UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Curso de Bacharelado em Ciência da Computação

Trabalho de Conclusão de Curso

C-BOT: um chatterbot para o ensino de programação

João Eduardo Seffrin Soares

Pelotas, 2022

João Eduardo Seffrin Soares

C-BOT: um chatterbot para o ensino de programação

Trabalho de Conclusão de Curso apresentado ao Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Profa. Dra. Larissa A. de Freitas

Pelotas, 2022

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

S676c Soares, João Eduardo Seffrin

C-BOT : um chatterbot para o ensino de programação / João Eduardo Seffrin Soares ; Larissa Astrogildo de Freitas, orientadora. — Pelotas, 2022.

46 f. : il.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) — Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2022.

1. Chatterbot. 2. Aprendizado. 3. Programação. 4. Ensino. I. Freitas, Larissa Astrogildo de, orient. II. Título.

CDD : 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

João Eduardo Seffrin Soares

C-BOT: um chatterbot para o ensino de programação

Trabalho de Conclusão de Curso aprovado, como requisito parcial, para obtenção do grau de Bacharel em Ciência da Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 9 de junho de 2022

Banca Examinadora:

Profa. Dra. Larissa Astrogildo de Freitas (Orientadora)
Doutora em Computação pela Pontifícia Universidade Católica do Rio Grande do Sul.

Prof. Dr. Guilherme Tomaschewski Netto
Doutor em Oceanografia Física pela Universidade Federal do Rio Grande

Bruno Cascaes Alves.
Graduado em Engenharia de Computação pela Universidade Federal de Pelotas.

RESUMO

SOARES, João Eduardo Seffrin. **C-BOT: um chatterbot para o ensino de programação**. Orientador: Larissa A. de Freitas. 2022. 46 f. Trabalho de Conclusão de Curso (Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2022.

A natureza abstrata dos conceitos de programação se mostra como um dos fatores que explicam as dificuldades que muitos estudantes encontram na absorção destes conteúdos. Além disso, os estudantes precisam se adaptar as diferentes sintaxes de linguagens de programação e entender como trechos de código se relacionam. Logo, abordagens alternativas de ensino de programação, principalmente em estágios iniciais, são necessárias. Tendo em vista este contexto, o objetivo deste trabalho é criar um chatterbot para auxiliar no ensino de programação. O chatterbot desenvolvido, denominado C-BOT, é capaz de dissertar sobre certos conceitos da programação e testar os conhecimentos do estudante de forma mais intuitiva e engajadora.

Palavras-chave: Chatterbot. Programação. Ensino. Aprendizado.

ABSTRACT

SOARES, João Eduardo Seffrin. **C-BOT: a chatterbot for teaching programming**. Advisor: Larissa A. de Freitas. 2022. 46 f. Undergraduate Thesis (Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2022.

One of the factors that explain students' difficulties in programming is the abstract nature of programming concepts. In addition, students need to adapt to the different syntaxes of programming languages and understand how to code snippets relate. Therefore, alternative approaches to teaching programming, especially in the early stages, it is necessary. Having this context in mind, the objective of this work is to create a chatterbot to assist in teaching programming. The chatterbot developed, called C-BOT, can lecture on certain programming concepts and test students' knowledge more intuitively and engagingly.

Keywords: Chatterbot. Programming. Teaching. Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AIML	*Artifical Intelligence Markup Language*
API	*Application Programming Interface*
AM	Aprendizado de Máquina
IA	Inteligência Artificial
NLU	*Natural Language Understanding*
PLN	Processamento de Língua Natural

SUMÁRIO

1 INTRODUÇÃO

A crescente importância da área tecnológica em nossa sociedade demanda por profissionais qualificados para atuar em diversos espaços do mercado de trabalho (GLASS, 2016), sendo a criação e a compreensão de códigos-fonte fundamental dentro desta qualificação. Em contrapartida, é frequentemente reportado que os cursos relacionados a área de tecnologia da informação enfrentam altas taxas de desistências e reprovações, tanto no exterior (N. ASHILL, 2016), como no Brasil (A. SACCARO, 2019).

Um dos principais fatores por trás do alto índice de evasão está relacionado com a atividade de programar, geralmente precedido da presença de frustração e da falta de motivação com a programação em si (BENNEDSEN, 2008), percebida pelos estudantes como complexa e de difícil aproximação (T. DARADOUMIS J. M. M. PUIG, 2016).

Jenkins (JENKINS, 2002) descreve que diversos fatores podem influenciar negativamente o aprendizado da programação, podendo ser: o nível abstrato dos conceitos de programação; diferentes competências necessárias na resolução de problemas; pouco ou nenhum contato anterior com o pensamento computacional ou a necessidade de aprender diversas sintaxes, semânticas e estruturas particulares de cada linguagem de programação.

Cursos de programação são geralmente compostos de múltiplas aulas que abordam conceitos básicos da computação, combinadas a tutoriais em classe que exercitam a prática da construção de códigos. Contudo, é essencial que o aluno seja encorajado a praticar o ato de programar em um contexto extraclasse (HOBERT, 2019). A prática é fundamental para sedimentar os conceitos aprendidos em aula, no entanto, mesmo que um entendimento geral desses conceitos já seja obtido, esse não torna o processo de programar menos desafiador para os estudantes.

Hobert (HOBERT, 2019) destaca que o processo criativo de programar demanda muito dos estudantes, essa situação se agrava quando a orientação por parte do professor carece de alguma maneira, por exemplo: as restrições do tempo destes profissionais e consequentemente a impossibilidade de acompanhar de forma contínua o

progresso individual de cada aluno.

Os trabalhos de (T. CROW, 2018) e (S. L. LAU, 2018) destacam que uma gama de ferramentas que auxiliam o aprendizado da programação no contexto extracurricular existem na literatura, particularmente na forma de tutores virtuais. Essas ferramentas buscam trazer um nível de suporte aos alunos e também diminuir a quantidade de suporte necessário vindo dos professores. Porém, segundo os autores, esses sistemas não oferecem um nível de suporte considerado tão eficiente quanto o aprendizado em classe, com contato direto com o professor.

Chatterbots podem ser definidos como sistemas de informação que dispõem de uma interface baseada em linguagem natural (M. SOLLNER, 2018). Geralmente busca-se conduzir conversações de uma maneira proativa ou reativa, a partir da interação com o usuário (HOBERT, 2019). Atualmente, diversas pesquisas buscam trazer os chatterbots para um contexto pedagógico, sendo que esse foco vem após anos de evolução desde os primeiros chatterbots (WOLFF, 2019). A utilização de chatterbots em um contexto educacional abrange diversos espectros, como: suporte pedagógico (F. CLARIZIA F. COLACE, 2018) e sistemas de perguntas e respostas (S. BASAK, 2020).

Chatterbots educacionais mostram-se efetivos em auxiliar no aprendizado de conteúdos difíceis (F. CLARIZIA F. COLACE, 2018), já que, além de possuirem a capacidade de adaptação a individualidades, eles permitem aos estudantes aprender em seu próprio ritmo (M. SU, 2020) de forma ágil, motivadora e inspiradora (B. J. KUN, 2019).

O presente trabalho tem como objetivo apresentar um chatterbot capaz de auxiliar o aluno a programar, sendo capaz de dissertar sobre conceitos fundamentais de programação estruturada, tais como: seleções, iterações e a utilização de variáveis. Assim como, verificar a integridade de códigos na sintaxe da linguagem de programação C, tornando o aprendizado flexível e de rápido acesso.

Para atingir este objetivo, buscamos estudar e implementar técnicas de PLN no desenvolvimento de um chatterbot, empenhando características de chatterbots educacionais já existentes que apresentaram resultados positivos para o ensino, como também a modelagem de uma base de dados que alimente o chatterbot de modo a fornecer conhecimentos sobre algoritmos relevantes para a formação do aluno. Por fim, buscamos argumentar a favor da utilização de métodos de aprendizado de programação alternativos e mais acessíveis em um contexto extracurricular.

Este documento é organizado da seguinte forma: o capítulo 2 descreve o referencial teórico sobre os conhecimentos utilizados neste trabalho; o capítulo 3 descreve trabalhos na literatura com características similares ao do presente trabalho; o capítulo 4 descreve o desenvolvimento do C-BOT; o capítulo 5 descreve os resultados obtidos; por fim, o capítulo 6 apresenta as considerações finais e os trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo o referencial teórico é descrito, ou seja, os principais conceitos para o entendimento deste trabalho, são eles: Processamento de Língua Natural, Chatterbots, plataformas utilizadas na construção de Chatterbots e Linguagens de Programação.

2.1 Processamento de Língua Natural

Segundo (L. MOUSSIADES, 2020), o Processamento de Língua Natural (PLN) é uma área de conhecimento da Inteligência Artificial (IA) e geralmente tem base em técnicas de Aprendizado de Máquina (AM), com o objetivo geral de permitir a um computador entender e manipular linguagens naturais no empenho de diversas tarefas. O processamento engloba a manipulação de uma linguagem natural em forma de texto escrito ou falado através de um computador, a partir do desenvolvimento de certas técnicas, como NLU (*Natural Language Understanding*), e chatterbots (JUNG, 2019). NLU extrai o significado de um trecho de texto escrito pelo usuário e responde este texto de forma apropriada. Para isso, dada uma mensagem, o chatterbot tem como objetivo identificar qual a intenção do usuário com essa mensagem.A intenção pode ser uma saudação ou uma pergunta específica. Após, será definido que tipo de ação deve ser tomada para cada intenção. Ainda, entidades podem ser extraídas da mensagem. Entidades podem ser definidas como diferentes valores dentro do texto que serão relevantes para o entendimento do texto (por exemplo: nome de pessoas, locais, etc.). Contextos também são relevantes, esses são cadeias de informação armazenadas pelo chatterbot durante a conversação. Os quais podem ser relevantes em diferentes momentos da conversa (JUNG, 2019).

2.2 Chatterbot

Um chatterbot tem como objetivo conduzir conversações de maneira similar a seres humanos, podendo responder de maneira reativa ou pró-ativa (HOBERT, 2019). Interações com chatterbots geralmente acontecem a partir de interfaces em linguagem

natural. Esse objetivo é atingido através da aplicação de metodologias de diferentes áreas como IA, AM ou PLN. A entrada e a saída em linguagem natural é processada por um componente presente na arquitetura do chatterbot, este componente processa a entrada inserida pelo usuário e computa uma resposta apropriada a partir de uma banco de conhecimento, assim o componente de geração de linguagem natural gera uma saída que é mostrada ao usuário (K. RAMESH S. RAVISHANKARAN, 2017).

Durante a década de 60 foi desenvolvido o primeiro chatterbot, conhecido como Eliza, o qual utilizava mecanismos simples para a realização de conversas, como as técnicas de "*Pattern Matching*", comparando a entrada do usuário com uma resposta dentro da base de conhecimento do chatterbot, porém, isso tornava as respostas repetitivas e robóticas (L. MOUSSIADES, 2020). Em 1995 surgiu o chatterbot conhecido como ALICE, o qual utilizou de métodos que seriam a base para métodos modernos, dispondo de AIML (do inglês, *Artificial Intelligence Markup Language*), uma linguagem de marcação baseada em "*tags*", um aprimoramento das técnicas de "*Pattern Matching*" e reconhecimento de padrões (L. MOUSSIADES, 2020). Atualmente, a evolução no campo de PLN e o crescimento do interesse comercial em chatterbots permitiu que esses espalhassem-se para diversos domínios, chatterbots modernos notáveis são assistentes virtuais, como Siri[1], Cortana[2] e Alexa[3] (L. MOUSSIADES, 2020).

Adamopoulou e Moussiades (L. MOUSSIADES, 2020) destacam que os chatterbots podem ser classificados a partir de diferentes parâmetros, como seu domínio de conhecimento, serviços oferecidos ou objetivos. Eles podem ser divididos em 3 categorias levando em consideração o processamento de texto de entrada e a geração das respostas. A maioria dos primeiros chatterbots desenvolvidos utilizava a categoria "*Rule-based*" (HOBERT, 2019), esses chatterbots baseiam-se em um conjunto de regras formadas para reconhecer o significado (formato léxico) de diferentes palavras, sua base de conhecimento é organizada na forma de padrões de conversações. Um banco de dados baseado em regras permite ao chatterbot responder a tipos diferentes de texto, porém esses sistemas não respondem muito efetivamente aos erros de gramática por parte do usuário e consideram apenas a mensagem enviada na geração de uma resposta (do inglês, *single-turn conversation*)(L. MOUSSIADES, 2020). A categoria "*Retrieval-based*" oferece mais flexibilidade ao selecionar diferentes respostas candidatas a partir da base de conhecimento, então aplicando um método de casamento (do inglês, "matching") com a entrada do usuário, para selecionar a resposta mais apropriada (L. MOUSSIADES, 2020). A categoria "*Generative*" é considerado mais eficiente que as anteriores, já que considera mensagens anteriores do usuário como contexto para a geração de respostas, utiliza métodos de AM e Aprendizado

[1] https://www.apple.com/siri/
[2] https://www.microsoft.com/en-us/cortana
[3] https://developer.amazon.com/pt-BR/alexa

Profundo (do inglês, *Deep Learning*), porém, apresenta desafios na sua construção e treinamento (L. MOUSSIADES, 2020).

Recentemente os chatterbots espalharam-se para diversos domínios, isso resultou em uma disponibilidade e interesse crescente em chatterbots pedagógicos (HOBERT, 2019). Estudos relatam que diversos chatterbots pedagógicos cumprem o seu objetivo de auxiliar no ensino (F. CLARIZIA F. COLACE, 2018), outro fator positivo é que certos estudantes relatam que se sentem mais confortáveis ao interagir com um chatterbot em comparação a um amigo ou a um professor (R. CARPENTER, 2006). Chatterbots tem a habilidade de apontar às individualidades que cada aluno passa no processo de aprendizado, como também características compartilhadas entre a turma (O. KNILL J. CARLSSON, 2004). Eles também mostram-se capazes de auxiliar professores em ensinar, ao engajar estudantes e simplificar certos passos do aprendizado (A. ADE-IBIJOLA, 2021). Segundo (HUBERT, 2019), o impacto que os chatterbots e a IA podem trazer a educação é espantoso, trazendo diversas dinâmicas únicas, como reforço de conteúdos,fornecimento de *feedbacks* e pontuações automáticas. No campo da computação, os chatterbots fornecem conteúdos para o aprendizado de programação básicas ou fornecem uma maneira de testar conhecimentos, como quizzes (HOBERT, 2019), e buscam empenhar um papel similar ao de um professor, através de interações baseadas em linguagem natural.

2.3 Plataformas de Desenvolvimento de Chatterbot

O desenvolvimento de chatterbots geralmente se dá através de linguagens de programação como Java e Python ou através da utilização de plataformas estado-da-arte. Essas permitem a construção de chatterbots de uma maneira rápida ao oferecer métodos predefinidos de AM e PLN, além da integração com plataformas externas através de APIs, facilitando a adição de recursos ao chatterbot (L. XUAN T. PHAM, 2018). Essas plataformas geralmente possuem funcionalidades em comum como funcionamento na nuvem e a compatibilidade com diferentes linguagens de programação, porém podem divergir em certas características. São exemplos de plataformas: Rasa[4], Dialogflow[5] (Google), Azure Bot Framework[6] (Microsoft), Alexa[7] (Amazon) e Bots na aplicação Messenger[8] (Facebook) (A. ADE-IBIJOLA, 2021).

[4]https://rasa.com/
[5]https://dialogflow.cloud.google.com/
[6]https://dev.botframework.com/
[7]https://developer.amazon.com/pt-BR/alexa
[8]https://developers.facebook.com/docs/messenger-platform/

2.4 Linguagens de Programação

Linguagens de programação são notações para a construção de algoritmos que empenham certo tipo de tarefa, geralmente através da computação em uma máquina abstrata (AABY, 1996). Esse tipo de linguagem geralmente contém abstrações, como a criação de objetos-conceito e a manipulação de seus atributos. Esses objetos são geralmente definidos como estruturas de dados ou variáveis, e também contam com a manipulação de estruturas de controle, essas ditam o fluxo de controle de um código, ou seja, a ordem em que instruções e linhas de código serão executadas (SCHMIDT, 1994). O conhecimento sobre estrutura de dados, variáveis e manipulação de estruturas de controle são habilidades fundamentais para resolver problemas através da programação (CHAO, 2020).

2.4.1 Linguagem de Programação C

Linguagens de programação estruturadas são largamente adotadas no ensino de programação introdutória (A. PEARS S. SEIDMAN, 2009), linguagens estruturadas tornam o aprendizado facilitado ao utilizar estruturas de controle básicas em seu design. Uma das linguagens de programação mais populares da atualidade é a linguagem de programação C[9]. A linguagem de programação C foi desenvolvida por Dennis Ritchie em 1973 [10]. Esta linguagem apresenta escopo léxico em suas variáveis, o que permite variáveis definidas fora de funções a serem utilizadas em todo o código. [11].

- **Variáveis e Tipos de Dados**

  Variáveis são espaços de armazenamento abstratos, associados a um endereço da memória, que possuem um nome distinto e atribuída a uma certa quantidade de informação referida como valor. Um valor de uma variável pode ser ou não alterado durante o percurso de um programa (KNUTH, 1997). O valor de uma variável também pode ser impresso, geralmente na forma de texto, muitas vezes através de uma linha de código que referência a variável em questão (IMAGING, 2012). Variáveis podem conter um tipo de dado definido (tais como: números inteiros ou reais, valores lógicos, *arrays*). Tipos de dados irão definir como a informação dentro da variável em questão poderá se comportar e como o programador planeja utilizá-la (SHAFFER, 2011). Uma coleção de valores de um mesmo tipo de dados pode ser definida como uma variável do tipo *array*. Onde cada valor é comumente referenciado através de um número indexador, indicando uma posição na coleção de valores. *Arrays* são representadas por va-

[9] https://www.tiobe.com/tiobe-index

[10] https://www.nytimes.com/2011/10/14/technology/dennis-ritchie-programming-trailblazer-dies-at-70.html

[11] https://courses.cs.washington.edu/courses/cse341/08au/general-concepts/scoping.html

lores em uma sequência na forma de uma linha de uma dimensão (SEBESTA, 2009).

- **Operações Aritméticas**

  Operações aritméticas (tais como: soma, subtração, produto e divisão) envolvendo dados através de variáveis são um dos fundamentos em qualquer linguagem de programação, nessas operações variáveis e seus valores são geralmente utilizadas como operandos, ainda, os operadores estão sujeitos a diferentes precedências de operação `https://en.cppreference.com/w/c/language/operator_precedence`.

- **Função**

  Uma função é uma sequencia de linhas de código, embaladas como uma unidade própria e pode ser utilizada em programas sempre quando sua execução deve ser feita de forma particular em um programa. (STROUSTRUP, 2013)

- **Recursão**

  Recursão, em programação, é o ato de chamar uma função dentro de si mesmo com o objetivo de resolver um problema maior com a execução da função em instâncias repetidas (EPP, 2019),

- **Estrutura de Dados**

  Dados ou variáveis que os encapsulam podem ser organizados em formatos que permitem fácil acesso e modificação, estes formatos são denominados estruturas de dados (L. RONALD, 2009). Um exemplo de estrutura de dados são as árvores (WEGNER, 2003), um tipo onde seu dado, denominado nodo, está presente em uma hierarquia de níveis, e é conectado a nenhum ou mais nodos em um nível inferior, denominados nodo-filho, porém cada nodo é apenas conectado a um nodo-pai em um nível hierarquico superior, exceto no caso do primeiro dado da árvore, denominado raiz, que não possui um nodo-pai (GARNIER, 2009). Árvores binárias são um sub-tipo desta estrutura de dados em que cada nodo possui no máximo até dois nodos-filho, denominados esquerdo e direito.

- **Algoritmo de Busca Binária**

  Um algoritmo de Pesquisa ou Busca Binária tem como objetivo encontrar a posição (indexador) de um valor ’x’ dentro de uma coleção de valores já ordenados do menor para maior (uma *array* ordenada), para isso, o algoritmo compara o valor de ’x’ com o valor localizado na metade da array, se ’x’ é menor que este valor, isso significa que ’x’ pode apenas estar localizado na parte da array anterior ao

elemento metade, já que pela lógica da ordenação, todos números também serão menores, como ’x’ é, a mesma lógica se aplica se ’x’ for maior que o valor metade, assim, o algoritmo repete o processo recursivamente com a metade em questão, realizando a mesma lógica anterior e particionando em diante a coleção de valores total até encontrar o número ou terminar a execução ao restar apenas um elemento em uma partição com (WEISSTEIN, 2021).

- **Algoritmo de Busca em Profundidade**

  Algoritmos de travessia de árvores binárias visitam todos os nodos desta apenas uma vez e são classificados de acordo com a ordem que estas visitas acontecem. Este processo pode também ser chamado de busca ou pesquisa.Na busca em profundidade, visita-se um nodo-filho sempre antes de seu nodo-pai, neste tipo de travessia há ainda três ordens mais comuns implementadas por algoritmos, em-ordem, pré-ordem e pós-ordem. A busca em profundidade em-ordem dá prioridade de visitar nodos-filho esquerdos para depois visitar o nodo-pai destes, então por fim visitando os nodos-filho à direita deste, repetindo a lógica (PFAFF, 2004).

- **Algoritmo de Ordenação**

  Certos algoritmos realizam a ordenação númerica de uma *array*, ou seja, trocam valores de posição de modo que estes valores estejam em ordem númerica (crescente ou decrescente). Algoritmos de ordenação são abordados em aulas de introdução a programação, onde auxiliam na explicação de conceitos de programação como estrutura de dados e recursão. Exemplos de algoritmos de ordenação são: *Bubble Sort* (ASTRACHAN, 2003) ou *Quicksort* (HOARE, 1962)

3 TRABALHOS RELACIONADOS

Neste capítulo os trabalhos relacionados são apresentados. A partir de uma análise da literatura, um número de chatterbots com características similares ao chatterbot proposto neste trabalho foram encontrados. De maneira geral, é de interesse destacar chatterbots que foram utilizados em um contexto de ensino de programação ou de idiomas.

Em um estudo feito por 5 estudantes da Universidade do Vietnã (L. XUAN T. PHAM, 2018), é apresentada uma aplicação móvel, desempenhando o papel de um assistente pessoal no aprendizado da língua inglesa. Um chatterbot foi desenvolvido como uma das ferramentas da plataforma, este empenha diversas funções como a notificação de estudantes para suas lições e sugerir dicas para quizes. O desenvolvimento do chatterbot se deu a partir da plataforma DialogFlow, no que empenha-se o uso de diferentes contextos para uma conversa, a utilização de contextos permite ao chatterbot responder apropriadamente, os principais contextos são: Contexto geral, para conversas casuais; Contexto de Quiz, onde espera-se respostas do usuário; Contexto de aprendizado, em que o chatterbot fornece o conteúdo na forma de cartas, o usuário pode requisitar exemplos ou discutir sobre o conteúdo em questão; Contexto de revisão, onde o chatterbot busca reforçar um conteúdo já aprendido utilizando subcontextos já descritos. A comunicação com o chatterbot acontece a partir de diferentes situações: interações casuais; responder a requisições de conteúdo (quizzes ou lições de gramática e vocabulário); oferecer dicas ou explicações para um certo problema de aprendizado; oferecer reforço de conteúdos já aprendidos por um usuário. Outra parte interessante do chatterbot é a opção da utilização de comandos predefinidos para a conversação, isso permite ao usuário escolher opções de palavras ou frases ao invés de digitar e é uma característica com a capacidade de auxiliar o usuário quando há uma dificuldade de interagir com o chatterbot, por razões como uma barreira linguística ou dificuldade de entender o funcionamento do aplicativo, ainda assim, a partir da coleta de dados referente ao período de dois meses, em que o sistema foi disponibilizado através da Google Play Store[1], foi relatado que a proporção e quantidade

[1] https://play.google.com/store/

de interações com o chatterbot foi substancialmente baixa, isso foi atribuído a uma possível falta de familiarização inicial para com o chatterbot.

Hobert (HOBERT, 2019) apresenta o projeto de um sistema de ensino de programação inteligente, denonimado Coding Tutor. O sistema é capaz de comunicar-se com o usuário através de um chatterbot e também conta com um editor de código, buscando auxiliar estudantes individualmente na construção de códigos, especialmente quando há a indisponibilidade de tutores humanos. O artigo apresenta as capacidades do chatterbot, ele é capaz de apresentar os conceitos teóricos por trás dos exercícios; corrigir os exercícios e dar *feedbacks* aos usuários. O principal método por trás da metodologia de ensino empenhada pelo chatterbot é denominado "caminhos de ensino", esses ditam a ordem de atividades que o chatterbot deve seguir em cada exercício, um exemplo é a necessidade, presente ou não, da introdução à um certo conteúdo revelante ao exercício, ou a requisição de orientação detalhada ou suplementar após a explicação inicial, logo, esses caminhos variam de acordo com a individualidade dos estudantes. Isso permite que estudantes com mais experiência não sejam submetidos a mesma concentração de conteúdo que estudantes inexperientes.

O sistema foi desenvolvido utilizando tecnologias baseadas nas linguagens HTML, CSS e JavaScript, a interface foi desenvolvida a partir da framework Bootstrap 4[2] e o editor de código foi construído a partir do editor Ace[3]. Uma avaliação foi feita através de um questionário aplicado a 40 estudantes de um curso de programação introdutório, esses estudantes foram previamente introduzidos ao software através da possibilidade de utilizar o chatterbot ao resolver um exercício específico de programação.

Daud et al. (S. H. M. DAUD, 2020) propõem um chatterbot que tem como objetivo auxiliar o aprendizado de conceitos de programação em Java, mais precisamente, conceitos de seleção e repetição. O chatterbot seria capaz de prover explicações em relação aos conteúdos de programação apartir de perguntas dos estudantes pelo módulo de conversação do chatterbot. O chatterbot também é capaz de fornecer a visualização de códigos exemplo que utilizam conceitos de seleção (*if*, *else*) e repetição (*while*, *do while*, *for*). Os autores não tiveram a oportunidade de usar o chatterbot em um contexto prático e planejam melhorar o chatterbot futuramente.

Python-bot é um chatterbot apresentado por Okonkwo Chinedu e Ade-Ibijola em seu artigo (A. ADE-IBIJOLA, 2021), o chatterbot é capaz de explicar conceitos de programação a alunos, fornece problemas de programação predefinidos e permite ao usuário entrar em contato com um tutor humano caso necessário. É relatado que a API utilizada na implementação é conhecida como SnatchBot, uma plataforma web para a construção de chatterbots, oferecendo um editor em maior parte livre de código, a plataforma fornece ao desenvolvedor uma unidade de interface e uma unidade de conhe-

[2]https://getbootstrap.com/
[3]https://ace.c9.io/

cimento baseada em AIML. Para o processamento do texto do usuário, a plataforma conta também com um banco de respostas predefinidas que permite a aplicação do chatterbot na web através de uma unidade de integração. O desenvolvimento do chatterbot na plataforma acontece primariamente através da criação de interações, estas ditam os tipos de tópicos abordados na conversa. No caso do Python-bot, exemplos de interações definidas são : Introdução, Nome, Adeus, Exercícios, Conceitos, entre outros. O SnatchBot suporta vários algoritmos de PLN já predefinidos dentro da API, para a utilização dos módulos de PLN foram definidos diversos tipos de entidades e intenções para que Python-bot seja capaz de entender o contexto a partir de uma entrada e responder apropriadamente. Python-bot começa uma interação ao solicitar ao usuário que ele se apresente, com o objetivo de armazenar o nome do aluno como também o tempo total de cada conversa. Então o chatterbot fica a disposição do aluno para detalhar explicações conceituais de Programação de Python ou o passo-a-passo da construção de códigos. O uso do chatterbot no aprendizado da linguagem Python foi testado por estudantes universidade de Johanesburgo, a partir de uma avaliação de opinião do uso do chatterbot, cerca de 80% da turma se mostraram favoráveis ao uso dele.

Johnson et al. (JOHNSON; E. ZANOLI, 2021) faz o uso do chatterbot em uma maneira gamificada para também auxiliar o aprendizado remoto da língua inglesa, inspirando-se em "*escape rooms*", onde um número de pessoas deve descobrir pistas e resolver enigmas dentro de um determinado ambiente para atingir um certo objetivo, o chatterbot é construído através do Telegram. As atividades oferecidas pelo chatterbot envolvem uma dinâmica de grupo em uma mesma sala de chat, estas podem ser: (i) correção de sentenças; (ii) adivinhar palavras a partir de dicas fornecidas pelo chatterbot em um período de tempo; e (iii) discussão entre os usuários a partir de um contexto oferecido pelo chatterbot no tempo de três minutos. O chatterbot foi testado por 12 estudantes russos com inglês em nível intermediário, os estudantes apresentaram satisfação em testar o chatterbot. Os desenvolvedores planejam construir mais ferramentas para o chatterbot, como por exemplo elementos visuais e a utilização de mais técnicas de PLN para a avaliação de desempenho.

Pode-se observar que os chatterbots descritos são similares em certos aspectos, por exemplo todos conseguem explicar alguma forma de conhecimento através de textos emitidos para o usuário. No caso de chatterbots voltados para o escopo da programação temos funcionalidades que exploram a introdução ao modo de programar códigos-fontes de maneiras diversas. Essas funcionalidades são familiares com as funcionalidades oferecidas por C-BOT. A Tabela 1 exibe os trabalhos relacionados e suas características classificadas em colunas, apresentando a plataforma utilizada para a construção (framework), as funcionalidades em termos gerais, a área de atuação e linguagem de programação / idioma tratado.

Tabela 1 – Sumarização dos trabalhos relacionados.

Trabalho	Framework	Propósito	Área	Linguagem de Programação/ Idioma
(L. XUAN T. PHAM, 2018)	Dialogflow	Quizzes e Explicações	Idiomas	Inglês
(HOBERT, 2019)	-	Construção de Algoritmos	Programação	C++
(S. H. M. DAUD, 2020)	-	Explicações e Visualização de Algoritmos	Programação	Java
(A. ADE-IBIJOLA, 2021)	SnatchBot	Explicações e Construção de Algoritmos	Programação	Python
(JOHNSON; E. ZANOLI, 2021)	-	Explicações e Gameficação	Idiomas	Inglês
Trabalho Proposto	Rasa	Quizzes, Explicações, Construção e Visualização de Algoritmos	Programação	C

4 ABORDAGEM PROPOSTA

Neste capítulo detalhes referentes a abordagem proposta são explicados. Primeiramente será detalhado o funcionamento geral do chatterbot proposto (C-BOT), logo após a metodologia de ensino usada será apresentada, também será apresentado o banco de dados de treinamento do chatterbot proposto e o seu fluxo de conversação.

4.1 Funcionamento Geral do C-BOT

Para o desenvolvimento do chatterbot, foi escolhido a framework Rasa. Rasa conta com um módulo de PLN, que extrai intenções e suas entidades a partir de uma solicitação do usuário ("*Request*" e "*Interpreter*"), buscando compreender a vontade do usuário (T. BOCKLISCH J. FAULKNER, 2017).

De forma geral o funcionamento do chatterbot começa no envio de uma mensagem de texto pelo usuário através do app Messenger para uma entidade que representa o chatterbot e que foi registrada no aplicativo. Esta entidade está conectada à uma máquina que hospeda o servidor responsável pela conexão. Esta máquina também contém a implementação do chatterbot propriamente dito. Após estabelecida a conexão, o texto é recebido pelo chatterbot.

O próximo estágio do funcionamento é o processamento do texto de entrada. O chatterbot possui em sua base de dados, exemplos de entradas de texto e seus respectivos grupos de intenção. Assim, o algoritmo de PLN do chatterbot emite uma previsão sobre qual intenção a entrada de texto atual se encaixa, e o nível de confiança por trás desta previsão, numa escala de 0 a 1.

Após a intenção ser prevista, o framework Rasa deve agir de acordo com esta intenção, para isso, o framework conta com um tipo de dado que dita um fluxo de conversação a partir de cada intenção ou das suas ramificações, as quais podem existir em cada fluxo, chamadas de "*stories*".

Cada "*story*" possui a ordem de ações que um chatterbot deve tomar dada uma combinação de intenções. Estas ações podem ser respostas em texto ou imagens, as imagens são enviadas pelo chatterbot através de "*links*" único, todas as imagens

utilizadas pelo trabalho foram elaboradas e então hospedadas através do website de hospedagem de imagens"*imgur*" em um `https://imgur.com/`. No caso de C-BOT, as "*stories*" podem, por exemplo, indicar quando e como cada quiz começa, pausa ou encerra.

As ações possíveis pelo chatterbot estão inseridas em seu banco de dado. Exitem tipos de ação já implementada pela framework Rasa, como por exemplo, a ação de enviar um trecho de texto como resposta para o usuário. Esta ação deve ser definida em um arquivo domínio do banco de dados e inserida em uma "*story*" para ser executada no momento definido. Porém, ações personalizadas podem ser construídas pelo desenvolvedor e executadas pelo chatterbot. Estas ações personalizadas devem ser implementadas dentro do framework Rasa utilizando a linguagem de programação Python[1]. No caso de C-BOT, a validação das respostas dos quizzes é uma ação personalizada.

C-BOT pode ser encaixado na categoria de chatterbots *Retrieval-based*, tendo em conta o funcionamento descrito, ou seja, a escolha de ações-resposta com base no contexto das conversas (seguindo fluxos de conversação) e o casamento da entrada do usuário com exemplos dentro do banco de dados do chatterbot.

4.2 Metodologia de Ensino do C-BOT

Com o objetivo de ser um chatterbot capaz de auxiliar estudantes a entenderem conceitos e métodos de programação, o funcionamento geral de C-BOT busca simular uma metodologia de ensino que pode ser dividida em duas funcionalidades, são elas: (1) explicações teóricas de conceitos de programação; (2) aplicação de quizzes sobre o funcionamento de certos algoritmos. Ambas funcionalidades são brevemente introduzidas pelo chatterbot após uma mensagem de saudação.

Todas as explicações teóricas de conceitos de programação e de códigos-fonte utilizados pelo C-BOT foram extraídas do website "*geeksforgeeks*"[2]. Este website é uma plataforma voltada para o ensino de conceitos de programação e de algoritmos.

No C-BOT optou-se por tratar sobre conceitos de programação em explicações em texto e introduzir/interagir com algoritmos em quizzes. Os conceitos de programação escolhidos para estarem no escopo do deste chatterbot foram: variáveis, operações aritméticas, recursão e estrutura de dados (árvore binária). Já os algoritmos escolhidos foram: Algoritmo de Busca Binária, Algoritmo de Busca em profundidade e o Algoritmo de Ordenação (*Quicksort*).

Ambas funcionalidades do chatterbot solicitam do usuário algum tipo de requisição. A parte de explicações em texto é geralmente feita pelo chatterbot a partir de

[1] https://docs.python.org/3/

[2] https://www.geeksforgeeks.org/why-geeksforgeeks-is-an-essential-platform-for-cs-it-students/

uma pergunta direta do usuário, como por exemplo: “Como funciona uma recursão?”. O chatterbot discorrerá sobre a natureza de uma recursão em programação, a sua utilidade e exibirá uma imagem com um trecho de código que utiliza uma recursão.

Além da recursão, a versão atual de C-BOT também discorre sobre o uso de variáveis, operações aritméticas e estruturas de dados (árvore binária) na linguagem de programação C. Algumas destas opções de conteúdos são introduzidas pelo chatterbot a partir de uma requisição. Espera-se do usuário o interesse em realizar perguntas sobre diversos assuntos, buscando tornar a busca de conhecimento sobre a linguagem de programação C a partir do uso do C-BOT mais instigante e não tão robótica, mesmo que o banco de dados atual do chatterbot não possua um vasto escopo. Por fim, acreditamos que este modelo de ensino se porte mais naturalmente a medida que novos assuntos sejam adicionados no banco de dados.

Com objetivo de evitar que o fluxo de conversação se torne muito maçante, as explicações são geralmente separadas em partes. Por exemplo, a explicação sobre árvores binárias é dividida em teórica e prática. A partir da requisição inicial do usuário, uma explicação teórica sobre árvores binárias é dada. Após, é perguntado ao usuário se o mesmo deseja aprender mais na forma prática. Se o usuário deseja aprender mais, a explicação continua através do uso de imagens com trechos de código. Caso o usuário já esteja satisfeito com a explicação inicial, a conversa termina.

A parte de quiz é composta de perguntas e respostas. Um quiz é iniciado após a requisição do usuário, logo após o chatterbot discorre sobre as opções de quiz, existindo um para cada algoritmo (Busca Binária, Busca em Profundidade e Quicksort). A escolha é feita através de uma entrada por parte do usuário e a partir daí o chatterbot descreve brevemente o algoritmo que vai ser trabalhado e exibe o algoritmo em uma imagem. Após, é perguntado ao usuário se o mesmo deseja começar o quiz. Caso o usuário concorde em começar o quiz, a execução do quiz se inicia. Caso o usuário não concorde em começar o quiz, a execução do quiz se encerra.

A estrutura da execução do quiz se dá por uma série de explicações em etapas. Cada etapa é composta por uma pergunta que deve ser respondida corretamente para o prosseguimento do quiz. Cada pergunta refere-se a uma linha de código que empenha uma certa ação no código-fonte final. Portanto, cabe ao usuário completar as partes do algoritmo. A visualização do código-fonte final se dá por uma imagem que é enviada no começo do quiz. Nela, as linhas de código que integram o quiz estão incompletas, contudo, optamos por permitir a permanência de certos detalhes em cada linha (como: variáveis). Essa metodologia se assimila a resolução de um quebra-cabeça e busca tornar a construção de um algoritmo mais flexível, trazendo foco para o “como” um algoritmo é construído, assim como estimulando a memória visual dos estudantes ao fornecer a imagem de um código-fonte corretamente compilado a cada acerto.

Por exemplo, no quiz para o algoritmo de Busca Binária, o código-fonte incompleto é mostrado ao usuário. Se o usuário concorda em começar o quiz, o chatterbot prossegue detalhando o funcionamento de cada parte relevante do algoritmo que deve ser representada em uma linha de código pelo usuário. Detalhes referentes à esta execução e ao funcionamento de um algoritmo de Busca Binária serão discutidos no próximo capítulo.

4.3 Banco de Dados de Treinamento do C-BOT

Para que o chatterbot tenha a capacidade de realizar o papel de um tutor de acordo com a metodologia descrita, um banco de dados de treinamento foi criado.

O banco de dados encontra-se no formato ".yml". No arquivo "nlu.yml" são inseridas todas intenções esperadas por parte do usuário em uma conversa juntos de seus respectivos exemplos de mensagens. Algumas intenções previstas pelo C-BOT são ações fundamentais à uma conversa, como uma saudação ou adeus. Outras, são ações personalizadas, como o interesse em aprender recursão ou o desejo de realizar um quiz sobre ordenação. A Figura 1 ilustra as intenções de saudação (intro), adeus e pedido para realizar um quiz (exercício) e os exemplos de mensagens.

```
version: "3.0"
nlu:
- intent: intro
  examples: |
    - oi
    - olá
    - bom dia
    - boa tarde
    - boa noite
    - Vamos lá
    - Eai cara
    - oii

- intent: adeus
  examples: |
    - tchau
    - adeus
    - até mais
    - até mais tarde
    - tenha um bom dia
    - bye bye
    - não obrigado


- intent: exercicio
  examples: |
    - exercicio
    - quero fazer exercicio
    - quais exercicios posso fazer?
    - exercicios
    - quero fazer um quiz
    - quiz
    - fazer quiz
    - fazer exercicio
```

Figura 1 – Três das intenções no arquivo nlu.yml. Fonte: Própria.

Nos arquivos "rules.yml" e "stories.yml" são inseridas regras e histórias. As regras

permitem que o chatterbot siga caminhos de conversação a partir de intenções do usuário, também ditam que tipo de resposta deve ser dada, em cada etapa de um determinado caminho, ou seja, regras ditam caminhos que são imútaveis. Enquanto que, as histórias dão um caminho generializado permitindo variações ou mudanças entre caminhos, caso haja um desvio de intenção. As Figuras 2 e 3 ilustram uma parte destes arquivos, sendo que as histórias mostradas refletem os fluxos tomados pelo chatterbot a partir das intenções de aprender sobre árvores binárias, com o desejo de aprender mais detalhes afirmado e negado em cada intenção, e uma história para o começo de um quiz de busca binária e o caso de que o começo deste seja confirmado pelo usuário. As regras na imagem indicam o fluxo seguido após o pedido por ajuda em cada um dos quizzes.

```
- story: tree
  steps:
  - intent: tree
  - action : utter_tree
  - action : utter_tree_dois
  - action : utter_continue
  - intent : afirmação
  - action : utter_tree_tres

- story: tree deny
  steps:
  - intent: tree
  - action : utter_tree
  - action : utter_tree_dois
  - action : utter_continue
  - intent : negação
  - action : utter_okay

- story: Activate binary search form
  steps:
  - intent: exercicio
  - action: utter_exes
  - intent: bs_exe
  - action: utter_bs_exe
  - action: utter_bs_exe_um_um
  - action: utter_bs_exe_um_dois
  - action: utter_bs_exe_dois
  - action: utter_exe_ask
  - action: utter_exe_ask_dois
  - intent: afirmação
  - action: utter_bs_exe_tres
  - action: utter_bs_exe_quatro
  - action: bs_form
```

Figura 2 – Arquivo de histórias de conversação. Fonte: Própria.

No arquivo “domain.yml” a estrutura e as forma de resposta são declaradas, o arquivo contém o universo com declarações dos seguintes componentes do chatterbot: as respostas do chatterbot para cada interação com o usuário; as estruturas dos quizzes, que são declarados como ”forms”, sendo formulários em que cada resposta é um ”slot” ativado e preenchido pelo usuário;ações personalizadas que validam se um ”slot” foi respondido corretamente; e as intenções no escopo do chatterbot para uma entrada do usuário. A Figura 4 mostra parte do arquivo domínio com todas as intenções declaradas para o uso do chatterbot e duas respostas/ações do chatterbot, no

```
rules:

- rule: ajuda bfs
  condition:
  - active_loop: bs_form
  steps:
  - intent: ajuda
  - action: action_deactivate_loop
  - active_loop: null
  - action: utter_exe_pause

- rule: ajuda dfs
  condition:
  - active_loop: dfs_form
  steps:
  - intent: ajuda
  - action: action_deactivate_loop
  - active_loop: null
  - action: utter_exe_pause

- rule: ajuda qs
  condition:
  - active_loop: qs_form
  steps:
  - intent: ajuda
  - action: action_deactivate_loop
  - active_loop: null
  - action: utter_exe_pause
```

Figura 3 – Arquivo de regras de conversação. Fonte: Própria.

caso as mensagens de saudações.

```
intents:
  - intro
  - adeus
  - afirmação
  - negação
  - declara
  - condicional
  - loop
  - ajuda
  - teste
  - dica
  - challenge
  - exercicio
  - bs_exe
  - dfs_exe
  - qs_exe
  - operar
  - recursar
  - print
  - tree
  - resume
  - code
  - unknown


responses:
  utter_intro:
  - text: "Olá! Eu sou C-BOT e posso lhe ensinar sobre alguns conceitos e algoritmos da linguagem de programação C."

  utter_intro_dois:
  - text: "Você deseja saber detalhes sobre o que posso oferecer?"
```

Figura 4 – Parte do arquivo domínio com intenções e duas ações. Fonte: Própria.

No arquivo “actions.py” foram construidas as ações personalizadas de validação

de cada resposta dos quizzes, elas analisam a entrada do usuário e verificam se esta entrada condiz com a linha de código resposta, essas ações foram desenvolvidas a partir da linguagem de programação Python, que é a linguagem suportada no desenvolvimento de Rasa. A Figura 5 ilustra parte da implementação da classe responsável por validar as respostas do quiz de Busca Binária, contando com dois dicionários para as respostas e parte da validação da primeira resposta.

```
class ValidateBSForm(FormValidationAction):
    def name(self) -> Text:
        return "validate_bs_form"
    @staticmethod
    def bs_um_db() -> List[Text]:
        return ["meio=e+(d-ea)/2;","meio=e+(d-e)/2;"]

    @staticmethod
    def bs_dois_db() -> List[Text]:
        return ["return buscaBinaria(arr,meio+1,dir,x);","buscaBinaraaia(arr,meio+1,dir,x);"

    def validate_bs_um(
        self,
        slot_value: Any,
        dispatcher: CollectingDispatcher,
        tracker: Tracker,
        domain: DomainDict,
    ) -> Dict[Text, Any]:

            if slot_value in self.bs_um_db():
                dispatcher.utter_message(text="Certo! Agora precisamos comparar o valor dent
                dispatcher.utter_message(text="Como você pode ver no código, já há uma condi
                dispatcher.utter_message(text="Mas e a chamada recursão para se 'arr[m]' for
                dispatcher.utter_message(image="https://i.imgur.com/hwb76nP.png")
                return {"bs_um": slot_value}
            else:
                dispatcher.utter_message(text="Tente de novo")
                return {"bs_um": None}
```

Figura 5 – Parte da implementação de ações personalizadas. Fonte: Própria.

Por fim, no arquivo “config.yml” são especificadas e personalizadas as políticas e componentes de PLN que serão utilizados pelo chatterbot para processar textos, este processo é denominado pipeline.

No C-BOT, foram utilizadas as políticas como: *RulePolicy* que indicam o uso de regras para fluxos fixos de conversação; *TEDPolicy* e *UnexpecTEDIntentPolicy* (*Transformer Embedding Dialogue*) que auxiliam na predição de ações do chatterbot, *TED* que utilizam *transformers*, um tipo de modelo de aprendizado profundo, que neste caso, atribui diferentes pesos de importância para partes de um texto (A. VASWANI, 2017) e a consideração das histórias estabelecidas nos dados de treino para a predição de ações, denominada *MemoizationPolicy*.

Também foram utilizados os componentes como *WhitespaceTokenizer* empenhando a tokenização do tipo de espaços em branco, a tokenização é um processo responsável por identificar cada palavra corretamente ao separar o texto por espaços em branco, partes denominadas *tokens*`https://rasa.com/docs/rasa/components/#tokenizers`. Outros componentes utilizados foram *CountVectorsFeaturizer*,o qual agrupa os *tokens* separados pela tokenização junto de intenções e ações-respostas respectivas a fim de classificar a intenção da mensagem, como também a resposta

apropriada junto do componente *ResponseSelector*. O componente *DIETClassifier* se encarrega de classificar as intenções dado uma sequência de tokens. Por fim, o componente *FallbackClassifier*, que permite a customização do nível de confiança que deve ser prevista para uma intenção para que uma mensagem não seja considerada fora de escopo de resposta do chatterbot. A Figura 6 ilustra todas os componentes e políticas relatadas anteriormente. O arquivo também define a língua natural tratada pelo chatterbot. Rasa permite o treinamento de um chatterbot em qualquer língua natural, porém, existe o suporte a determinadas línguas através de de modelos pré-treinados, o que diminui a quantidade de dados necessária para um modelo de PLN funcional[3].

```
language: pt

pipeline:
- name: WhitespaceTokenizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: DIETClassifier
  epochs: 100
  constrain_similarities: true
- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
- name: FallbackClassifier
  threshold: 0.3
  ambiguity_threshold: 0.1

policies:
- name: MemoizationPolicy
  max_history: 2
- name: RulePolicy
- name: UnexpecTEDIntentPolicy
  max_history: 5
  epochs: 100
- name: TEDPolicy
  max_history: 5
  epochs: 100
  constrain_similarities: true
```

Figura 6 – Arquivo de configuração de componentes e políticas. Fonte: Própria.

4.4 Fluxo de Conversação do C-BOT

Na Figura 7 é apresentado o fluxo de conversação do C-BOT. Nela as duas principais funcionalidades, explicações e quizzes, e como a conversação pode fluir de uma para outra em determinados pontos são apresentadas assim como a introdução do chatterbot, que contém a mensagem de saudaçoes do chatterbot junto a uma explicação breve das funcionalidades disponíveis, também consta no fluxograma a mensagem de adeus.

No fluxograma estão as quatro possibilidades da funcionalidade de explicação de

[3]https://rasa.com/docs/rasa/language-support

conceitos, estão disponíveis explicações a perguntas em relação a conceitos de variáveis, operações aritméticas, recursão e estrutura de dados (árvores binárias), a execução do fluxo de explicação para com o conceito de recursão será detalhada no próximo capitulo. Destaca-se que algumas dessas explicações possuem detalhes que só são emitidos pelo chatterbot mediante a afirmação do usuário, caso contrário o fluxo toma um estado neutro em que pode-se perguntar sobre outro conceito ou iniciar um quiz. Se algum conceito não possui detalhes, só a explicação inicial acontece.

A funcionalidade do quiz tem três opções distintas, algoritmos de Busca Binária, Busca em Profundidade e Quicksort estão disponíveis ao usuário após este declarar interesse em realizar um quiz, logo após a escolha, a introdução breve ao algoritmo é feita, e então um pedido de confirmação para o começo do quiz é feito, com as possiblidades de afirmação ou negação, se negado, o fluxo de conversação entra em um estado neutro, se afirmado, o quiz começa, dentro do quiz, a estrutura de funcionamento detalhada na seção de metodologia se inicia, com a possibilidade de erro ou acerto das linhas de código, enquanto há erros, o quiz não se encerra, porém cada acerto avança as etapas do quiz, até o seu encerramento.

Além disso, destaca-se a possibilidade da pausa do quiz em execução, o que permite ao usuário interagir com o chatterbot com explicações, começar um quiz novo ou continuar um quiz de sua escolha, logo, é possível ter múltiplos quizzes em progresso. Na versão atual do chatterbot não é possível encerrar um quiz até ter respondido todas as suas perguntas corretamente.

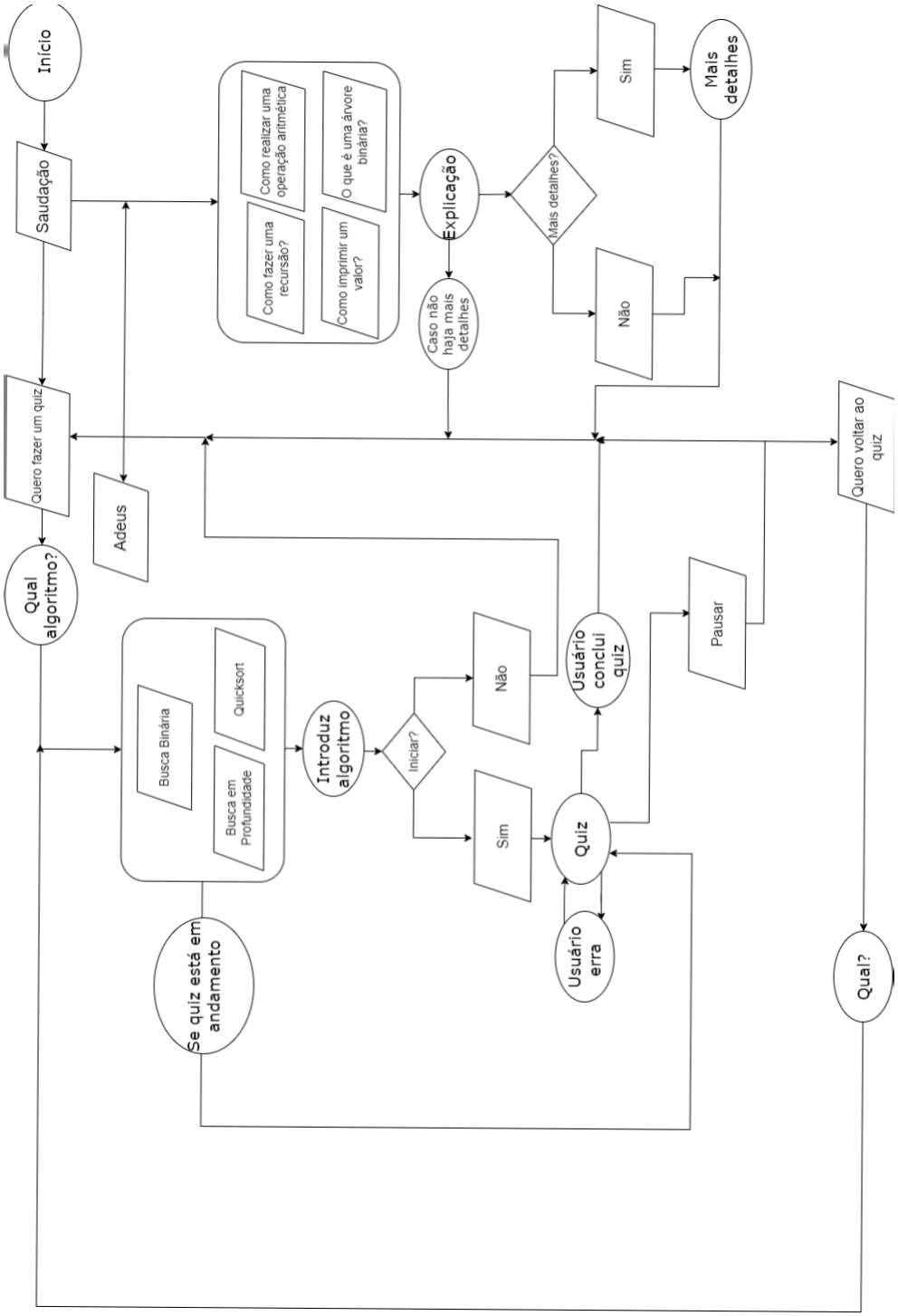

Figura 7 – Fluxo de Conversação do C-BOT. Fonte: Própria.

5 RESULTADOS OBTIDOS

O framework Rasa conta com funcionalidades de conexão com diversos canais de texto, entre eles estão: o Telegram, o Slack, o Messenger e também interfaces de chat construídas em HTML. Com o intuito de testar o uso do C-BOT por alunos de programação, optou-se por disponibilizar o chatterbot através do Messenger levando em consideração a natureza acessível dele, em navegadores ou aplicativos de celular.

A integração do chatterbot com algum meio na internet se dá através de um servidor aberto em uma máquina local que faz a comunicação entre o sistema online e o chatterbot[1]. Rasa empenha um servidor padrão na url: `http://localhost:5005/webhooks/rest/webhook`. Nota-se que para que este canal seja encontrado por um canal externo de comunicação, foi utilizado um aplicativo terceiro, denominado ngrok[2]. O ngrok, em sua forma gratuita, permite a abertura do servidor por apenas duas horas em cada execução, sendo necessário executar a aplicação novamente no fim de cada prazo.A conexão do chatterbot com o destino escolhido é estável e funciona abertamente. Foi registrada uma página respectiva ao chatterbot na rede social Facebook `https://www.facebook.com/C-BOT-113156641388618` , o que permite a comunicação da página com usuários através do aplicativo Messenger. Abaixo alguns testes realizados e os resultados obtidos com o uso do C-BOT.

Com base no fluxograma já apresentado, os exemplos de execução a seguir podem ser atribuidos em algumas partes 8, a ação em laranja indica a saudação do chatterbot e a introdução de seus detalhes, o fluxo de conversação em vermelho define a funcionalidade de explicações para com o conceito de recursão, o fluxo em azul define a funcionalidade de quiz para com o Algoritmo de Busca Binária, o fluxo em verde ilustra o caso do usuário errar uma resposta do quiz, destaca-se que o fluxo em azul-claro representa o caso que o usuário não confirma o começo do quiz, o fluxo em rosa representa a pausa do quiz em qualquer momento bem como a sua retomada, e o flxuo em amarelo indica a despedida do chatterbot.

A Figura 9 exibe uma mensagem de saudação do chatterbot em resposta à sauda-

[1] https://rasa.com/docs/rasa/messaging-and-voice-channels
[2] https://ngrok.com/

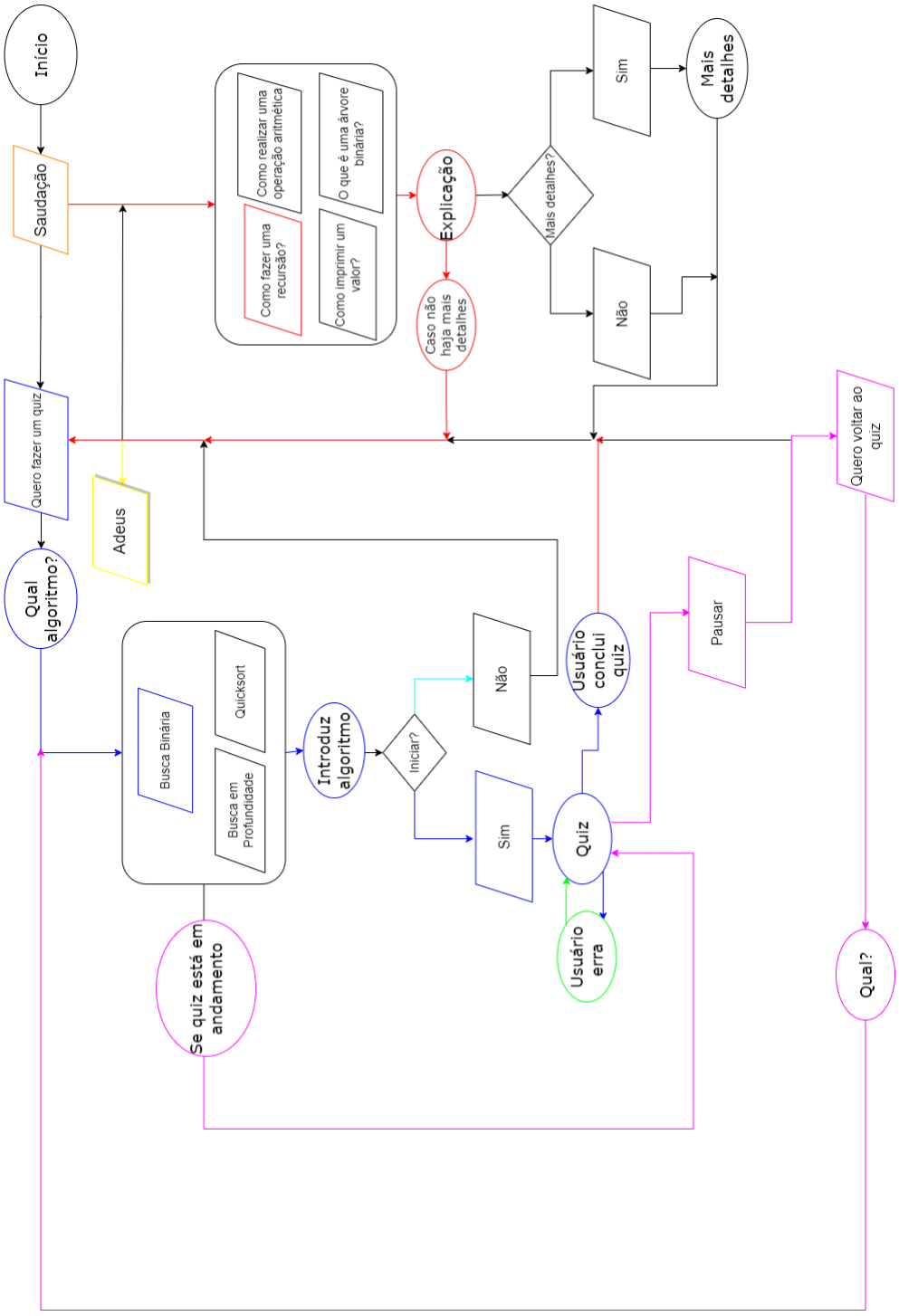

Figura 8 – Exemplos em fluxo de Conversação do C-BOT. Fonte: Própria.

ção do usuário, observa-se que após a introdução, o chatterbot pergunta se o usuário quer detalhes sobre o propósito de C-BOT. Caso a resposta seja afirmativa, detalhes sobre o chatterbot são apresentadas. A Figura 10 ilustra o caso em que a resposta do usuário seja negativa.

Conforme a Figura 11, após o usuário perguntar “Como realizar uma recursão?”, o chatterbot dá uma explicação sobre recursão em linguagem natural ao usuário (EPP, 2019). Após, um trecho de código ilustrando uma chamada recursiva também é im-

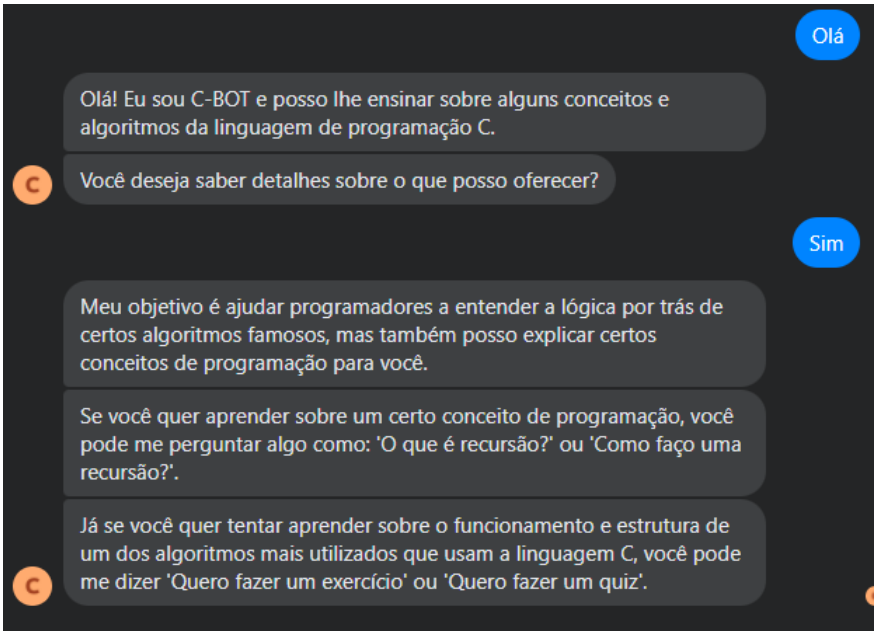

Figura 9 – Saudação do C-BOT e detalhes sobre o chatterbot. Fonte: Própria.

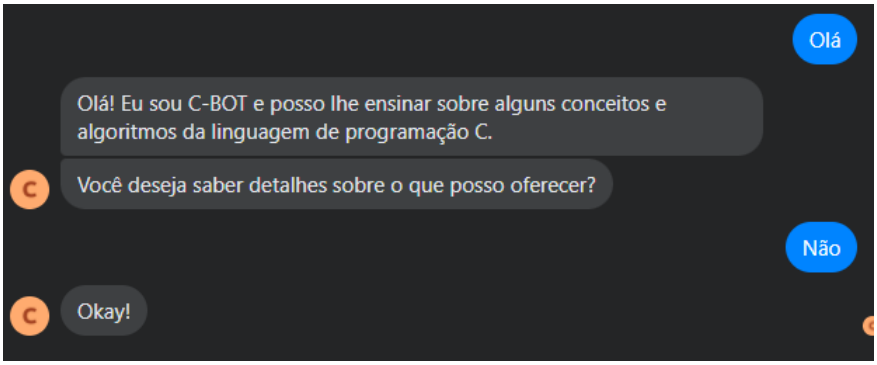

Figura 10 – Usuário não mostra interesse em saber detalhes sobre o chatterbot. Fonte: Própria.

pressa. No trecho de código, uma função que realiza um fatorial de forma recursiva é mostrado. Um fatorial de um número inteiro positivo ‘n’ consiste em uma série de produtos para todos números inteiros positivos menores que ‘n’ (A. KUHAIL, 2021).

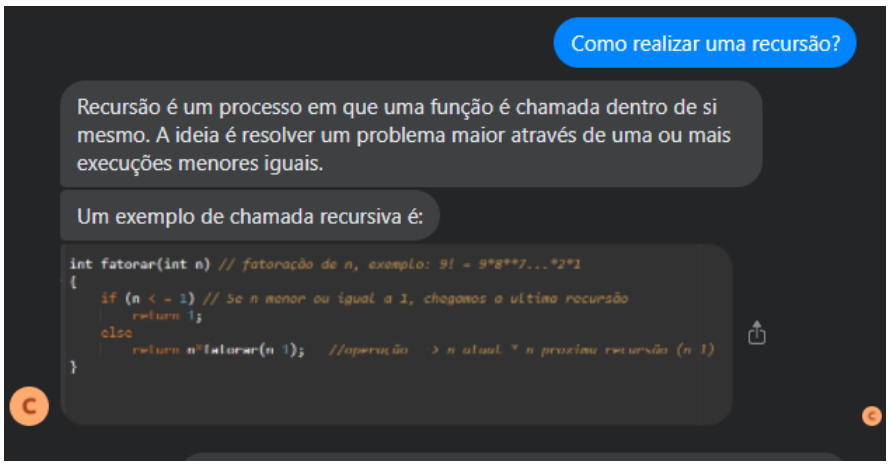

Figura 11 – Explicação sobre recursão. Fonte: Própria.

As Figuras 12, 13 e 14 exemplificam o quiz sobre o algoritmo de Busca Binária. Na Figura 12, após o pedido pela realização de um quiz, o chatterbot fornece ao usuário as opções de algoritmos. Neste caso, o usuário escolheu a Busca Binária. Após esta escolha, uma breve explicação é emitida.Na Figura 13, o chatterbot continua a explicação sobre Pequisa Binária e apresenta o código-fonte com as linhas de relevância para o quiz e com partes de trechos de código ocultos. Na Figura 14, o chatterbot pede uma confirmação por parte usuário para começar o quiz, neste exemplo, o usuário não deseja começar o quiz e a mensagem 'Okay!' é apresentada.

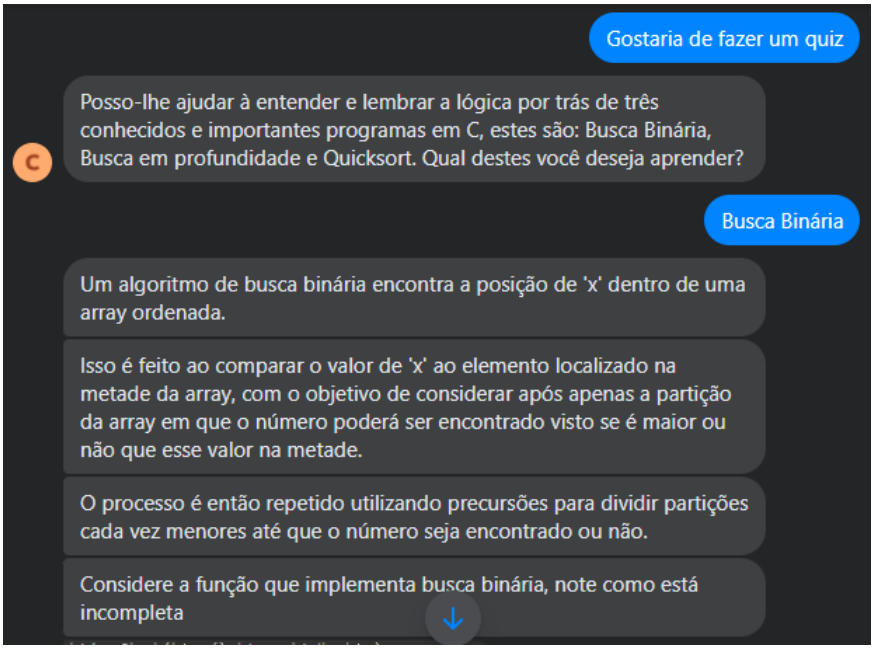

Figura 12 – Quiz sobre Busca Binária. Fonte: Própria.

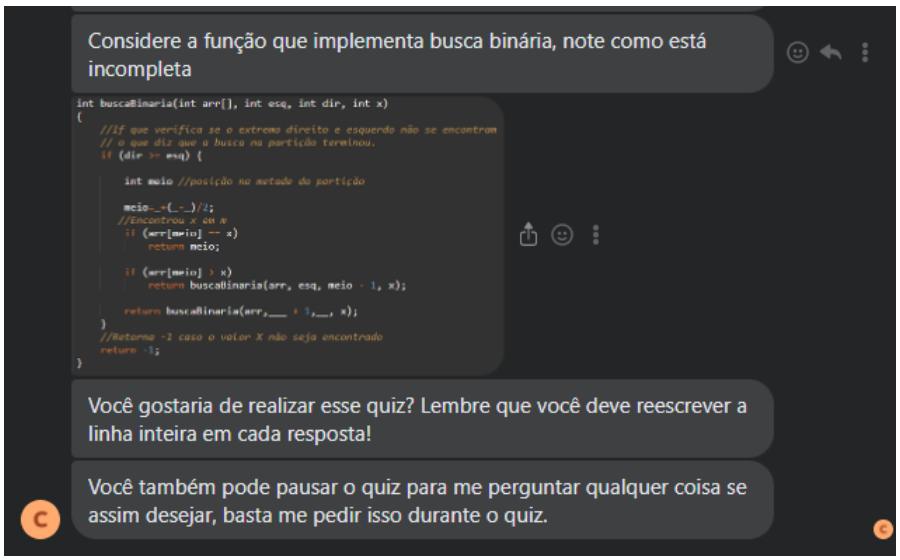

Figura 13 – Continuação do quiz sobre Busca Binária. Fonte: Própria.

Na Figura 15, o usuário confirma a realização do quiz, uma mensagem e uma imagem são apresentadas. A lógica por trás da imagem é descrita pelo chatterbot em

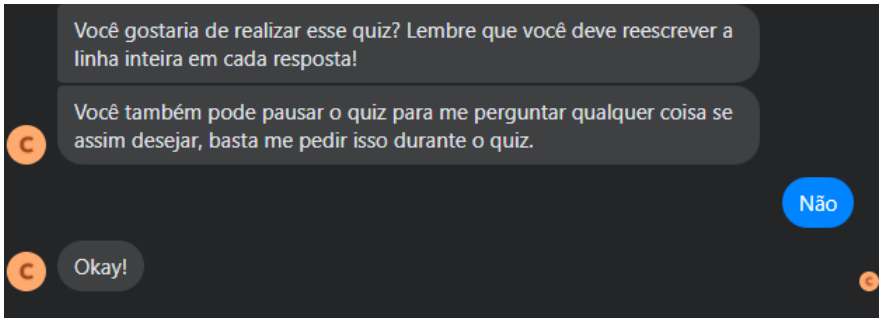

Figura 14 – Usuário nega o começo do quiz. Fonte: Própria.

texto, dado a sequência de valores 2, 5, 9, 10, 12, 15 e 20, busca-se encontrar o valor 20 e se considera a metade superior da *array*, 12, 15 e 20. Já, a imagem ilustra uma *array* processada pelo algoritmo de Busca Binária.

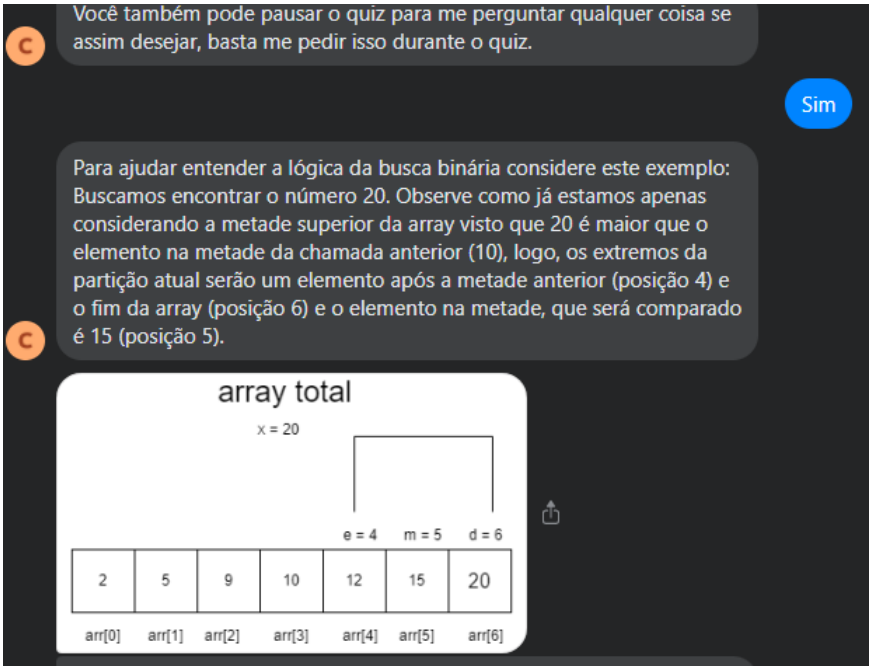

Figura 15 – Usuário confirma o começo do quiz. Fonte: Própria.

Na Figura 16, o chatterbot explica que o primeiro passo a ser realizado no código-fonte fornecido é encontrar a posição na metade entre a partição da *array* descrita anteriormente. Deve-se fazer um cálculo utilizando uma fórmula contendo as posições já conhecidas, ou seja, o começo e fim da partição. A imagem ilustra o uso das posições em cálculo, resultando na fórmula “e+(d-e)/2”, sendo ‘e’ o limite inferior da partição e ‘d’ o limite superior da partição, logo, espera-se que o usuário insira a linha de código que processa essa operação aritmética corretamente, levando em conta seus conhecimentos sobre programação e as partes de trechos de código não ocultos.

A Figura 17 mostra que após a linha de código ser inserida corretamente, o chatterbot parabeniza o usuário e o quiz entra em sua última etapa.

A Figura 18 apresenta a próxima etapa, onde o chatterbot pede para o usuário escrever os novos extremos da partição dentro da chamada recursiva. Assim como

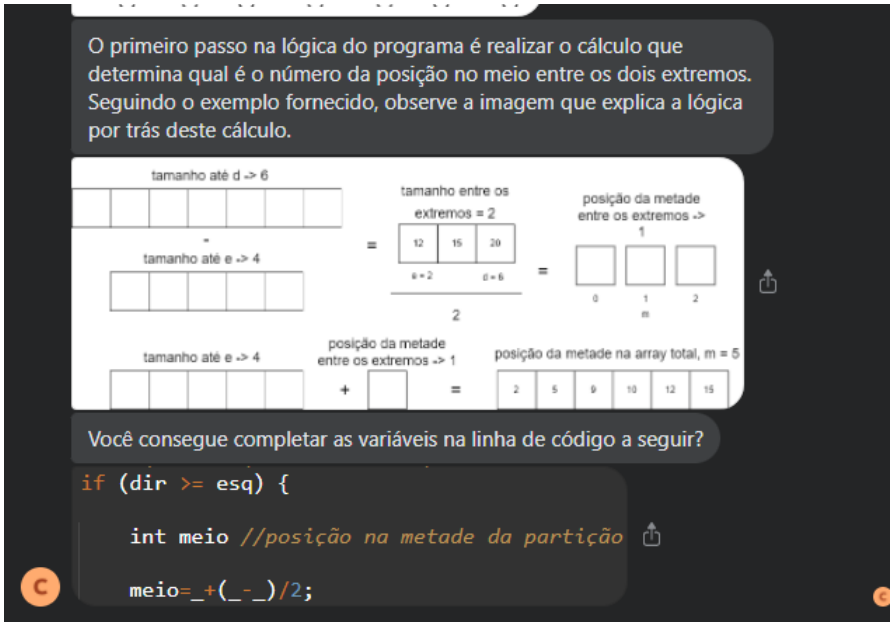

Figura 16 – Explicação da lógica do cálculo e parte de trecho de código oculto. Fonte: Própria.

na Figura anterior, caso a linha de código seja inserida corretamente, como na figura 19, o usuário recebe uma parabenização final indicando o fim do quiz.

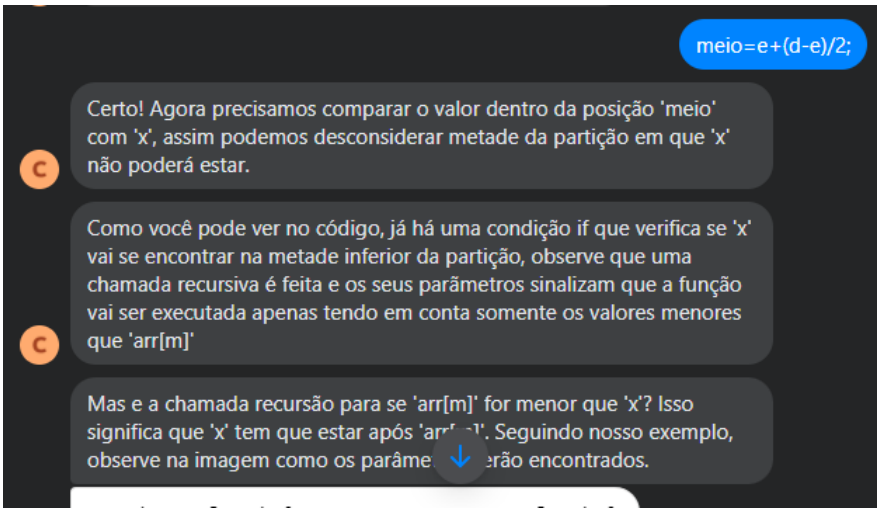

Figura 17 – Usuário acertou parte de trecho de código oculto. Fonte: Própria.

Na Figura 20, o usuário pede para pausar o quiz, assim, o chatterbot pausa o quiz e lembra o usuário que ele pode ser retomado a partir de uma requisição. Ela também mostra como o usuário retoma o quiz.

A Figura 21 ilustra o caso em que o usuário envia um parte trecho de código incorreta em qualquer etapa do quiz, o chatterbot pede para que o usuário tente novamente, e a requisição a resposta é emitida.

Outras possibilidades de interação ilustradas são o caso em que o usuário pergunta sobre algum conceito que ainda não se encontra no escopo do chatterbot, como na Figura 22, o chatterbot informa que ainda não pode responder a isso. Na Figura 23, uma mensagem de adeus é apresentada, respondido de forma simples pelo chatterbot.

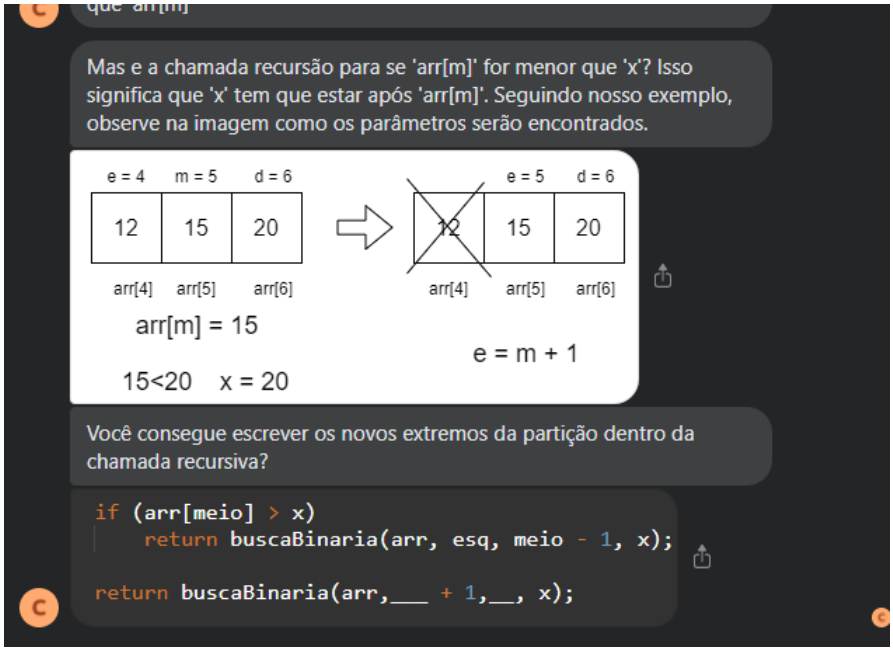

Figura 18 – Explicação dos novos extremos da partição dentro de uma chamada recursiva e parte de trecho de código oculto. Fonte: Própria.

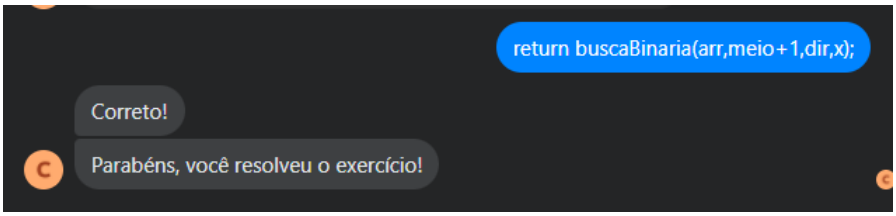

Figura 19 – Usuário acerta última linha de código e encerra o quiz. Fonte: Própria.

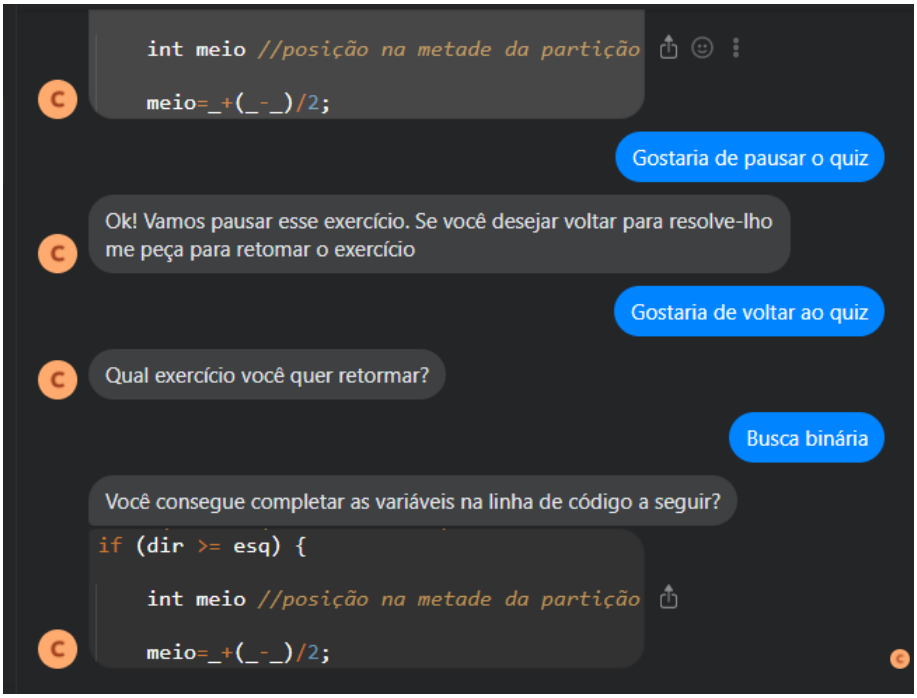

Figura 20 – Usuário pausa e retoma o quiz. Fonte: Própria.

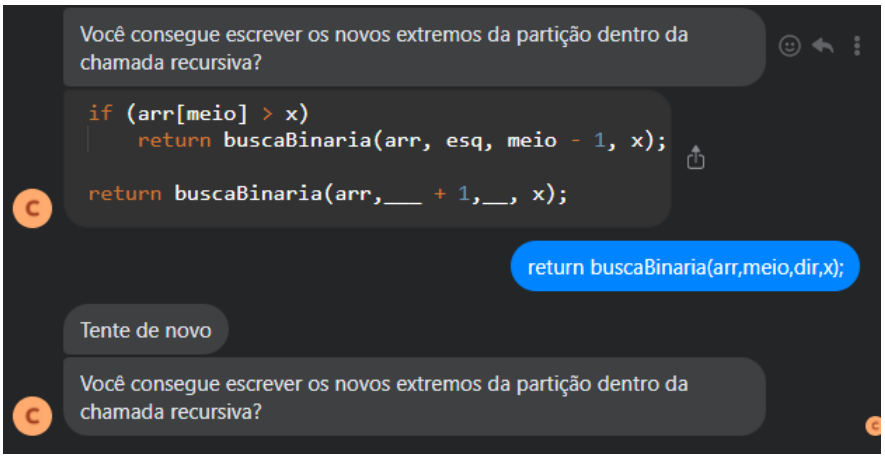

Figura 21 – Usuário errou parte de trecho de código oculto. Fonte: Própria.

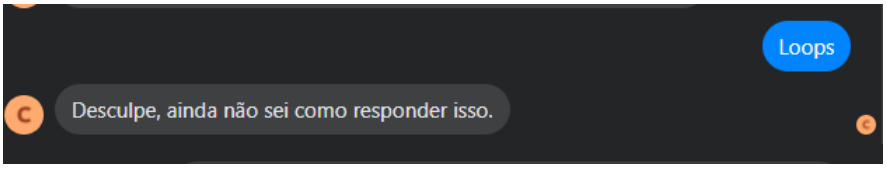

Figura 22 – Usuário pergunta sobre conhecimento não suportado pelo chatterbot. Fonte: Própria.

Figura 23 – Despedida do C-BOT. Fonte: Própria.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Neste trabalho, um chatterbot para o ensino de programação foi desenvolvido, o C-BOT. Assim como diversos chatterbots utilitários, o fluxo conversacional de C-BOT pode ser incrementado com novas funcionalidades que buscam auxiliar o ensino da programação, tais como: ordenar trechos de código de um algoritmo para que este em teoria funcione corretamente; respostas à perguntas mais específicas referentes à um certo conteúdo geral, customizando o aprendizado de cada conteúdo a partir das necessidades do aluno; e a adaptação do ensino ao registrar que dificuldades o usuário tem em um quiz, sugerindo explicações para conceitos relevantes as dificuldades.

Ainda considera-se tornar o chatterbot mais amigável, através da integração de novas intenções, como por exemplo uma intenção respectiva em relação a perguntas casuais como “Como você está?” e a resposta apropriada.

Outra possibilidade de aprimoramento na interatividade é a integração do chatterbot com outras plataformas de comunicação, em sua versão atual, C-BOT foi testado no aplicativo de comunicação Messenger com um limite de tempo curto de duas horas. Uma possibilidade estudada é a aplicação do chatterbot em uma página web com hospedagem constante, tal conexão também é suportada pela framework Rasa[1].

Por fim, como trabalhos futuros pretendemos aplicar o C-BOT em um contexto educacional. Dado o escopo e metodologia de ensino do chatterbot, este é potencialmente aplicável em diversas disciplinas de contexto de programação, para disciplinas para o curso de Ciência e Engenharia de Computação dentro da Universidade Federal de Pelotas (UFPEL), este é aplicável em cadeiras que abordam desde fundamentos básicos de programação como a utilização de funções e recursões em disciplinas como Algoritmos e Programação (AEP) ou para a memorização do funcionamento de algoritmos relevantes como o algoritmo de Busca em Profundidade, geralmente ensinado em uma disciplina como Algoritmos e Estruturas de Dados (AED) e Fundamentos de Inteligência Artifical (FIA). A aplicação do chatterbot em classe é essencial para o refinamento da metodologia de ensino do chatterbot. Sugere-se a formulação e aplicação de um questionário de satisfação no uso do C-BOT em

[1] https://rasa.com/docs/rasa/connectors/your-own-website

uma turma de programação, avaliação feitas também pelos chatterbots desenvolvidos nos trabalhos relacionados (JOHNSON; E. ZANOLI, 2021), (A. ADE-IBIJOLA, 2021) e (HOBERT, 2019). A implementação de C-BOT está disponível no repositório GitHub `https://github.com/jessoares/Chatbot`.

REFERÊNCIAS

A. ADE-IBIJOLA, C. W. O. e. Python-Bot: A Chatbot for Teaching Python Programming. **Engineering Letters**, South Africa, p.1–6, 2021.

A. KUHAIL, M. N. e. J. S. Teaching Recursive Thinking using Unplugged Activities. **Jornal desconhecido**, Abu Dhabi, p.2–3, 2021.

A. PEARS S. SEIDMAN, L. M. e. a. A survey of literature on the teaching of introductory programming. **ACM SIGCSE**, USA, p.1–2, 2009.

A. SACCARO, P. A. J. e. a. Fatores Associados à Evasão no Ensino Superior Brasileiro: um estudo de análise de sobrevivência para os cursos das áreas de Ciência, Matemática e Computação e de Engenharia, Produção e Construção em instituições públicas e privadas. **Estudos Econômicos (São Paulo)**, Brasil, p.1–3, 2019.

A. VASWANI, N. S. e. a. Attention Is All You Need. **Cornell University**, USA, NY, p.1–15, 2017.

AABY, A. **Introduction to Programming Languages**. USA: World Colleges Information, 1996.

ASTRACHAN, O. Bubble sort: an archaeological algorithmic analysis. **ACM SIGCSE Bulletin**, USA, p.1–5, 2003.

B. J. KUN, S. R. e. Quizbot: A dialogue-based adaptive learning system for factual knowledge. **CHI Conference on Human Factors in Computing Systems**, Brasil, p.2–3, 2019.

BENNEDSEN, J. **Teaching and learning introductory programming**: a model-based approach. Denmark: Editora desconhecida, 2008.

CHAO, P. Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. **Computers and Education 95**, Taiwan, p.1–2, 2020.

EPP, S. **Discrete Mathematics with Applications**. USA: ISBN, 2019.

F. CLARIZIA F. COLACE, M. L. e. a. An education support system for student. In International Symposium on Cyberspace Safety and Security. **Cyberspace Safety and Security**, Italy, p.2–3, 2018.

GARNIER, R. **Discrete Mathematics**:Proofs, Structures and Applications. USA: CRC Press, 2009.

GLASS, B. **Beyond point and click**: The expanding demand for coding skills. 1–7p. Disponível em: <https://academy.oracle.com/pages/Beyond_Point_Click_final.pdf>. Acesso em: 2021-09-03.

HOARE, C. A. R. Quicksort. **The Computer Journal**, USA, p.1–8, 1962.

HOBERT, S. Say Hello to 'Coding Tutor'! Design and Evaluation of a Chatbot-based Learning System Supporting Students to Learn to Program. **Fortieth International Conference on Information Systems**, Germany, p.1–7, 2019.

HUBERT, R. **6 Ways Artificial Intelligence and Chatbots are Changing Education**.

IMAGING, E. F. **ABC's of VDP. A Variable Data Printing basics Guide**.

JENKINS, T. **On the Difficulty of Learning to Program**. Massachusetts: Computers and Education, 2002.

JOHNSON, C.; E. ZANOLI, M. U. e. **Escapeling**: A Gamified, AI-Supported Chatbot for Collaborative Language Practice.

JUNG, S. Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. **Computer Speech and Language**, Korea, p.1–4, 2019.

K. RAMESH S. RAVISHANKARAN, A. J. e. a. A Survey of Design Techniques for Conversational Agents. **Communications in Computer and Information Science**, India, p.1–2, 2017.

KNUTH, D. **The Art of Computer Programming**. USA: American Scientist, 1997.

L. MOUSSIADES, E. A. e. An Overview of Chatbot Technology. **Artificial Intelligence Applications and Innovations**, Greece, p.1–3, 2020.

L. RONALD, H. T. H. e. **Introduction to Algorithms**. USA: The MIT Press, 2009.

L. XUAN T. PHAM, e. a. Chatbot as an Intelligent Personal Assistant for Mobile Language Learning. **ICEEL 2018**, Vietnam, p.1–9, 2018.

M. SOLLNER, R. W. e. Unleashing the Potential of Chatbots in Education: A State-Of-The-Art Analysis. **Academy of Management Annual Meeting Proceedings**, Germany, p.2–3, 2018.

M. SU, C. W. e. H. W. A chatbot using lstm-based multi-layer embedding for elderly care. **2017 International Conference on Orange Technologies**, Singapore, p.1–2, 2020.

N. ASHILL, S. E. e. The determinants of students' perceived learning outcomes and satisfaction in university online education: An update. **Decision Sciences Journal of Innovative Education**, USA, p.3–5, 2016.

O. KNILL J. CARLSSON, A. C. e. M. L. An artificial intelligence experiment in college math education. **Jornal desconhecido**, USA, p.1–2, 2004.

PFAFF, B. **An Introduction to Binary Search Trees and Balanced Trees**. USA: Free Software Foundation, Inc., 2004.

R. CARPENTER, L. F. e. Bots as language learning tools. **Language, Learning and Technology**, Hong Kong, p.1–4, 2006.

S. BASAK, S. S. e. An educational chatbot for answering queries. In Emerging Technology in Modelling and Graphics. **Springer**, Kolkata, India, p.1–5, 2020.

S. H. M. DAUD, N. H. I. T. e. N. H. M. Z. e-Java Chatbot for Learning Programming Language: A Post-Pandemic Alternative Virtual Tutor. **International Journal of Emerging Trends in Engineering Research**, Malaysia, p.1–6, 2020.

S. L. LAU, T. Y. S. e. Online Tools to Support Novice Programming: A Systematic Review. **IEEE Conference on e-Learning, e-Management and e-Services (IC3e)**, Malaysia, p.2–3, 2018.

SCHMIDT, D. A. **The structure of typed programming languages**. USA: MIT Press, 1994.

SEBESTA, R. W. **Concepts of Programming Languages**. USA: Addison-Wesley, 2009.

SHAFFER, C. A. **Structures and Algorithm Analysis in C++**. USA: Editora desconhecida, 2011.

STROUSTRUP, B. **The C++ Programming Language**. Germany: Addison-Wesley, 2013.

T. BOCKLISCH J. FAULKNER, N. P. e. a. Rasa: Open Source Language Understanding and Dialogue Management. **NIPS 2017 Conversational AI workshop**, USA, p.1–8, 2017.

T. CROW, A. L.-R. e. B. W. Intelligent tutoring systems for programming education: a systematic review. **Australasian Computing Education Conference**, Australia, p.1–3, 2018.

T. DARADOUMIS J. M. M. PUIG, M. A. e. a. **Analyzing students perceptions to improve the design of an automated assessment tool in online distributed programming**. Spain: Computers and Education, 2016.

WEGNER, P. **Encyclopedia of Computer Science**. USA: John Wiley and Sons, 2003.

WEISSTEIN, E. **Binary search**. USA: MathWorld, 2021.

WOLFF, S. H. e R. M. von. Say Hello to Your New Automated Tutor – A Structured Literature Review on Pedagogical Conversational Agents. **14th International Conference on Wirtschafts informatik**, Germany, p.1–7, 2019.