts regarding process", "I want
  to know about admission process", "enquiry about
  admission"],
 "responses": ["Okay, what do you wanna know?", "what are
  you doubts?", "Okay, you can ask"]
},
{"tag": "application_date",
  "patterns": ["when do i apply?", "What is the due date for
   application?", "what is last date of the application?","what
   is starting date of the application?" ],
  "responses": ["Application process will remain open at 04th
   of July to 14th of august", "from 4th July to 14th
   august","you can apply from 4th july to 14th august"]
},
{"tag":"Application_form",
  "patterns":["how to avail admission form?","From where to
   have admission form?","where can I get an application
   form? "],
  "responses":["Application form for admission is available at
    the Somaiya website","from somaiya website","you can get
    forms from somaiya website","visit somaiya website for
    forms"]
},
{"tag":"Location",
 "patterns":["where is the location of the college?","College is
  located at?","How to reach college?"],
 "responses":["Somaiya college of engineering is Located
  near vidyavihar station at central railway.","College is
   located in mumbai near vidyavihar railway station."]
},
```

Fonte: (SINGH et al., 2018)

Tal arquivo consiste em três partes, ou seja, etiquetas, Padrões e Respostas. A etiqueta define sobre o que a consulta é. Padrões e respostas, como o nome sugere, são usados para treinar o modelo com frases e obter respostas correspondentes. Os padrões são carregados e passados pelo processo de modulação de consulta. Aqui a PLN ocorre onde várias funções são aplicadas em forma de *pipeline* (técnica de segmentação de tarefas que possibilita que outras tarefas iniciem a execução antes do término das outras) que inclui Frases -> Tokenização -> Lematização -> Identificação de Partes do discurso. Esses dados são armazenados em forma de saco de palavras e usados como entrada para o modelo de treinamento. Usamos a técnica de saco de palavras para extração de recursos. A rede de alimentação direta é criada usando *tensorflow* com 4 camadas (1 camada de entrada + 2 ocultas + 1 camada de saída). O modelo treinado é salvo e usado para prever respostas tendo como entrada a consulta do usuário.

Durante o treinamento, o modelo cria um conjunto de palavras que é uma coleção de todas

as palavras únicas. Agora, quando a consulta é passada, ela passa pelo processo de modulação de consulta e convertida em saco de palavras. Aqui, as palavras presentes são representadas por 1 enquanto as outras como 0. Considere o exemplo a seguir.

Consulta: como conseguir admissão na faculdade?

A seguir, é apresentada a representação de um conjunto de palavras para a consulta especificada:

[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

Agora, quando o usuário fizer a consulta, essa consulta será passada por meio do mesmo processo de modulação de consulta e convertido em conjunto de palavras. Isso será fornecido como entrada para o modelo e o modelo retornará todas as etiquetas com probabilidades maiores que erro limite definido. O modelo de resposta selecionará a resposta que será mais precisa e com maior classificação dada pelo sistema de *feedback* como a seguir:

Consulta: Olá
Predição: [('saudação', 0,98416048)]
Resposta: Olá, obrigado por visitar!

Aqui, o modelo previu uma saudação de intenção com probabilidade de 0,98.

Consulta: Como obter admissão na faculdade?
Predição: [('Processo de Admissão', 0,99881846)]
Resposta: para admissão, a pessoa deve seguir os passos....

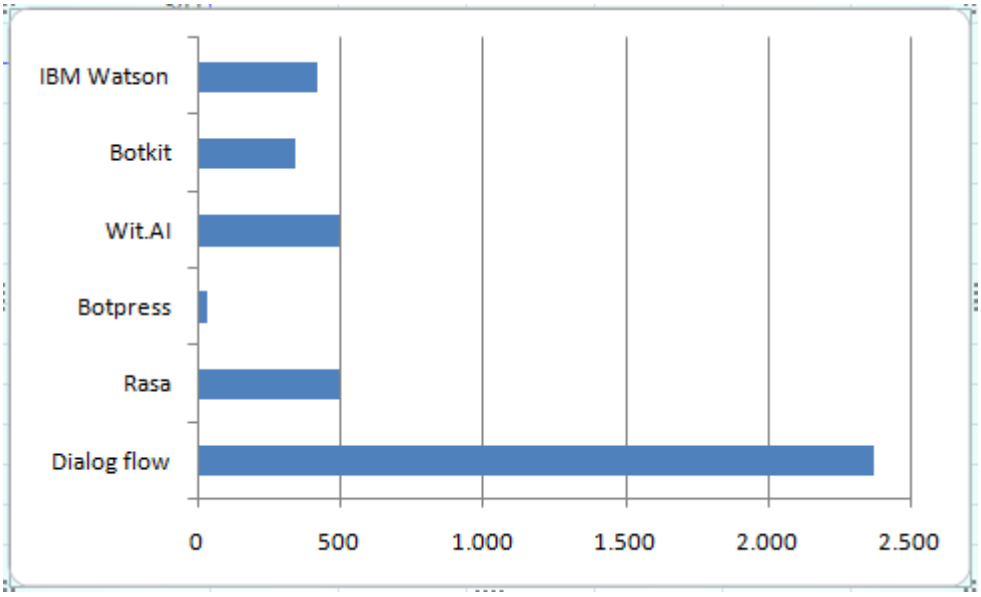
Da mesma forma, aqui, o modelo previu o processo de admissão com probabilidade de 0.9988

Por fim, (SINGH et al., 2018) concluem que o principal objetivo do estudo era fornecer um sistema para criar *chatbot* que pode ser usado por pequenas empresas como substituto de suporte ao cliente. O método demonstrado foi bem sucedido para domínios pequenos e fica mais preciso à medida que o tamanho do arquivo de intenção aumenta. (SINGH et al., 2018) constatam ainda que a precisão do *chatbot* é diretamente proporcional ao tamanho de arquivo de intenção usado para treinar o *chatbot*. Este método é claramente adequado em situações onde o domínio é estreito e o usuário interage com alguma relevância.

3.2 Revisão de Mercado

A revisão sistemática de mercado tem como objetivo encontrar e listar as principais tecnologias atuais para construção de *chatbots* inteligentes. Essa revisão de mercado foi realizado por Tiago Conceição Santos e está disponível na íntegra em: <https://github.com/quixote15/ludiico_researchs> para acesso público. Nessa revisão foram levantadas as principais *frameworks* atuais para desenvolvimento de *chatbots* baseados em aprendizado de máquina como mostra a figura 5

Figura 5 – Principais *frameworks*



Fonte: https://github.com/quixote15/ludiico_researchs/blob/master/TCC-TIAGO/Imagens/frameworks-stack.png

Todas as *frameworks* foram analisadas com maior aprofundamento no estudo e extraídas as principais características como mostra o Quadro 3.

Tabela 3 – Principais características identificadas nas *frameworks*

Identificação	Características
C1	*Open source*
C2	Integração com mensageiros externos
C3	Processa linguagem natural
C4	Usa aprendizado de maquina para inferir contexto
C5	Processa áudio
C6	Fornece interface gráfica
C7	Realiza análise estatística

Fonte: <https://github.com/quixote15/ludiico_researchs>

A partir do levantamento destas características foi construído um quadro associando estas a cada uma das *frameworks*. A tabela 4 exibe quais característica são apresentadas por cada *framework*.

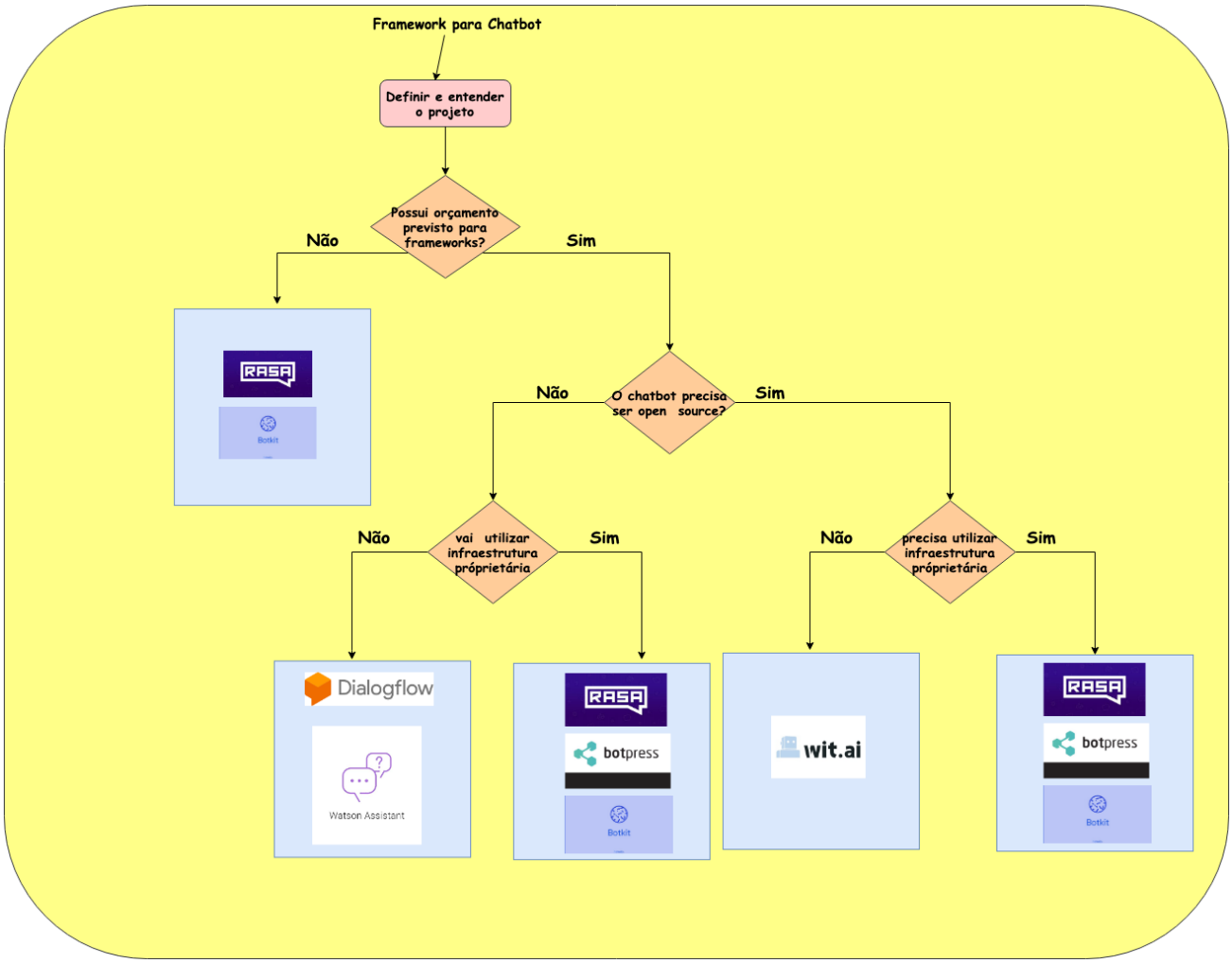
Tabela 4 – Associação das características com as *frameworks*

	C1	C2	C3	C4	C5	C6	C7
Dialog Flow			X	X	X		
Watson Assistant			X	X			
Rasa	X		X	X			
Wit.ai	X		X	X	X		
Botpress	X	X	X			X	X
Botkit	X	X			X	X	X

Fonte: <https://github.com/quixote15/ludiico_researchs>

por fim, baseado nas características das *frameworks* apresentadas acima foi desenvolvido um diagrama de decisão como mostra a figura 6 para auxiliar na tomada de decisão.

Figura 6 – Diagrama ilustrativo do processo de escolha das *frameworks* de desenvolvimento.



Fonte: <https://github.com/quixote15/ludiico_researchs>

3.3 Considerações Finais

A *framework Rasa* atende todas as característica discutidas neste estudo, isto é, *Open source*, processa linguagem natural e usa aprendizado de máquina para inferir contexto. Além de abranger as características discutidas na revisão sistemática dos estudos do estado da arte, ou seja, pode ser usada para implementar a arquitetura e o método discutidos nos estudos de (KARVE et al., 2018) e (SINGH et al., 2018), respectivamente. Portanto, este projeto de desenvolvimento do *chatbot* LIA utilizará a *framework Rasa*.

4
Desenvolvimento

Neste capítulo serão explanados os passos utilizados no desenvolvimento do *chatbot* LIA e suas respectivas tecnologias. Este capítulo está dividido da seguinte forma: seção 4.1 Descrição geral do sistema; 4.2 Requisitos do sistema; 4.3 Ferramentas e tecnologias utilizadas; 4.4 Representação da arquitetura; 4.5 Implementação; 4.6 Testes e 4.7 Cronograma de atividades.

4.1 Descrição geral do sistema

Esta seção apresenta uma visão geral da solução proposta e seus envolvidos, assim como o escopo levantado, tecnologias para seu desenvolvimento, propostas de estimativas de tempo e custo, e um planejamento da implementação.

4.1.1 Problema

O mercado imobiliário é um ramo que movimenta grandes valores e possui uma enorme volubilidade. O fluxo de caixa de uma construtora pode variar bastante, de acordo com o ritmo das vendas ou a necessidade de investir em novos empreendimentos. Um dos grandes empecilhos para um resultado ainda melhor do setor está no fato de que um cliente muitas vezes precisa se desfazer do seu imóvel ou bens para conseguir capital e, então conseguir negociar um imóvel novo.

Portanto, o objetivo desta solução está em facilitar as negociações de ambos os envolvidos, tanto das construtoras que está vendendo um imóvel novo, quanto do cliente que está tentando vender seu imóvel atual, para conseguir negociar um novo, uma vez que, a construtora pode assumir um papel de facilitadora da negociação e ajudar na efetivação da transação.

4.1.2 Principais *Stakeholders* e usuários

Nesta subseção é detalhado no documento as atribuições e responsabilidades no projeto.

Quadro 5: Principais *Stakeholders* e usuários

Papel	**Responsabilidades**	***Stakeholders***
Colaborador	Fornecer clareza da visão do projeto e discutir em reuniões sobre a importância e magnitude do projeto.	Breno Peixoto
Orientador	Orientar sobre temas específicas relacionados a inteligência artificial, aprendizado de máquina e processamento de linguagem natural	Hendrik Macedo
Coorientador	Orientar sobre temas relacionados a escrita e normas de documentos acadêmicos	Gilton Ferreira
Desenvolvedor	Desenvolver o sistema proposto com base nos métodos, técnicas e orientações explanadas neste trabalho, assim como realizar testes de funcionalidade	Diogo Lima
Usuários	Os usuários tem a função primordial de utilizar a aplicação desenvolvida e proporcionar os devidos *feedbacks*	Todos os membros do Laboratório para Universalização do Desenvolvimento, Inovação e Inteligência Computacional (ludii.co)

Fonte: este trabalho

4.1.3 Regras de Negócio

As regras de negócio tem a função de detalhar as funcionalidades particulares do sistema/software elencando pontos cruciais aos quais a aplicação deve satisfazer como os descritos a baixo:

- Qualquer usuário poderá utilizar a aplicação;
- Ao registrar novas informações o sistema informará ao usuário;
- Sistema ficará online sempre que possível;
- Sistema poderá enviar notificações aos usuários informando alguma atualização;
- Tempo de resposta de no máximo 5 segundos;
- Sistema deve se recuperar de falhas.

4.2 Requisitos do sistema

Para descrever os requisitos deste projeto foi utilizada a técnica de Histórias de Usuário (User Stories) com o intuito de simplificar tal documentação usando algo funcional e descritivo.

4.2.1 Requisitos Funcionais

Quadro 6: Requisitos Funcionais

Identificação	**Descrição**	**Classificação (Importante, Essencial ou Desejável)**
RF01	Como um **usuário** interessado em comprar um imóvel no bairro X, **gostaria** de visualizar imóveis naquele bairro a **fim** de escolher o que melhor se encaixa a meu perfil	Essencial
RF02	Como **desenvolvedor** de *chatbots* **devo** classificar as intenções do usuário a fim de compreendê-lo	Essencial
RF03	Sendo **desenvolvedor** de *chatbots* **devo** implementar/utilizar um módulo de compreensão no *chatbot* **com o objetivo** de que ele entenda a linguagem natural do humano	Importante
RF04	Sendo **desenvolvedor** de *chatbots* **devo** utilizar técnicas de aprendizado de máquina no desenvolvimento do *chatbot* com o objetivo de que ele aprenda a cada conversa com usuário	Desejável
RF05	Como **desenvolvedor** de *chatbots* **devo** definir alguns principais fluxos de conversas do domínio em questão	Importante
RF06	Como **desenvolvedor** de *chatbots* **devo** implementar o *chatbot* para que ele guarde das respostas dadas pelo usuário informações específicas, por meio de *Slots* do Rasa, **a fim** de buscar imóveis armazenados no banco de dados de acordo com essas informações	Importante
RF07	Sendo **desenvolvedor** de *chatbots* **devo** implementar o *chatbot* para que nunca desvie o fluxo de conversa do domínio em questão mesmo quando o usuário lhe pergunte algo fora do domínio	Desejável
RF08	Como **desenvolver** de *chatbot* **devo** usar os *FormActions* do *Rasa* para forçar que o usuário forneça todas as informações que o *chatbot* precisa para fazer a busca pelos imóveis	Importante
RF09	Como **desenvolver** de *chatbot* **devo** usar o *TensorFlow* no *Rasa* para permitir que o sistema aprenda por meio de aprendizado de máquina	Essencial

Fonte: este trabalho

4.2.2 Requisitos Não-funcionais

Quadro 7: Requisitos Não-funcionais

Identificação	**Descrição**	**Classificação (Importante, Essencial ou Desejável)**
RNF01	Como **desenvolver** de *chatbot* **preciso** garantir a confiabilidade dos dados fornecidos pelo usuário no momento da conversa com o *chatbot*	Essencial
RNF02	Interface intuitiva: o sistema deverá possuir interface simples, e que necessite de uma baixa curva de aprendizagem do usuário, e deve possuir uma quantidade de etapas para a realização de tarefas.	Desejável
RNF03	O interface adaptativa: o sistema deve ser adaptável a diversas resoluções de telas.	**D**esejável
RNF04	Não interrupção: o sistema precisa está sempre online	**I**mportante

Fonte: este trabalho

4.3 Ferramentas e tecnologias utilizadas

Esta seção traz uma descrição rápida de todas as tecnologias usadas neste projeto como a seguir:

[Ubuntu 18.04.03 LTS:] o *Ubuntu* é um sistema operacional *open source* amplamente utilizado por cientistas da computação. Para mais informações acesse:<https://ubuntu.com/>;

Framework Rasa: a *framework Rasa* ou *Rasa Stack* possui um conjunto de ferramentas de aprendizado de máquina para que desenvolvedores possam criar *chatbots* contextuais, diferente de daqueles baseados em regras pré-definidas. Essa *framework* é composta de dois módulos que são independentes e podem ser usados separadamente. O módulo principal (Rasa Core) e módulo de processamento de linguagem natural (chamado de *Rasa natural language understanding ou Rasa NLU).* Para mais informações acesse: <https://rasa.com/>;

***Python* 3.7:** a linguagem *Python* está entre as linguagens de programação mais populares da atualidade. Quando o contexto é sobre inteligência artificial esta torna-se a principal linguagem de programação da atualidade, tendo em vista que a maior parte das aplicações/algoritmos são construídos nela. Para mais informações acesse: <https://docs.python.org/3/>;

Docker compose: o *Docker Compose* é uma ferramenta para definir e executar aplicativos *Docker* de vários contêineres. Com o *Compose*, você usa um arquivo *YAML* para configurar os

serviços do seu aplicativo. Em seguida, com um único comando, você cria e inicia todos os serviços da sua configuração. Para saber mais consulte: <https://docs.docker.com/compose/>;

***GitHub*:** o *GitHub* é uma plataforma de hospedagem de código-fonte com controle de versão usando o *Git*. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou *Open Source* de qualquer lugar do mundo. Para mais informações acesse: <https://github.com/>;

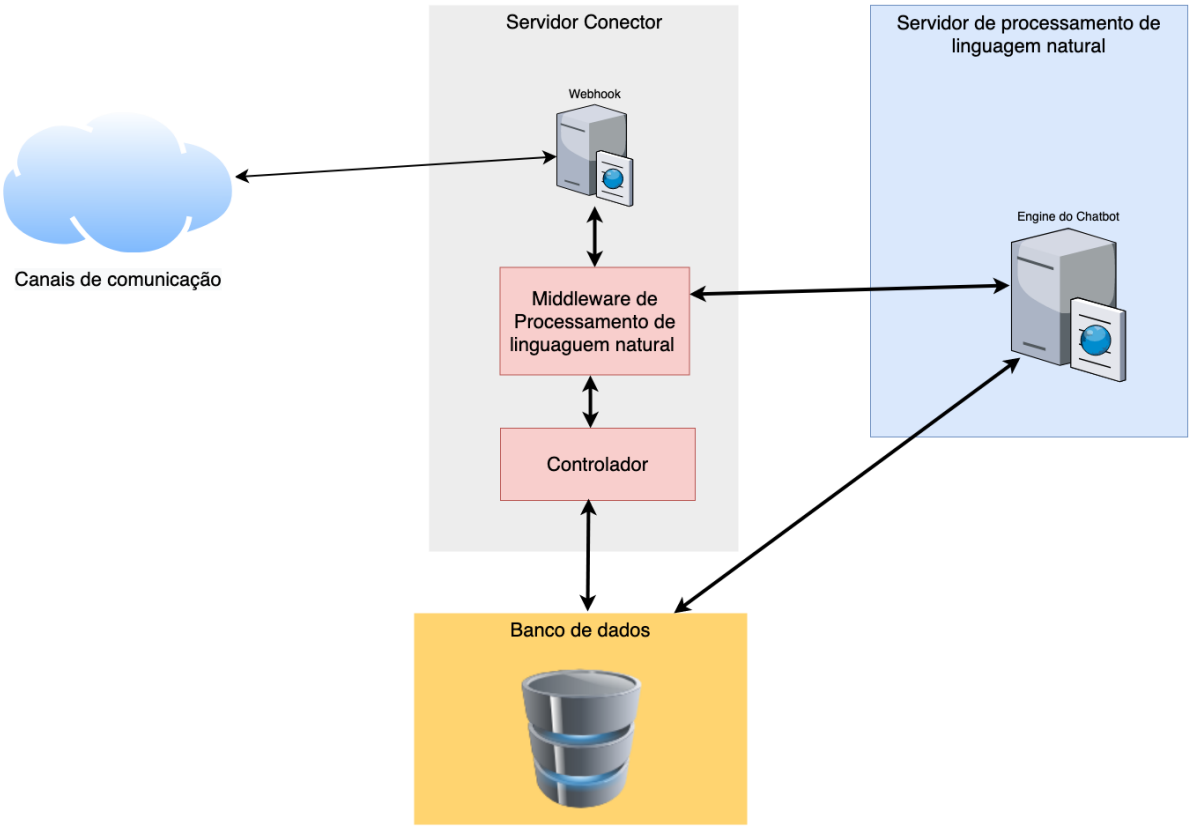
***DigitalOcean*:** a *DigitalOcean*, *Inc*. é um provedor de infraestrutura em nuvem americano com sede na cidade de *Nova York* e com data centers em todo o mundo. A *DigitalOcean* fornece aos desenvolvedores serviços em nuvem que ajudam a implantar e dimensionar aplicativos executados simultaneamente em vários computadores. Para mais informações acesse: <https://www.digitalocean.com/>

MongoDB : é um software de banco de dados orientado a documentos de código aberto e multiplataforma escrito na linguagem C++. Classificado como um programa de banco de dados *NoSQL*, o *MongoDB* usa documentos semelhantes a *JSON* com esquemas. Para mais informações acesse: <https://www.mongodb.com/>.

4.4 Representação da Arquitetura

A arquitetura geral proposta é ilustrada na figura 7. Ela está dividida em 4 partes principais: canais de comunicação, servidor conector, banco de dados (BD) e servidor de processamento de linguagem natural (PLN).

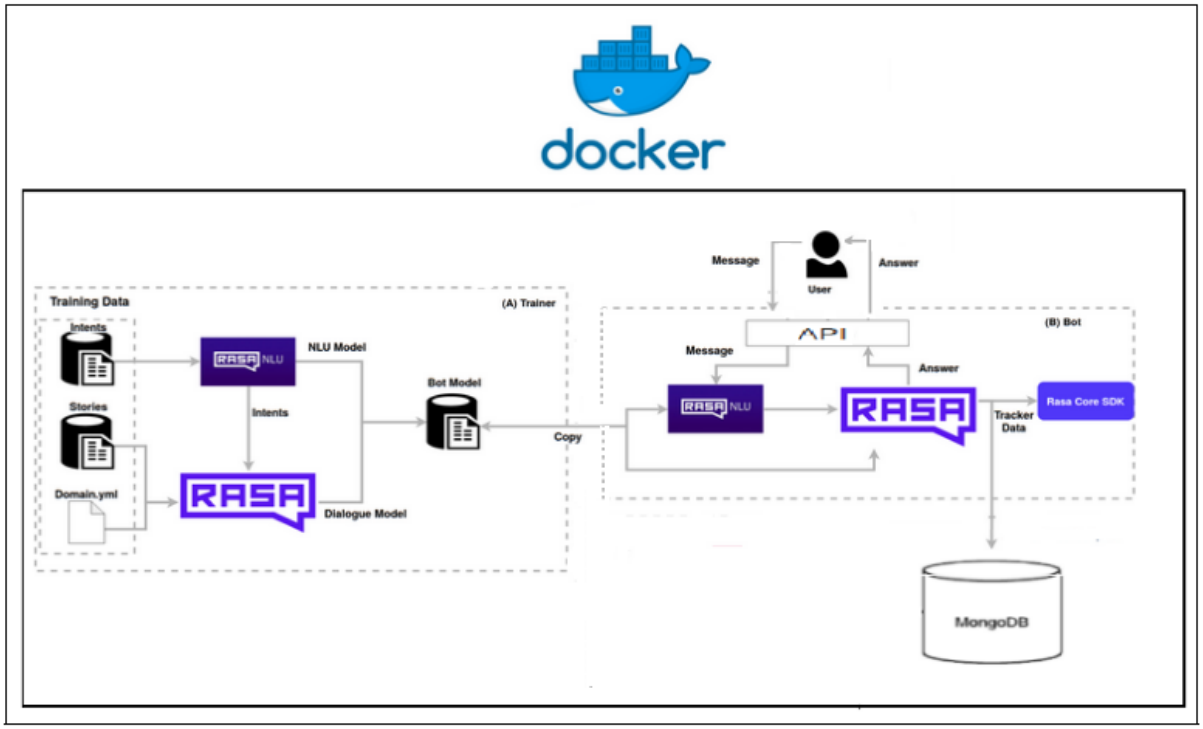
Figura 7 – Arquitetura geral do sistema



Fonte: https://github.com/quixote15/ludiico_researchs/blob/master/TCC-TIAGO/Imagens/Botkit-architecture.png

Este projeto está restrito apenas ao desenvolvimento dos módulos de servidor de PLN e banco de dados. Portanto, aqui não abordaremos o desenvolvimento do módulo canais de comunicação e servidor conector. O detalhamento do servidor de PLN e Banco de dados do *chatbot* LIA é apresentado na figura 8.

Figura 8 – Detalhamento do servidor de PLN e BD do LIA



Fonte: este trabalho

O servidor de PLN é constituído basicamente pela *Rasa Stack,* isto é, *Rasa Core* e *Rasa NLU*. O Rasa Core (núcleo) é uma ferramenta que usa aprendizado de máquina para inferir possíveis ações a serem executadas pelo *chatbot*. O aprendizado de máquina é feito com o constante *feedback* do usuário, e a partir disso, são calculadas as probabilidades de execução de cada ação. Essa abordagem é conhecida como aprendizado iterativo. o Core analisa as mensagens por meio do gerência de fluxo de conversas de modo a deixar o diálogo mais fluido e sem perder o contexto da conversa. Na próxima seção o Rasa Core será mais detalhado sobre aspectos de implementação.

Já o *Rasa NLU* é uma ferramenta para entendimento de linguagem natural *open source* capaz de classificar as intenções das conversas com usuários e extrair entidades em *chatbots*. Sendo assim, é possível extrair dados estruturados de uma conversa em linguagem natural. o módulo *NLU* pode ser usado separadamente em qualquer projeto de *chatbot* inteligente, visto que, ele independe de qualquer infraestrutura proprietária ou requisições a *api's* externas, pois todos os dados de treinamento podem ser armazenados localmente. Na próxima seção o *Rasa NLU* também será mais detalhado sobre aspectos de implementação.

Logo, percebe-se que esses dois componentes *(Rasa core* e *Rasa NLU*) juntos formam o 'cérebro' do *chatbot* LIA, uma vez que, são eles quem mantém todo o diálogo de uma conversa sem perder o contexto da mesma e, assim proporcionar experiências mais amigáveis com os

usuários do LIA.

Em sequência vem o servidor de Banco de dados que é utilizado para persistir de dados e realizar consultas no mesmo. O BD aqui utilizado foi o *MongoBD* por ser uma alternativa *open source,* orientado a documentos dentre outras características.

Por fim, também utilizamos a ferramenta *Docker* que permite isolamento do ambiente de trabalho por meio de *conteiners* evitando assim conflitos de versões de software. Com a tecnologia *Docker compose* pode criar uma ou mais imagens executáveis de software e posteriormente só basta executá-las em um ambiente com *Docker* instalado e pronto! Temos tudo funcionando em poucos minutos.

4.5 Implementação

Esta seção descreve os principais passos para construção e execução do *chatbot* LIA usando a *framework Rasa*. Neste trabalho foi utilizado como referência a versão 1.1.8 do *Rasa* que é a versão mais recente no período de desenvolvimento deste trabalho. Para obter mais informações acesse a documentação oficial o Rasa em: <https://rasa.com/docs/>

4.5.1 Instalação e execução

Para instalar e executar o *chatbot* LIA siga as seguintes instruções:

1. Baixe e instale o *Docker-Compose* em sua máquina.

Código 1 – Instalação do *docker* e *docker-compose*

```
1 docker -v && docker-compose -v
```

Para mais informações acesse: https://docs.docker.com/compose/install/

2. Clone o projeto *Lia-bot* executando:

Código 2 – Clonando o *git* do projeto

```
1 git clone https://github.com/diogolima25/LIA-bot.git
```

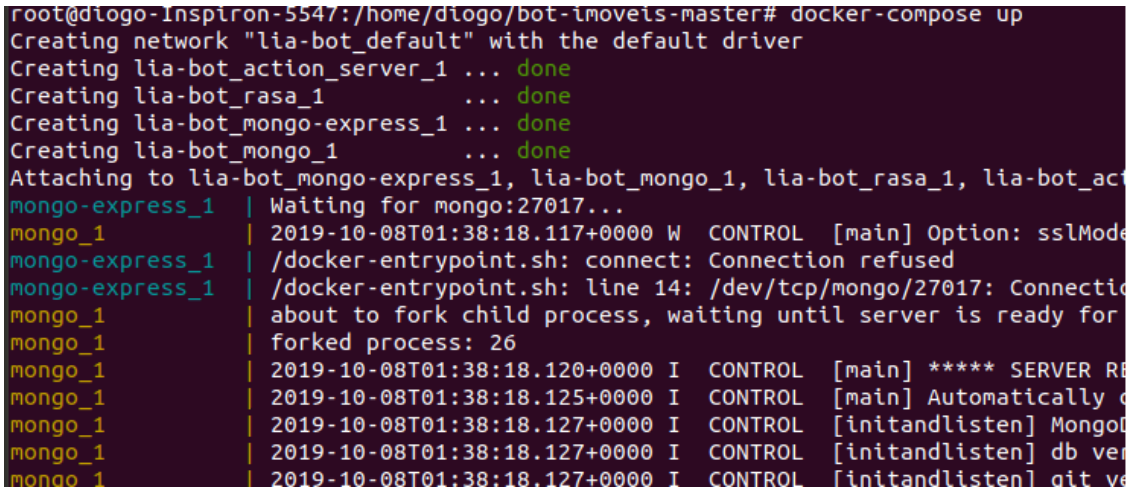
3. Vá até o diretório da pasta clonada abra um novo terminal como *root* e execute o seguinte comando:

Código 3 – Execução do LIA

```
1 docker-compose up
```

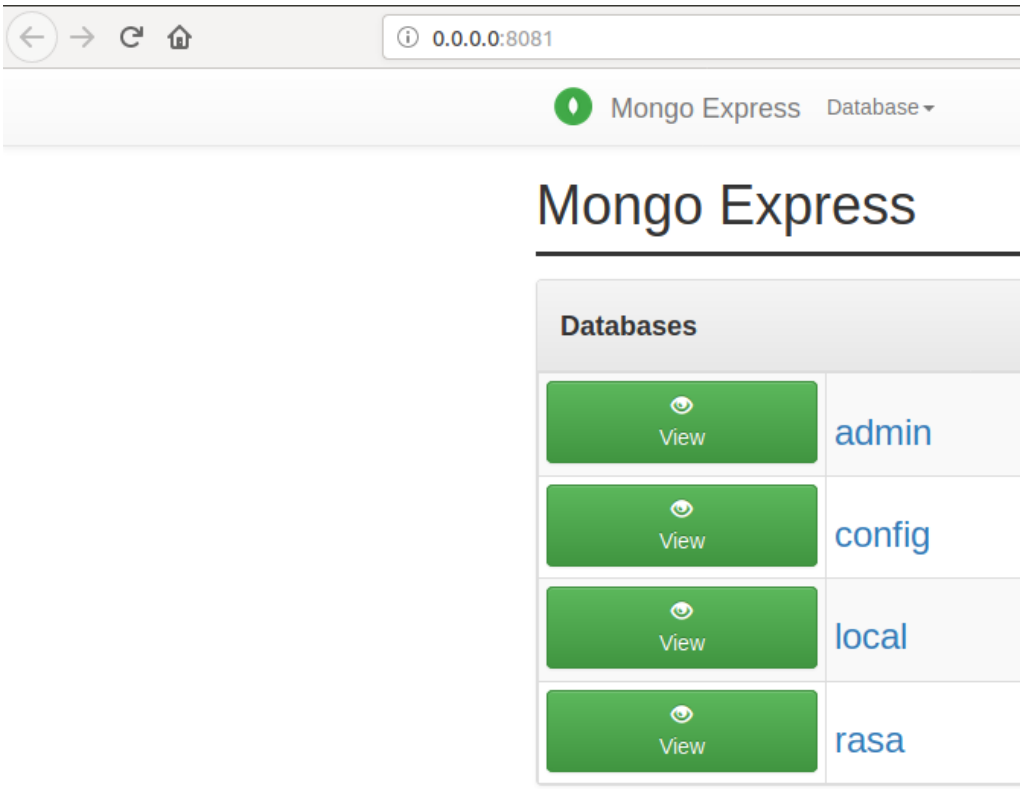
A execução desse comando ira baixar automaticamente todas as dependências, criar as imagens e levantar o servidor *MongoDB* localmente como mostra as figuras 9 e 10, respectivamente.

Figura 9 – Imagens docker criadas



Fonte: este trabalho

Figura 10 – Servidor mongoDB



Fonte: este trabalho

4.5.2 Rasa NLU

Nesta subseção será analisado cada arquivo considerado importante, do *Rasa NLU*, para desenvolver o *chatbot LIA*. Serão tomados como exemplo os arquivos reais do projeto fruto desde trabalho.

4.5.2.1 Arquivo de *intents*

O primeiro arquivo a considerar no projeto de um *chatbot* usando Rasa é o arquivo de *intents*. Nele contém os exemplos de treinamento com as possíveis intenções do usuário que diz como o *Rasa NLU* deve entender as mensagens do usuário.

No arquivo de *intents* as linhas que começam com '##' seguido do nome '*intent*' definem os nomes das intenções, que são grupos de mensagens com o mesmo significado. O trabalho do Rasa será prever a intenção correta, usando um modelo probabilístico, quando os usuários enviarem mensagens novas. O Código 4 mostra um exemplo de *intent* do arquivo citado:

Código 4 – Exemplo da *intent* cumprimentar.

```
## intent:cumprimentar
- oi, tudo bem
- bom dia
- boa tarde
- boa noite
- oi
- ola
- oie
- oiee
- opa tudo bem
- oi, como vai voce
- ola boa tarde
.
. // outras intents
.
```

Com esse exemplo sempre que o usuário entrar com uma dessas palavras ou frases o chatbot identificará que a intenção do usuário é cumprimentar e então responderá de acordo.

4.5.2.2 Arquivo *config.yml*

O arquivo *config.yml* é o arquivo de configuração que define os componentes que o *Rasa NLU* usará para treinar o modelo. O código 6 traz as configurações usadas neste projeto.

Código 5 – Arquivo de configuração do NLU.

```
language: pt

pipeline:
  - name: WhitespaceTokenizer
  - name: CRFEntityExtractor
  - name: EntitySynonymMapper
  - name: CountVectorsFeaturizer
    token_pattern: (?u)\b\w+\b
  - name: EmbeddingIntentClassifier
  - name: DucklingHTTPExtractor
    url: http://localhost:8000
    dimensions:
      - number
```

A chave *language* traz a configuração de em qual idioma o modelo será treinado.

No *Rasa NLU,* as mensagens recebidas são processadas por uma sequência de componentes. Esses componentes são executados um após o outro em um chamado pipeline de processamento. Existem componentes para extração de entidade, para classificação de intenção, seleção de resposta, pré-processamento e outros.

O *pipeline* especifica como o modelo *NLU* deve ser construído. A escolha de um *pipeline NLU* permite personalizar o modelo e ajustá-lo no conjunto de dados.

Neste projeto usamos o pipeline *supervised_embeddings*. A vantagem do pipeline *supervised_embeddings* é que seus vetores de palavras serão personalizados para o seu domínio. Por exemplo, em inglês geral, a palavra 'saldo' está intimamente relacionada à 'simetria', mas muito diferente da palavra 'dinheiro'. Em um domínio bancário, 'saldo' e 'caixa' estão intimamente relacionados e você deseja que seu modelo capture isso. Esse *pipeline* não usa um modelo específico de idioma; portanto, ele funcionará com qualquer idioma que você possa *tokenizar* (no espaço em branco ou usando um *tokenizer* personalizado). Abaixo detalhamos a funcionalidade de cada componente do pipeline *supervised_embeddings* utilizado neste trabalho:

- ***WhitespaceTokenizer*** cria um *token* para cada sequência de caracteres separados por espaços em branco;
- ***CRFEntityExtractor*** este componente implementa campos aleatórios condicionais para fazer o reconhecimento de entidade nomeada. Os *CRFs* podem ser vistos como uma cadeia de *Markov* não direcionada, onde os intervalos de tempo são palavras e os estados são classes de entidade. Os recursos das palavras (capitalização, marcação de PDV etc.) fornecem probabilidades para determinadas classes de entidade, assim como as transições entre as tags de entidade vizinhas: o conjunto mais provável de *tags* é calculado e retornado. Se forem utilizados recursos de *POS* (pos ou pos2), o *spaCy* deverá ser instalado.;
- ***EntitySynonymMapper*** mapeia os valores da entidade sinônima para o mesmo valor;
- ***CountVectorsFeaturizer*** cria uma representação em conjunto de palavras dos recursos de mensagem e etiqueta do usuário usando o *CountVectorizer* do *sklearn*. Todos os *tokens* que consistem apenas de dígitos (por exemplo, 123 e 99, mas não a123d) serão atribuídos ao mesmo recurso;
- ***EmbeddingIntentClassifier*** incorpora entradas do usuário e rótulos de intenção no mesmo espaço. *Embeddings* supervisionados são treinados maximizando a semelhança entre eles. Este algoritmo é baseado no *StarSpace*. No entanto, nesta implementação, a função de perda é um pouco diferente e camadas ocultas adicionais são adicionadas juntamente com o *dropout*. Esse algoritmo também fornece classificações de similaridade dos rótulos que não 'venceram'0;
- ***DucklingHTTPExtractor*** permite extrair entidades comuns, como datas, quantias em dinheiro, distâncias e outros em vários idiomas.

4.5.3 Rasa core

Nesta subseção será analisado cada arquivo considerado importante, do *Rasa Core,* para desenvolver o *chatbot* LIA. Serão tomados como exemplo os arquivos reais do projeto fruto desde trabalho.

4.5.3.1 Arquivo *config.yml*

No arquivo *config.yml* também estão as configuração dos componentes do *Rasa Core* que o modelo de gerenciamento de diálogo usará.

Código 6 – Arquivo de configuração.

```
policies:
  - name: KerasPolicy
    priority: 5
    epochs: 20
    batch_size: 10
    featurizer:
      - name: FullDialogueTrackerFeaturizer
        state_featurizer:
          - name: LabelTokenizerSingleStateFeaturizer
  - name: FallbackPolicy
    nlu_threshold: 0.6
    core_threshold: 0.6
    priority: 1
  - name: MemoizationPolicy
    priority: 2
    max_history: 2
  - name: FormPolicy
  - name: MappingPolicy
```

A chave *policies* define as políticas que o módulo *Core* do *Rasa* usará. A seguir detalhamos todas essas políticas que foram utilizadas no desenvolvimento do *chabot* LIA:

- ***KerasPolicy*** usa uma rede neural implementada no *Keras* para selecionar a próxima ação. A arquitetura padrão é baseada em um *LSTM*;
- ***FullDialogueTrackerFeaturizer*** sempre analisa o histórico completo da conversa antes de decidir qual decisão irá tomar. Como exemplo, digamos que você tenha uma intenção *out_of_scope* que descreva mensagens de usuário fora do tópico. Se o seu *chatbot* vê essa intenção várias vezes seguidas, você pode dizer ao usuário com o que pode ajudá-lo? Portanto, sua história pode ficar assim:

```
 * out_of_scope
      - utter_default
* out_of_scope
     - utter_default
* out_of_scope
     - utter_help_message
```

Logo, para que o Rasa Core aprenda esse padrão, o *max_history* deve ser pelo menos 3. No entanto, aumentar sua *max_history* acarretará em um modelo maior e o treinamento levará mais tempo;

- ***FallbackPolicy*** invoca uma ação de *fallback* se pelo menos um dos seguintes ocorrer: 1. O reconhecimento de intenção possui uma confiança abaixo de *nlu_threshold*. 2. A intenção com classificação mais alta difere em confiança da segunda intenção com a classificação mais alta em menos de *ambiguity_threshold.* 3. Nenhuma das políticas de diálogo prevê uma ação com confiança superior ao *core_threshold;*

- ***MemoizationPolicy*** O *MemoizationPolicy* apenas memoriza as conversas nos seus dados de treinamento. Ele prevê a próxima ação com confiança 1.0, se essa conversa exata existir nos dados de treinamento, caso contrário, prevê Nenhum com confiança 0.0;

- ***FormPolicy*** exemplo a política *FormPolicy* está definida porque neste projeto faz-se uso de formulários implementados pelo rasa;

- ***MappingPolicy*** é usado para mapear intenções diretamente para ações. Uma intenção só pode ser mapeada para no máximo uma ação. O *bot* executará a ação mapeada assim que receber uma mensagem da intenção de acionamento. Depois, ele ouvirá a próxima mensagem. Com a próxima mensagem do usuário, a previsão normal será retomada. O *MappingPolicy* também é responsável por executar as ações padrão *action_back* e *action_restart* em resposta a / *back* e / *restart*;

4.5.3.2 Arquivo *stories.md*

Agora será tratado o arquivo de *stories*, é através deste que será possível ensinar o *chatbot* a responder às mensagens dos usuários. A comunidade Rasa chama isso de gerenciamento de diálogos e é tratado pelo módulo Core. O Código 7 apresenta uma das *stories* do arquivo *stories* deste projeto.

Código 7 – Arquivo de *stories*.

```
## historia_1
* cumprimentar
    - utter_cumprimentar
* comprar{"tipo_imovel": "casa"}
    - utter_comprar
    - usuario_form
    - form{"name": "usuario_form"}
    - form{"name": null}
    - utter_continuar_conversa
    - crud_action
    - utter_resultado_busca_imoveis
* gostou_do_imovel
    - utter_gostou_do_imovel
* confirmar
    - utter_confirmar
* outras_duvidas
    - utter_outras_duvidas
    - utter_despedir
* despedir
    - utter_despedir
```

Os modelos básicos aprendem com dados reais de conversação na forma de "histórias" de treinamento. Uma história é um fluxo conversa real entre um usuário e um assistente. No Código 7 as linhas com *intents* refletem ao que o assistente está esperando como entrada do usuário e qual ação que ele deve responder caso aconteça o que foi esperado por ele. Nesse exemplo, para que se inicie a conversa o assistente já espera que o usuário o cumprimente por esse motivo a *storie* se inicia com a *intent cumprimentar*, desse modo quando o usuário diz 'olá' ou 'oi' o assistente entende que sua intenção é cumprimentar e então ele responde de volta com um 'olá' ou 'Em que posso ajudar?' vai depender de como foi programado para responder a uma intenção de cumprimento. Após entender como funciona o arquivo *domain.yml* essa *storie* ficará mais clara. Quanto mais fluxos de conversas como esse mais o *Rasa Core* vai aprender a identificar as *intents* corretas e consequentemente executar as ações que melhore cada vez mais seus diálogos com os usuários.

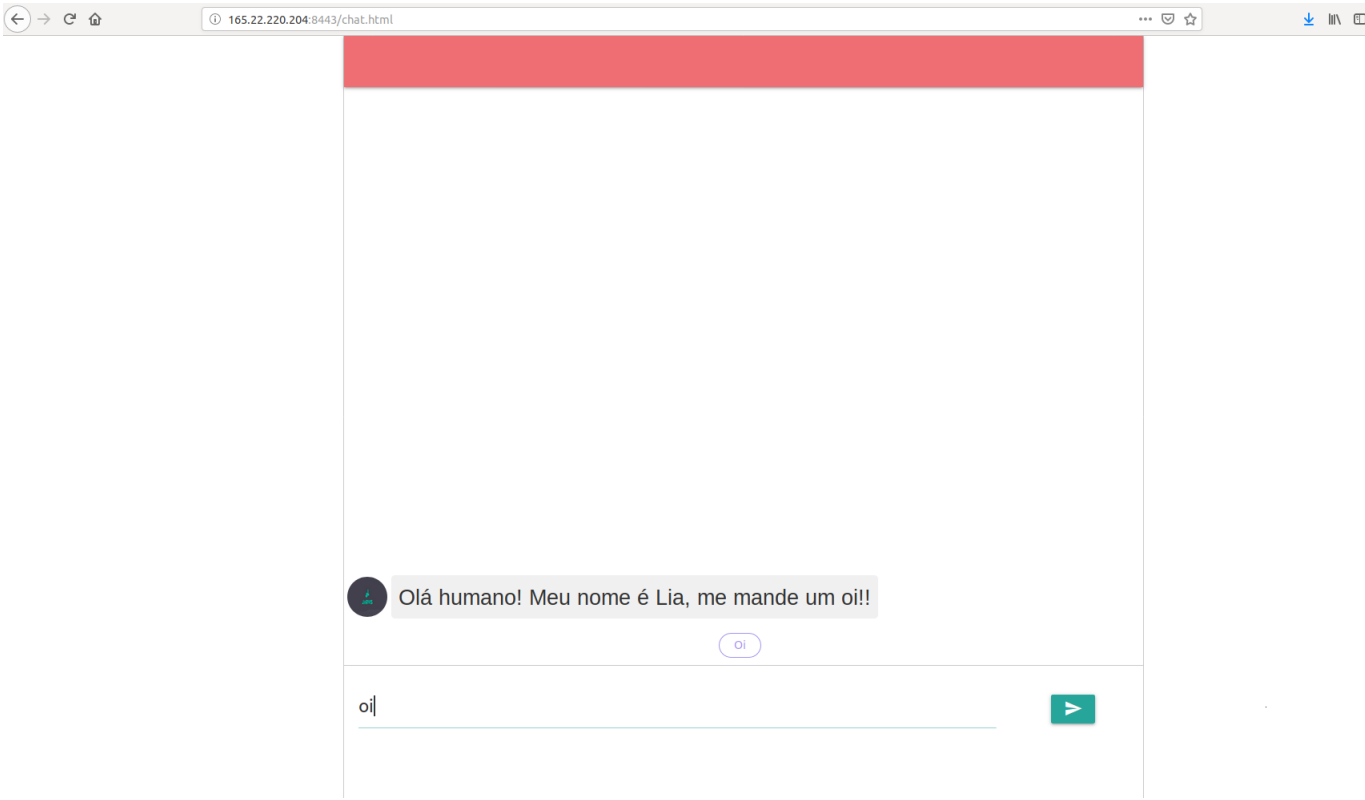
4.6 Testes de conversação

As figuras 11, 12, 13, 14, 15, 16, 17 e 18 mostram uma conversa com o *chatbot* Lia desenvolvido neste trabalho. Notamos que durante todo o diálogo de conversa ele consegue identificar com quem ele está conversando proporcionando, assim, experiências mais amigáveis que os *chatbots* que usam apenas regras pré-definidas. Também é possível notar que na transição

de uma conversa para outra ele sempre se mantém no contexto, isto é, no domínio de imóveis e mesmo que o usuário faça perguntas fora do contexto o Lia não permite perder o contexto da conversa.

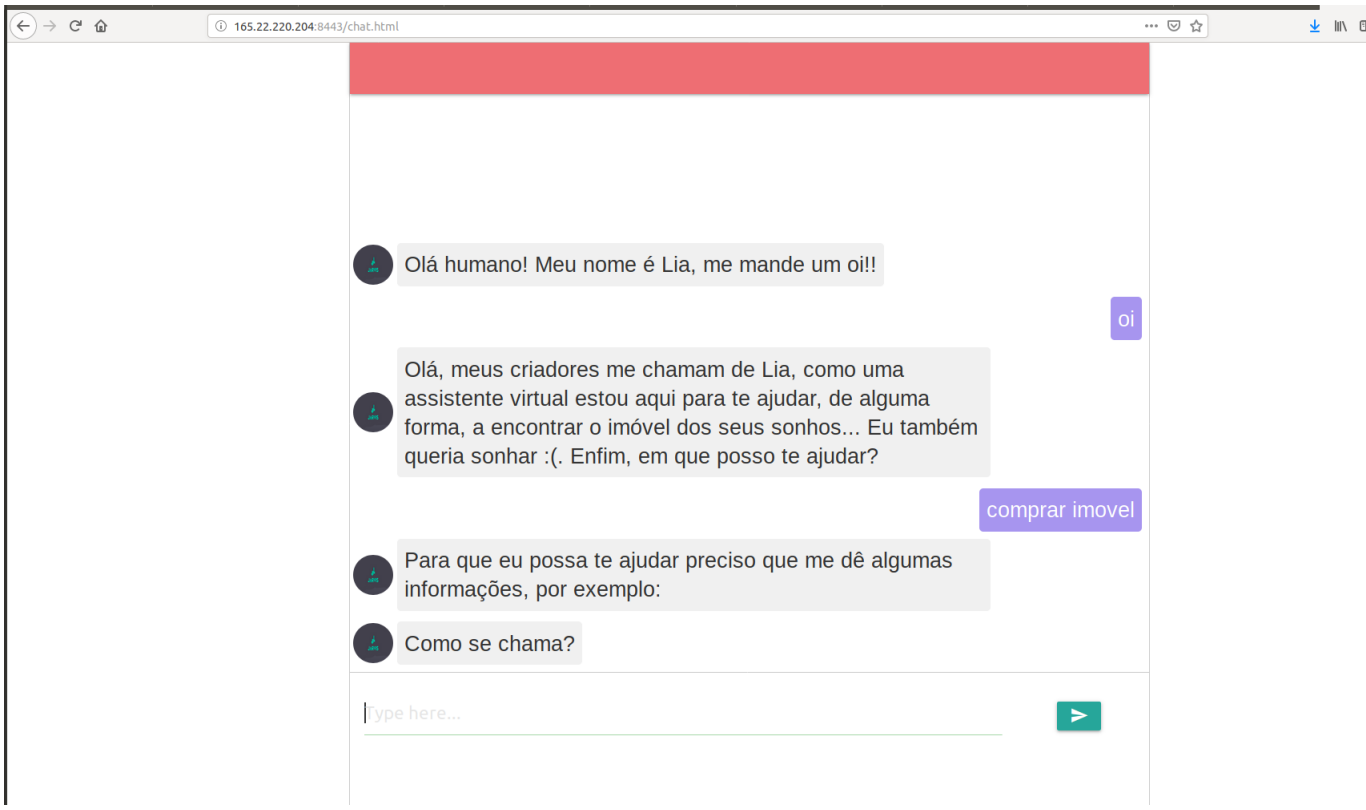
Por fim, notamos também na figura18 que o usuário termina um diálogo e já inicia outro com uma simples 'saudação'. Neste momento o Lia já conseguirá identificar com quem estava conversando e não irá pedir mais suas credenciais.

Figura 11 – Conversando com o *chatbot* Lia.



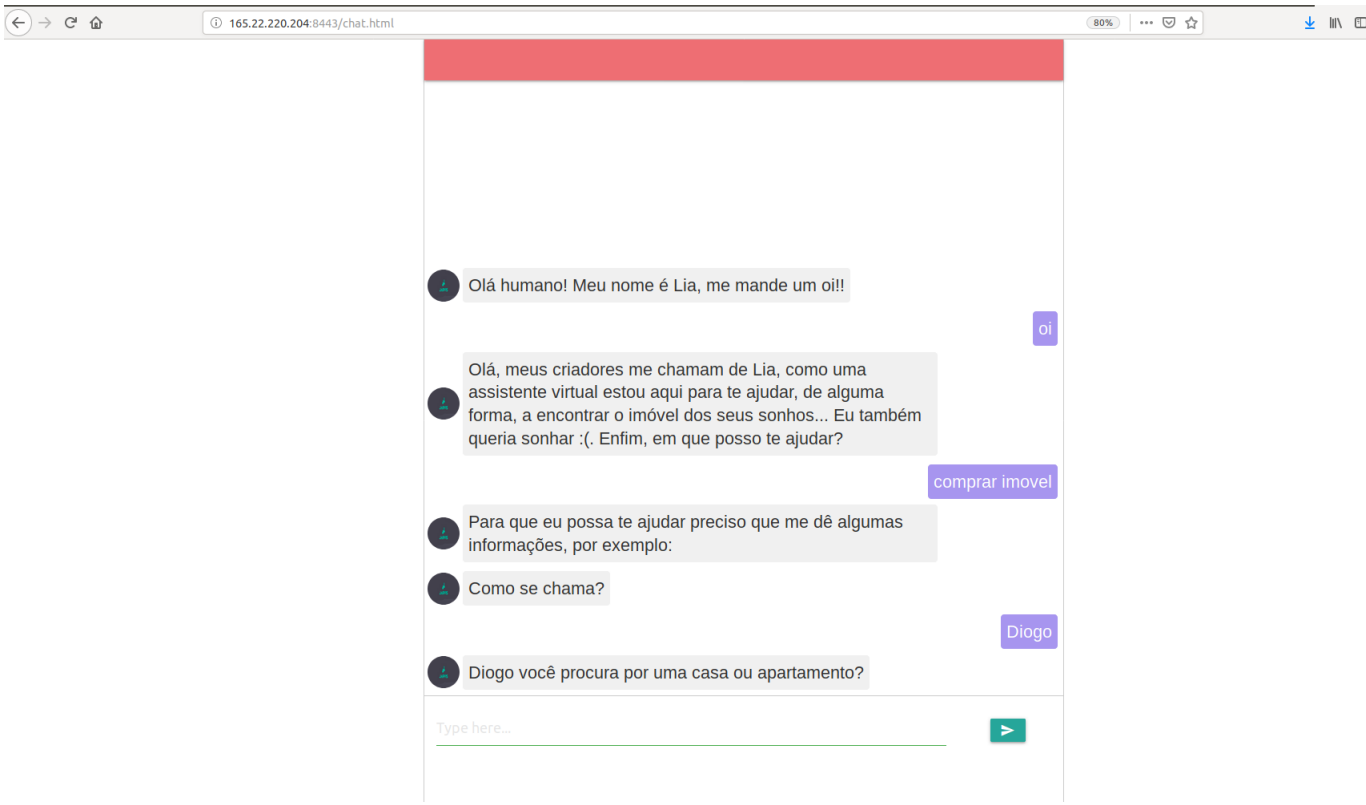
Fonte: esta pesquisa

Figura 12 – Conversando com o *chatbot* Lia..



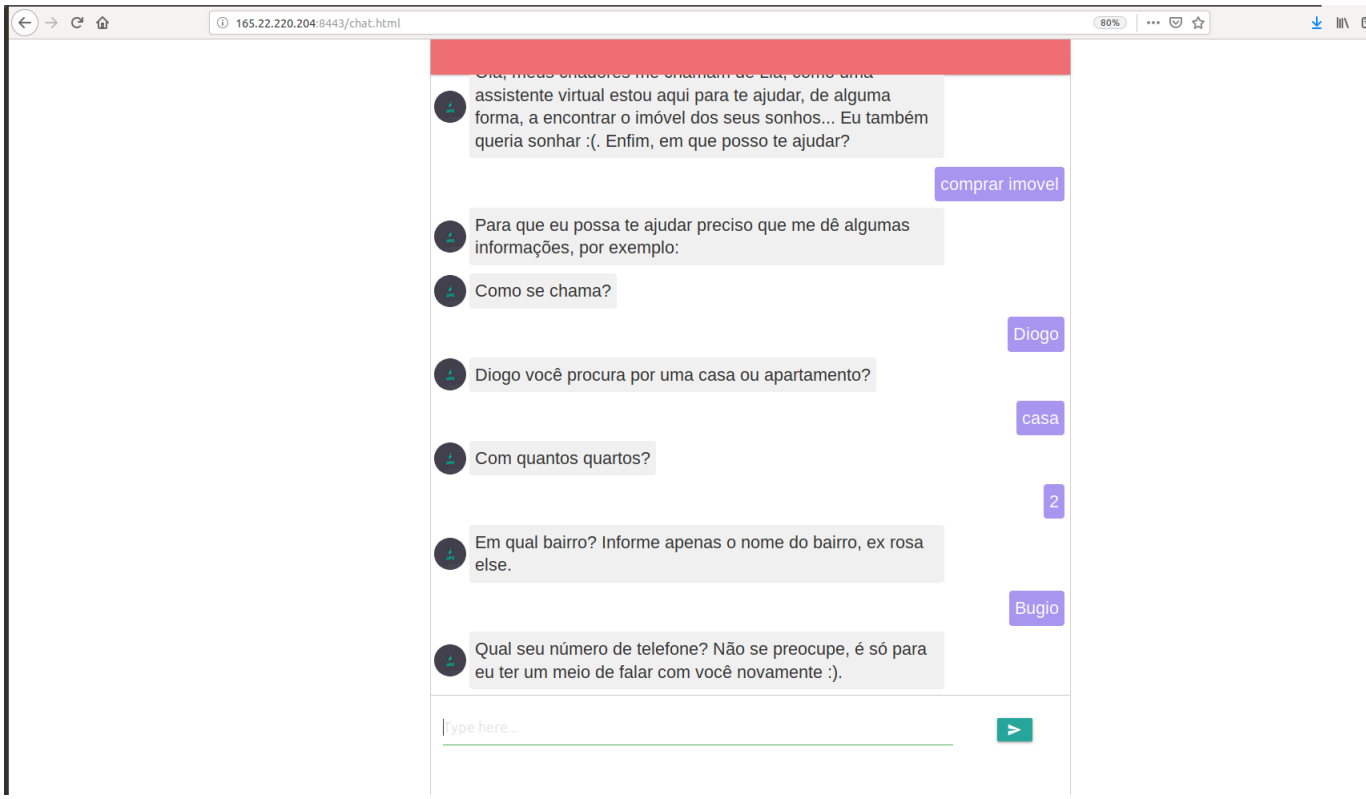
Fonte: esta pesquisa

Figura 13 – Conversando com o *chatbot* Lia..



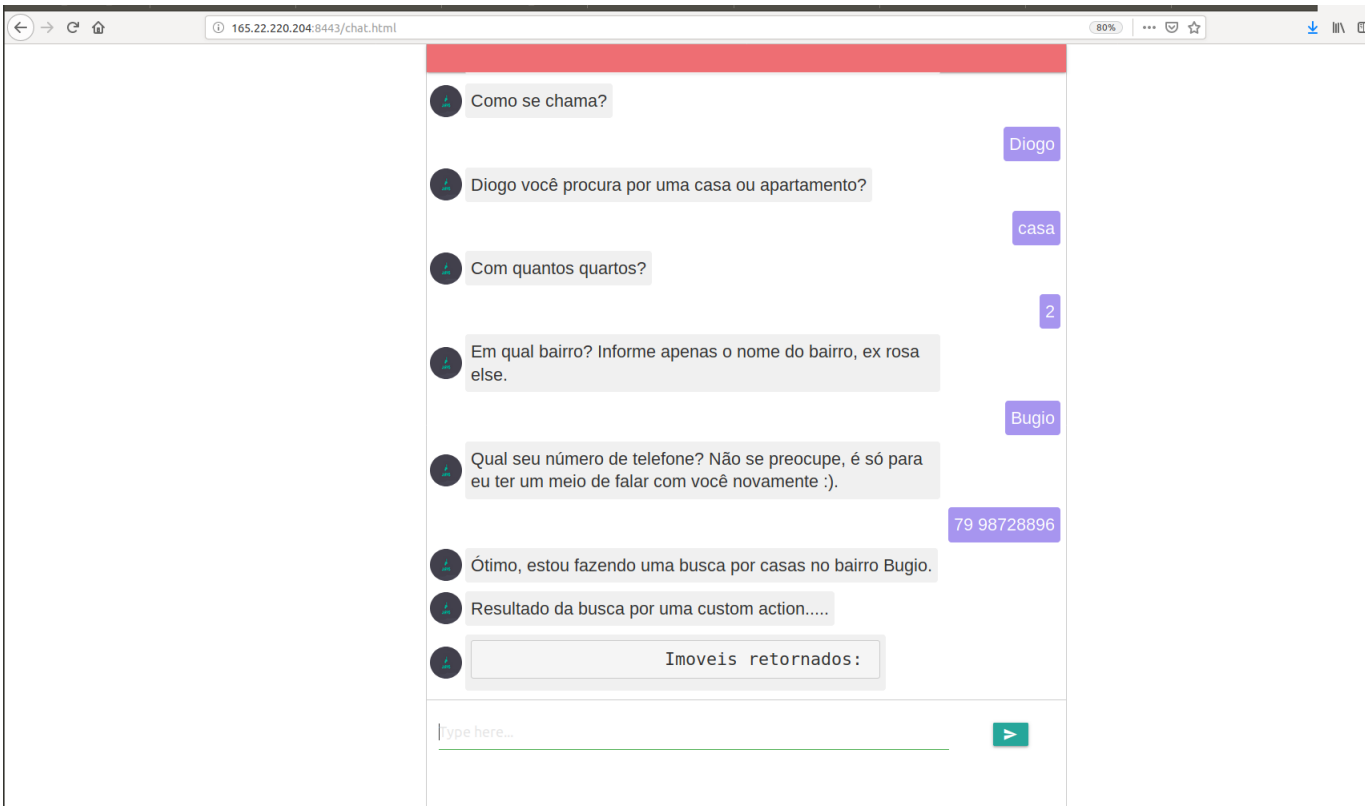
Fonte: esta pesquisa

Figura 14 – Conversando com o *chatbot* Lia..



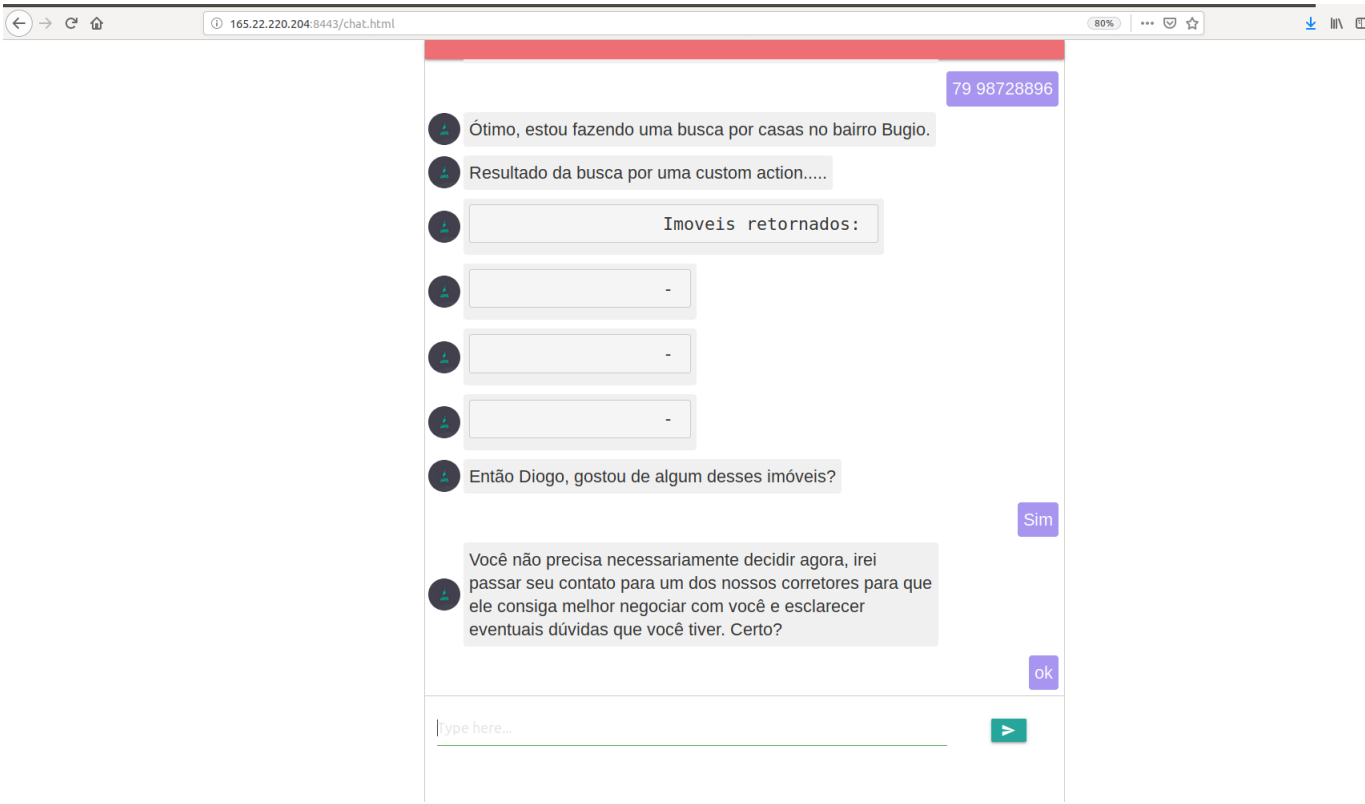
Fonte: esta pesquisa

Figura 15 – Conversando com o *chatbot* Lia..



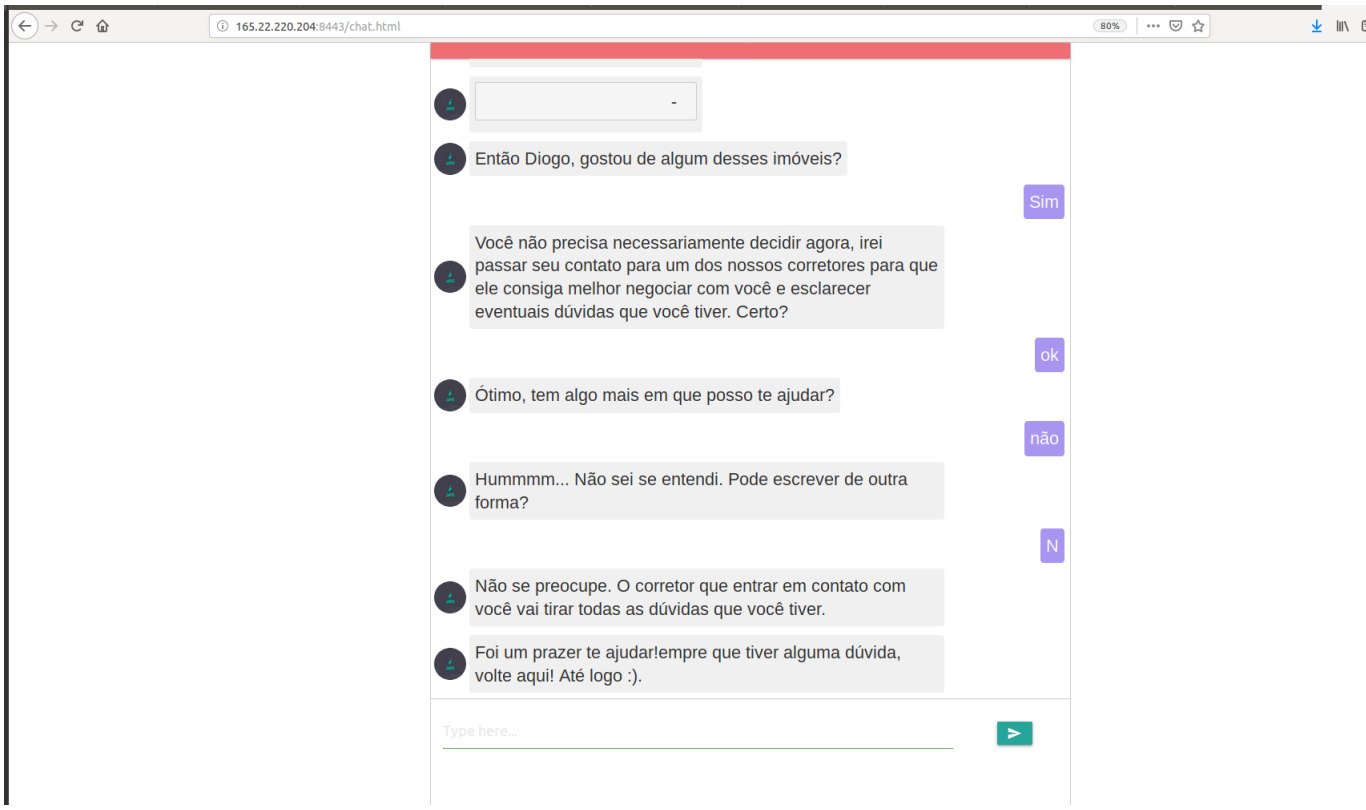
Fonte: esta pesquisa

Figura 16 – Conversando com o *chatbot* Lia..



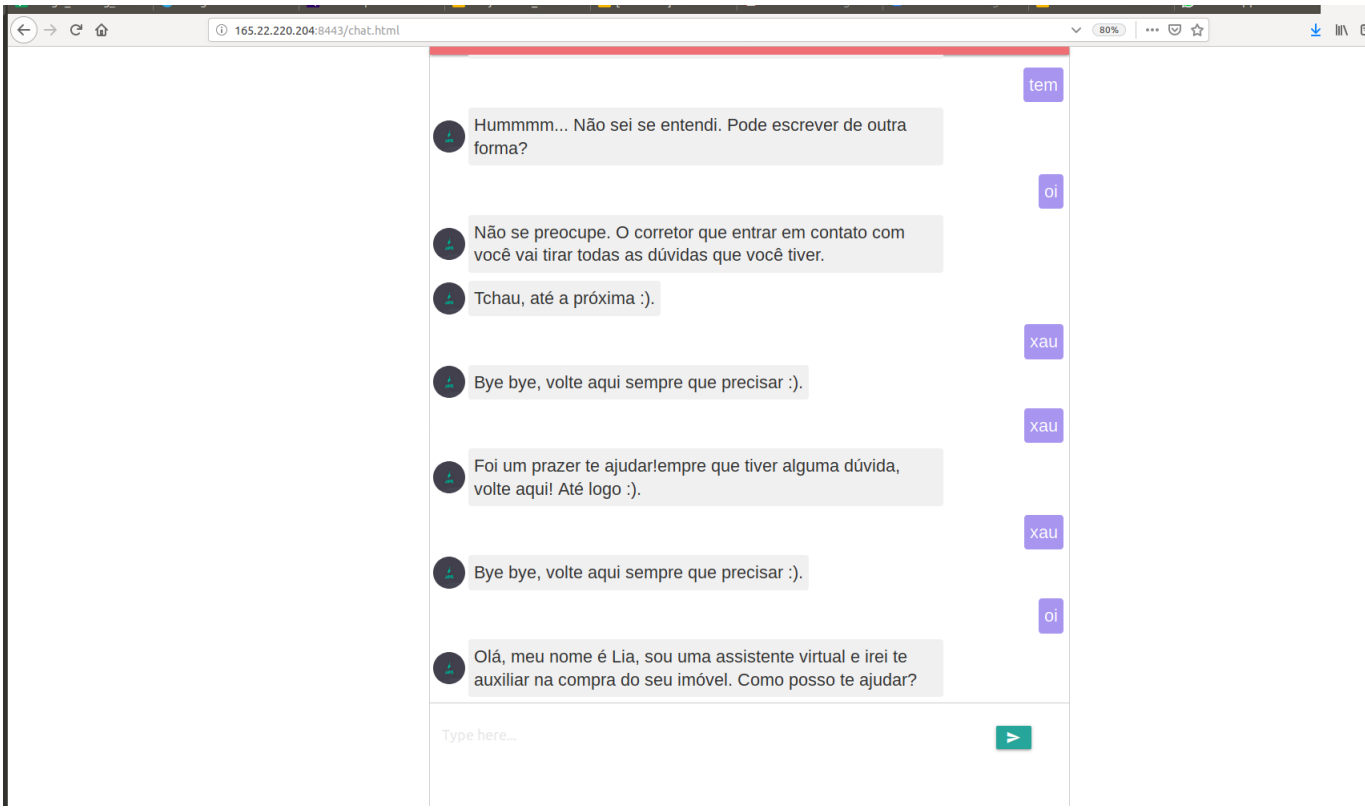
Fonte: esta pesquisa

Figura 17 – Conversando com o *chatbot* Lia..



Fonte: esta pesquisa

Figura 18 – Conversando com o *chatbot* Lia..



Fonte: esta pesquisa

4.7 Cronograma de atividades

Figura 19 – Cronograma de atividades do plano de trabalho.

Item	Atividade	2018		2019									
		nov	dez	jan	fev	mar	abr	mai	jun	jul	Agos	Set	Out
1	**Análise e Definição do Tema**												
1.1	Design sprint, thinking e canvas do Chatbot	X											
1.2	Minicursos sobre busca de anterioridade		X										
1.3	Apropriação do Tema		X										
2	**Introdução**												
2.1	Objetivos		X										
2.2	Definição do método de pesquisa		X										
3	**Fundamentação Teórica**			X									
3.1	Conceitos de chatbot				X								
3.3	Processamento de Linguagem Natural				X								
3.4	Aprendizagem de Máquina				X								
4	**Trabalhos Relacionados**					X							
4.1	Revisão sistemática do estado estado da arte					X							
4.2	Revisão de mercado						X						
4.3	Considerações finais												
	Desenvolvimento							X					
5.1	Descrição geral do sistema							X					
5.2	Requisitos do sistema							X					
5.3	Ferramentas e tecnologias utilizadas								x				
5.4	Representação da arquitetura								x				
5.5	Implementação							x	x	x	x	x	x
5.6	Testes de conversação								x	x	x	x	x
6	**Considerações finais**												x

Fonte: esta pesquisa

5

Considerações Finais

Com o avanço dos estudos na área de Inteligência Artificial, os *chatbots* estão cada vez mais presentes e popularizados, sendo em forma de serviço de Atendimento ao cliente, em forma de comunicação e marketing ou até em formas mais avançadas como realização de transações financeiras..

O presente trabalho visou o desenvolvimento do *chatbot* LIA para tratar sobre questões do domínio de imóveis mais especificamente compra e venda. Uma das principais vantagens na utilização de *chatbots* baseados em inteligência artificial é que eles podem reduzir em mais de 50% os gastos de empresas com material humano, visto que eles mantém uma conversa simples, intuitiva e sem perder o contexto. Além do mais, o curto tempo de resposta e a capacidade de interagir com milhões de usuários simultaneamente fazem dos *chatbots* uma ferramenta de grande utilidade comercial.

A implementação do *chatbot* LIA foi realizada por meio das tecnologias de código aberto *Rasa*, *MongoDB*, *Docker-compose* e *GitHub*. Sendo assim, este estudo verificou que é possível optar por uma alternativa de código aberto e grátis obtendo eficiência no desenvolvimento em contrapartida as soluções proprietárias. Após o desenvolvimento do *chatbot* pode-se concluir, a partir da primeira tarefa de verificação, que ele conseguiu responder todas as perguntas realizadas como demonstrado na seção testes de conversação.

Ficou comprovado nos testes de conversão que o *chatbot* LIA atingiu os objetivos propostos uma vez que conseguiu manter o diálogo da conversa sem sair do contexto e respondendo a todas as perguntas realizadas nos testes. Também vimos que ele abordou todos os requisitos funcionais e regras de negócio definidas neste estudo.

Por último, para trabalhos futuros faz-se necessário ampliar a base de conhecimento do *bot*, isto é, desenvolver mais *intents*, *stories* e o *tamplate* do domínio. Além disso, também é de suma importância ampliar a base de dados e incorporar algum tipo de algoritmo de recomendação que consulte os dados do banco e faça recomendações específicas de imóveis a potenciais clientes.

Referências

AL-ZUBAIDE, H.; ISSA, A. A. Ontbot : Ontology based chatbot. 2016. Citado 2 vezes nas páginas 34 e 39.

ALLEN, J. Natural language understanding. The Benjamin/Cummings Pub., p. 654, 1995. Citado 3 vezes nas páginas 23, 24 e 26.

AMEUR, R.; HEUDIN. Interactive intelligent agent architecture. 2006. Citado 2 vezes nas páginas 34 e 35.

ATIYAH, A.; JUSOH, S.; ALMAJALI, S. An efcient search for context-based chatbots. 2018. Citado 2 vezes nas páginas 34 e 39.

AUGELLO, A. et al. A modular framework for versatile conversational agent building. 2011. Citado 2 vezes nas páginas 34 e 38.

BOUILLON, P. Traitement automatique des langues naturelles. p. 245, 1998. Citado na página 24.

BRANDTZAEG, P. B.; FØLSTAD, A. Why people use chatbots. p. 377–392, 2017. Citado 2 vezes nas páginas 15 e 22.

COPPIN, B. Inteligêncial artificial. Gen LTC, p. 234, 2017. Citado na página 27.

CORONADO, M.; IGLESIA, C. A.; MARDOMINGO, A. A personal agents hybrid architecture for question answering featuring social dialog. 2015. Citado 2 vezes nas páginas 34 e 37.

EVENS, M. W. Relational model of the lexicon: Representingknowledge in semantic networks,. New York: Cambridge University Press,, p. 41–74, 1992. Citado na página 26.

FRANCONI, E. Description logics for natural language processing. 2001. Citado na página 23.

GUTHRIE, L. et al. The role of lexicons in natural language processing. Communications of the ACM, V.39,N.1,, p. 63–72, 1996. Citado na página 25.

GóMEZ, A.; ROPERO, J.; LEóN, C. A fuzzy logic system for classifying the contents of a database and searching consultations in natural language. 2006. Citado 2 vezes nas páginas 34 e 37.

JACOB, R. J. User interface. john wiley and sons ltd. 2003. Citado na página 20.

JUSTO, A. V. et al. Exploring ontologies to improve the empathy of interactive bots. 2018. Citado 2 vezes nas páginas 34 e 38.

KAR R.; HALDAR, R. Applying chatbots to the internet of things: Opportunities and architectural elements. 2016. Citado 3 vezes nas páginas 21, 22 e 23.

KARVE, S. et al. Context sensitive conversational agent using dnn. 2018. Citado 5 vezes nas páginas 34, 35, 41, 43 e 49.

KIM, K.; HONG, J.; CHO, S. Intelligent web interface using flexible conversational agent with semantic bayesian networks. 2005. Citado 2 vezes nas páginas 34 e 39.

KITCHENHAM, B. A.; BRERETON, O. P.; BUDGEN, D. Using mapping studies as the basis for further research – a participant-observer case study. Information and Software Technology, v. 53, p. 638 –651, 2011. Citado na página 29.

KITCHENHAM, B. A.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. EBSE 2007-001, Technical Report EBSE, Keele University and University of Durham, 2007. Citado 3 vezes nas páginas 17, 28 e 29.

KOSHINDA, M.; INABA, M.; TAKAHASHI, K. Machine-learned ranking based non-task-oriented dialogue agent using twitter data. 2015. Citado 2 vezes nas páginas 34 e 36.

L'ABBATE, M.; THIE, U.; KAMPS, T. Can proactive behavior turn chatterbots into conversational agents? 2005. Citado 2 vezes nas páginas 34 e 36.

LEE, D.; OH, K. The chatbot feels you – a counseling service using emotional response generation. 2017. Citado 2 vezes nas páginas 34 e 35.

LOBATO, L. M. P. Sintaxe gerativa do português: da teoria padrão à teoria da regência e ligação. Rio de janeiro: Ed. Vigília,, p. 558, 1986. Citado na página 24.

MARTINEZ, W. L. Graphical user interfaces. wiley interdisciplinary reviews: Computational statistics. p. 119–133, 2011. Citado na página 20.

MAZUEL, L.; SABOURET, N. Generic command interpretation algorithms for conversational agents. 2006. Citado 2 vezes nas páginas 34 e 36.

MAZUEL, L.; SABOURET, N.; CHOI, H. Genericnaturallanguagecommandinterpretationinontology-baseddialogue systems. 2006. Citado 2 vezes nas páginas 34 e 35.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. : Fundamentos teóricos em aprendizado de máquina. 2013. Disponível em: <disponívelem:<http://eduardovalle.com/teach/ia368-r/>>. Citado na página 27.

MORNARD, M. C.; BARANAUKAS, J. A. Aplicações de inteligencia articial: Uma visão geral. São Carlos: Instituto de Ciencias Matematicas e de Computação de São Carlos, 2000. Citado na página 27.

NEGI, S. et al. Automaticallyextractingdialogmodelsfromconversationtranscripts. 2009. Citado 2 vezes nas páginas 34 e 40.

NUNES, M. et al. Introdução ao processamento das línguas naturais. Notas Didáticas do ICMC (Instituto de Ciências Matemáticas e de Computação), São Carlos, 1999. Citado 2 vezes nas páginas 24 e 25.

OWDA, M.; BANDAR, Z.; CROCKETT, K. Conversation-based natural language interface to relational databases. 2007. Citado 2 vezes nas páginas 34 e 37.

PAIKARI, E.; HOEK, A. van der. a framework for understanding chatbots and their future. p. 13–16, 2018. Citado na página 21.

PETERSEN, K. et al. Systematic mapping studies in software engineering. p. 2–3, 2008. Citado na página 28.

RAVI, R. Intelligent chatbot for easy web-analytics insights. 2018. Citado 2 vezes nas páginas 34 e 38.

RUSSEL, S.; NORVIG, P. A. Artificial intelligence: A modern approach. 1995. Citado na página 26.

SAINT-DIZIER, P. On the polymorphic behavior of word-senses. p. 29–56, 1999. Citado na página 26.

SCAPINI, I. K. Relações entre itens lexicais. 1º Encontro do CELSUL, Florianópolis, p. 393–429, 1995. Citado na página 25.

SGANDERLA, R. B.; FERRARI, D. N.; GEYER, C. F. Bonobot: Um chatterbot para interação com usuários em um sistema tutor inteligente. p. 435–444, 2003. Citado 2 vezes nas páginas 15 e 20.

SHAH, A. et al. Problem solving chatbot for data structures. 2018. Citado 2 vezes nas páginas 34 e 38.

SIMON, P. Too big to ignore: The business case for big data. 2013. Disponível em: <disponívelem:<http://www.uel.br/cce/dc/wp-content/uploads/TCC-AndreDelGrossi-BCC-UEL-2013.pdf>.> Citado na página 27.

SINGH, R. et al. Chatbot using tensorflow for small businesses. 2018. Citado 7 vezes nas páginas 34, 37, 43, 44, 45, 46 e 49.

VARGHESE, E.; PILLAI, R. A standalone generative conversational interface using deep learning. 2018. Citado 2 vezes nas páginas 34 e 39.

VIEIRA, R.; LIMA, V. L. S. d. Lima, vera l. s. de. lingüística computacional: Princípios e aplicações. JAIA, SBC, Fortaleza, Brasil,, 2001. Citado na página 24.

WILCOX, B.; WILCOX, S. Making it real: Loebnerwinning chatbot design. 2013. Citado 2 vezes nas páginas 34 e 36.

WITTEN, I. H.; FRANK, E. Data mining: Practical machine learning tools and techniques with java implementations (the morgan kaufmann series in data management systems). Morgan Kaufmann, 1999. Citado na página 27.

XU, A. et al. A new chatbot for customer service on social media. In: ACM. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. [S.l.], 2017. p. 3506–3510. Citado na página 15.