OER-Chat: An Open Chatbot to Support the Reuse of Open Educational Resources of Introductory Programming

William Simão de Deus
University of São Paulo
williamsimao@usp.br

Ellen Francine Barbosa
University of São Paulo
francine@icmc.usp.br

Abstract

This study presents OER-Chat, a chatbot designed for introductory programming. OER-Chat adopts Open Educational Resources (OER) instead of providing textual responses, which is common among other chatbots. Essentially, OER-Chat offers educational materials for introductory programming to address users' doubts or questions, such as slides, courses, or open textbooks. By doing so, it promotes the reuse of these materials and fosters Open Education. We evaluated OER-Chat using a qualitative paradigm. Obtained results demonstrated the efficiency of OER-Chat for the participants and identified areas for future improvement to enhance its performance.

1. Introduction

Chatbots are a type of chatting robot, simulating conversations between humans and a computer program (Dahiya, 2017). In essence, they are applications designed to engage in conversations resembling human interaction by using specific algorithms to provide a conversational interface.

ChatGPT[1] is an example of chatbot. It provides an interface between Artificial Intelligence (AI) and human beings. ChatGPT represents the advance of AI applications, drawing public attention across the globe (Tlili et al., 2023). However, there are ongoing discussions regarding the use of AI. For instance, it is common to use closed or proprietary materials (e.g., books, images, videos, or web articles) to train data models. In many cases, the authors of these materials are not remunerated for their work, raising ethical concerns regarding their use (Ray, 2023).

On the other hand, introductory programming is a common field of investigation by using AI and chatbots initiatives (Carreira et al., 2022; Okonkwo & Ade-Ibijola, 2022; Verleger & Pembridge, 2018). However, these chatbots are primarily focused on the students' perspective. They are often used as online tutors, answering questions or checking code snippets to support beginners in programming. As a result, the perspective of content producers for education, such as teachers, is often overlooked and not the primary focus of these chatbots.

Considering this scenario, our objective in this study is to present an OER-Chat, a chatbot designed to support the reuse of Open Educational Resources (OER) for introductory programming. OER are learning, teaching and research materials in any format and medium that reside in the public domain or are under copyright that have been released under an open license, that permit no-cost access, re-use, re-purpose, adaptation and redistribution by others (UNESCO, 2019). In short, OER-Chat provides OER (e.g., such as slides, courses, open textbooks, etc...) to address users' doubts or questions about programming. By doing so, it promotes the reuse of these materials and fosters Open Education. In this sense, we elaborated the following Research Question (RQ): How do participants respond to using the OER-Chat? For solving it, we adopted a qualitative research paradigm. Our results demonstrated the efficiency of OER-Chatbot in essential aspects, including User experience, Usability, and Adherence of Results.

The remainder of this study is organized as follows: Section 2 presents the background. Section 3 introduces the research design. Section 4 summarizes the evaluation process. Section 5 concentrates results. Final remarks are presented in Section 6.

2. Background

As mentioned, chatbots serve as an interface to simulate conversations between human beings and computer applications (Dahiya, 2017). For example, if a user writes the following command: "What is Object-Oriented Programming?", the chatbot must generate a response.

Following this assumption, many authors have been

[1] https://openai.com/blog/chatgpt

URI: https://hdl.handle.net/10125/107008
978-0-9981331-7-1

researching how chatbots can support the teaching and learning of introductory programming. For instance, Verleger and Pembridge (2018) introduced the "EduBot"; Carreira et al. (2022) proposed the "Pyo"; and, Okonkwo and Ade-Ibijola (2022) designed the "Revision-Bot". These are examples of chatbots for beginners in programming. However, they have two similar characteristics: (1) they were designed for students and (2) they were developed to solve basic doubts about programming.

In this sense, the main difference between our chatbot (OER-Chat) and others is related to its educational purpose. OER-Chat is specifically designed to support collaborative education by providing OER in the context of introductory programming. OER were described by (UNESCO, 2019) as educational materials, open licensed, free of charge and developed to be used and reused by users. Thus, when our chatbot generates a result, it presents a list of OER (e.g., slides, videos, open textbooks, among others) for introductory programming. The aim is to promote open education by offering users free and open materials instead of textual responses. We also intend to facilitate the dissemination of open education by presenting a similarity index between the OER and the user command. Thus, the OER-Chat responses exclusively for introductory programming area, sparing users from unrelated content like computer networking or database programming. This saves time, effort, and maintains user concentration.

Herrera et al. (2019) proposed a chatbot for an open and free online course provider. Holotescu (2016) introduced the "MOOCBuddy", an open chatbot designed to act as a recommender system for users. Our study differs from these because our chatbot was specifically designed to support the introductory programming, whereas both works were designed for a general context. Thus, OER-Chat encourages users to engage with programming, facilitating contributions from teachers and students to these OER.

Identifying an OER for introductory programming is a complex task for any computer system. Firstly, introductory programming covers a course for beginners students during the first year of programming education, which includes various subjects, such as problem-solving, programming concepts, syntax, semantics, among others (Medeiros et al., 2019). In addition, many terms are overlapped in the Computer Science area, as occurs with "bug", "string", "tree", etc. (Roesler, 2021). As a consequence, OER do not follow a terminological standard and adopt different terms to represent the same subject, such as *"loop", "iteration"* or *"repeat"*. To the best of our knowledge, there is no strategy capable of reducing these challenges in OER for introductory programming. These problems together prevent the identification of OER. As UNESCO (2019) warned: it is essential to act to reduce barriers to create, access, re-use, adapt and redistribute OER.

In addition, an efficient open chatbot also needs to overcome a barrier considering the OER context: diversity. OER are available using various digital formats, such as textbooks, syllabus, texts, courses, presentations, among others (Deus & Barbosa, 2020). As a consequence, each resource has an own logical structure. To deal with this barrier, OER-Chat classifies OER into two groups: (1) web files, which are resources in the .html extension, which allow access to the OER content through valid URL; (2) digital files, which are identified by metadata. In this case, the chatbot indicates a digital file to the user which can be downloaded.

3. OER-Chat Architecture

OER-Chat is composed by six distinct modules. They are integrated, as presented in Figure 1. Each module was designed considering the aforementioned challenges, being described next.

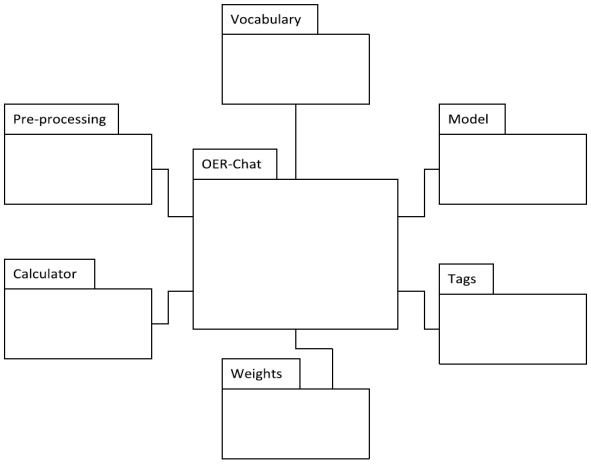

Figure 1. OER-Chat - Architecture

1. Pre-processing module: Pre-processing is the first module and it was inspired by Roesler (2021). In short, this module is responsible for using Natural Language Processing to understand commands send by users and to identify best results. Briefly, this module processes instructions and generates a matrix of data. Six actions are performed, as detailed in Table 1. In addition, Figure 2 illustrated how the preprocessing module works.

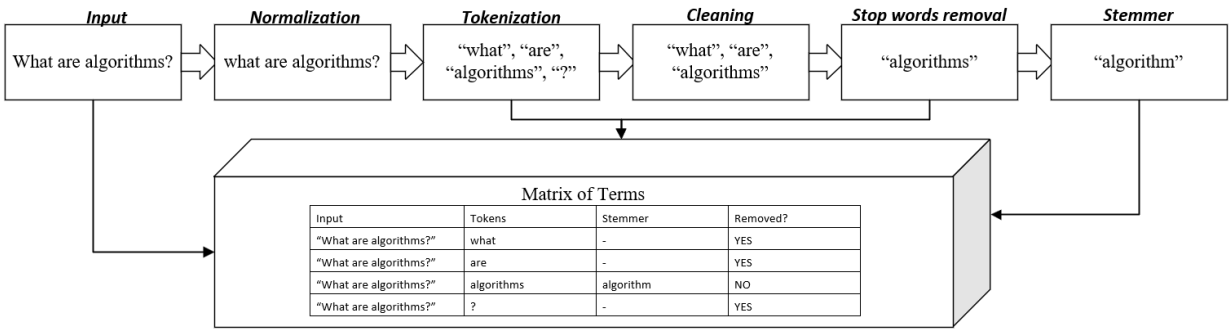

Figure 2. An example of the pre-processing module

Table 1. The pre-processing steps

Step	Term
Normalization	Conversion to lower case
Tokenization	Split words according to blank space and/or hyphen;
Cleaning	Special characters (e.g., symbols, accents, punctuation and numbers) are removed;
Stop words	Ordinary terms, like “the”, “a”, “an”, among others, are removed;
Stemmer	each token is converted to stemmer format; and,
Matrix	generation of a matrix of data.

2. Vocabulary module: The vocabulary is the second module. It works as an initial framework to: (1) identify whether the user's dialogue is within the scope of teaching and learning of introductory programming; (2) verify potential OERs; and, (3) identify synonyms for users' input.

In order to create the vocabulary, we selected an OER database of programming education provided by Deus and Barbosa (2020). This database encompasses 3990 OER of programming. We manually revised each resource, removing invalid entries (i.e., offline resources, OER with bugs, or OER with advanced programming concepts). After, each OER was processed by our *pre-processing module*, generating the matrix of terms. Finally, the matrix was filtered by a software following three criteria:

- *Academic purpose:* many terms were unrelated to the academic purpose. For instance, *“becoming”* or *“seems”*. At the same time, terms such as *“loops”* or *“recursion”* were very important to the vocabulary. Thus, we calculated the frequency of each term using the British National Corpus (BNC)[2] To select relevant terms, non-academic terms from the BNC corpus that recurred with great frequency in OER ($> 150\%$) were removed, as suggested by Lei and Liu (2016) and Roesler (2021).

- *Minimum Frequency:* relevant terms are used several times, and irrelevant terms have an inconstant use. To identify them, we adopted the measure provided by Roesler (2021): 0.28 occurrences per 10.000 words.

- *Technical meaning:* a technical dictionary was also adopted to identify technical terms, as proposed by Roesler (2021) and Lei and Liu (2016). For this purpose, we used the Oxford Dictionary of Computer Science[3]. In short, a valid term for our vocabulary must appear in this dictionary. We performed a manual verification to check each selected word.

After concluding the selection of terms, 163 terms were identified. These terms are represented in Table 2. Also, they are integrated into a new module, presented next.

3. Introductory Programming module: The third module is dedicated to Introductory Programming. It was developed to create a match between the user command and OER available using the vocabulary. In this sense, we classified manually each term in the vocabulary module. As a result, four areas emerged:

- **Foundations:** this area presents basic concepts of programming, such as the concept of variables, bugs, compilers, etc. The main idea is to introduce essential programming aspects and concepts, and not to code.

- **Techniques:** this area summarizes more advanced concepts of introductory programming. In this case, the user is interested in writing initial codes. For example, using a matrix, an array or a vector.

[2]The BNC corpus has more than 100 million terms, and it is divided into different areas, being one of the most complete in the English language.

[3]https://www.oxfordreference.com/

Table 2. The Vocabulary and The Introductory Programming Model

Area	Term
Foundations	algorithm, alias, ansi, archive, ascii, attribute, b, bug, byte, compiler, computer, console, cpu, cs, desktop, developer, documentation, encoding, execute, flowchart, hardware, ide, implementation, informatics, information, interpreter, language, learning, lisp, logical, merging, mouse, pc, precedence, primitive, processor, programmer, programming, ram, semaphore, static, terminal, unicode.
Techniques	api, app, application, array, balanced, buffer, coding, comparator, compatibility, concatenation, conditional, debugging, declaration, decoding, decomposition, default, definition, delete, dictionary, directive, directory, domain, dynamic, entity, exception, expression, failure, garbage, header, heap, heapsort, hierarchy, identifier, increment, initialization, insert, instance, instantiation, instruction, integer, io, is, iteration, leaf, library, listing, location, loop, matrix, memory, min, namespace, node, obj, operand, operator, overflow, parameter, pointer, precision, procedure, pseudocode, queue, quicksort, recursion, recursive, refresh, register, routine, script, scripting, sequential, signature, sorting, stack, structure, subroutine, subset, synchronous, testing, unsigned, var, variable, vector.
Languages	c, haskell, interface, java, javascript, jdk, pascal, php, python, ruby, syntax.
Paradigms	abstraction, aspect, encapsulation, generic, inheritance, oop, overloading, paradigm, polymorphic, polymorphism.

- **Languages:** In this area the user is looking for some resource based on a programming language. At this level, the commands are related to specific problems of a programming language, such as Java, C or Python.
- **Paradigms:** this area presents the most sophisticated approach to the user. When he/she is looking for an OER to solve a problem using Object-Oriented Programming.

These four areas are connected to each other. For example, if a user sends the following command: *How to do inheritance in Java?* He/she is looking for OER aimed at teaching a specific programming language (Java) and a specific paradigm (Object-Oriented). Thus, the vocabulary is adopted to define the scope while the model is used to identify similar occurrences located at the same intersection (paradigms and languages). In Figure 3 we summarize this procedure.

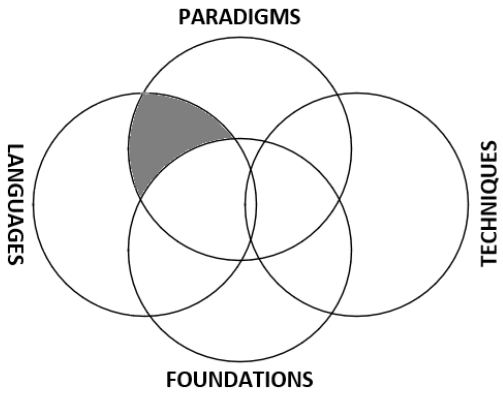

Figure 3. The gray area shows the intersection between Paradigms and Languages.

4. Tags module: The fourth module is based on the use of OER Tags. According to Kabir et al. (2015), we can isolate informative blocks from large web pages by using tags. OER-Chat has two tags categories: (1) HTML tags, which are used to structure OER as web pages, showing title, subtitles, paragraphs, images, links, among other features; (2) metadata, which includes descriptive information for each resource, such as the brief description, keywords, authors' names, and more.

HTML tag are used in websites and Metadata are adopted in digital files (such as videos, slides, audios, among others). Thus, OER-Chat recommends both websites and digital files to users, identifying if the HTML tags are available or just the metadata.

5. Weights module: The fifth module is designed to define a set of weights to identify the best result and address distortions present among OER, inspired by Kabir et al. (2015). For example, if one OER has the term "variable" in its title while another OER has the term "variable" in its description, the contexts in which these terms are used are not equivalent. The title serves as a unique identifier, both in HTML tags and metadata keys, while the description can appear multiple times, utilizing different HTML tags and metadata. Thus, this module checks term occurrences, their origins, and assigns a weight to each term.

The module is updated when new OER are added, ensuring that all resources are also updated. This module assigns a higher weight to OER that have terms in unique identification fields, such as the title. It also assigns weights to terms present in the vocabulary and terms that exist within certain areas or intersections of the model. In the current version, the module assigns weights according to HTML tags and metadata.

Thus, weights are computed automatically for HTML tags, with $H1$ and $title$ tags receiving a weight of 1; P tag receives a weight of 0.5, and the other tags receive 0.1. For metadata, title receive a weight of 1; course, description, subject, relation, and abstract receive 0.5 and other metadata receive 0.1.

6. Probability Calculator module: The last module is the probability calculator. In short, this module takes into consideration the procedures performed by the other modules to determine the OER that best suits the user's input. The calculator offers three options: if no OER is found for the user input, a message is returned, suggesting that the user add new terms or rewrite the entry. If only one resource is found, that material is returned to the user. If a list of resources is found, the list is organized based on a mathematical calculation that determines the similarity index between the input and the OER.

In this sense, the probability is given by a set of equations. Initially, let P be the result of pre-processing module as follows:

$$P = [word_1, word_2, ..., word_n] \tag{1}$$

Where $word_x$ is each word processed by the first module. Next, the probability calculator checks the occurrence in the vocabulary module:

$$V_a = \sum_{k=1}^{N} P[word_x, area] \tag{2}$$

Where V_a is the sum of all identified occurrences by area ($area$). After, the probability calculator identifies the tags and weights:

$$TW = \sum_{k=1}^{N} V_a|T_w| + \sum_{k=1}^{N} V_a|M_w| \tag{3}$$

Where TW is the final result, considering the weight from HTML Tags ($|T_w|$) and metadata keys ($|M_w|$). Finally, the module performs this processing to check which OER has the highest similarity index. Sorting results are presented in ascending order.

4. Validation

Our RQ is the following: How do participants respond to using the OER-Chat? In order to solve it, we have preliminarily evaluated OER-Chat with five participants, who are teachers and/or general users interested in introductory programming. For this, we invited participants from the CAED research group. This research group investigates how information technology can be applied to facilitate teaching and learning. All are specialists in the field and have knowledge in Open Education and/or Programming Teaching.

In addition, we selected the EduCapes[4] repository. This repository has a large collection of OER for introductory programming; however, it has several problems with the OER classification system. In short, due to the large concentration of OER, it is a difficult task to identify relevant resources within EduCapes.

4.1. Instantiation

We collected OER from EduCapes repository to add to the OER-Chatbot database. To accomplish this, we generated an EduCapes dataset through a combination of a web crawler, following the assumptions from Deus and Barbosa (2020). In essence, the web crawler initiated searches using the terms found in the vocabulary and subsequently downloaded all the resulting data into a .csv file. The data is available on the following address: https://www.doi.org/10.5281/zenodo.8339188.

4.2. Design and Participants

We conducted the evaluation by adopting a qualitative paradigm (Selltiz et al., 1991). That is, our intention was to carefully select users with relevant profiles and conduct an in-depth study on their experience using OER-Chat. This proposal is due to the fact that OER-Chat is an interdisciplinary tool, which is focused on the area of information technology and education. For this reason, we invited five participants from the CAED research group research group:

P1: A general user interested in developing educational programming applications. The participant stated that he uses OER casually.

P2: A teacher of introductory programming. He has been working with introductory programming for over 10 years and has been using OER casually for the last 3 years.

P3: A developer interested in OER and open education, he has experience as a teacher of introductory programming and currently works in the software industry. The participant has more than five years of experience in introductory programming and using OER. He states that he uses OER frequently.

P4: A systems analyst with more than 10 years of experience in introductory programming and over five years of experience with OER. The participant uses OER occasionally.

P5: A teacher of introductory programming and a user interested in developing educational programming applications. He has over 10 years of experience with introductory programming and uses OER casually.

[4]https://educapes.capes.gov.br/

4.3. Procedure

We also requested approval of the research from the Brazilian Ethics Committee. Our research was approved under the process 66316222.4.0000.5390. Our research consisted of three stages: (1) presentation of the consent form and initial data collection (background, experience, etc.); (2) Use of the chatbot; (3) Collection of feedback and comments after using the chatbot.

4.3.1. Consent form and initial data collection To fulfill this purpose, we developed a digital questionnaire survey to introduce the validation process, present the consent form, and collect data from each participant. Consequently, this stage included questions regarding the participants' profiles, their experience with introductory programming, OER, and chatbots. These data were gathered to assess the profile of each participant, summarizing their key characteristics and perceptions.

4.3.2. Use of chatbot Participants accessed and used the chatbot voluntarily. The aim was to simulate the demand for OER based on educational and personal purposes. Hence, the search topic was not limited to a specific programming language or technology. Following the assumptions from Selltiz et al. (1991), participants used the tool freely to simulate a real-world research scenario. In addition, we also did not impose a time limit. Each user utilized the chatbot based on their interest. Once they felt they had formed a well-informed opinion about the chatbot, the participant stopped using it and submitted their feedback and comments to us.

4.3.3. Collecting feedbacks and comments Data were collected through open questions, which explored the user's perception of the chatbot. If any answer was not clear enough, the participant was invited to clarify the points presented to the researcher.

To achieve this purpose, we adopted the framework proposed by Selltiz et al. (1991). In summary, the questions were designed to avoid introducing personal biases. For instance, instead of asking "What challenges did you face using the chatbot?", we asked "Please describe your experience using the chatbot". In the first case, users might feel compelled to report a challenge, even if none arose. In the second case, if users encountered a challenge, it would naturally emerge in their comments.

In addition, if any question exhibited bias, an opposing question was asked immediately afterwards. For instance, if we asked "What NEGATIVE feelings describe your experience using the Chatbot?", we would then ask "Which POSITIVE feelings describe your experience when using the Chatbot?". In doing this, our intention was to neutralize the introduction of biases.

4.4. Measurements

To analyze our data, we adopted the framework proposed by Pandit (1996): Firstly, we adopted the *Open Coding*. In summary, we separated the collected data into pieces to examine them more closely. For this, all feedbacks and comments were separated into sentences and each sentence received a code for further analysis. During this process, sentences with similar codes were grouped together, generating initial concepts.

Next, we performed the *Axial Coding*. Thus, we revisited each code and sentence in order to establish a list of categories and sub-categories. For example, we noticed some feedback and comments received were about the usability of the chatbot. So we created a category called "Usability". Next, this category received a list of items that were related, such as design and simplicity.

After carrying out the entire analytical process, we derive our understanding of the results. Thus, we present the final perception of the efficiency of the OER-Chatbot in order to support the use of OER for introductory programming as well as points for future improvements.

5. How participants respond when using OER-Chat

5.1. Sentiment analysis

Sentiment analysis seeks to identify which feelings are provoked in users when using an application. This analysis is relevant for chatbots because if users experience negative emotions such as anger, frustration, or irritation, probably they will not continue using the chatbot. On the other hand, positive emotions tend to demonstrate user interest in the chatbot. For this purpose, we adopted the guidelines from Selltiz et al. (1991) to list positive and negative feelings. In short, we created two lists: one for positive sentiment and one for negative sentiment. The lists were complementary to each other, presenting a positive sentiment and its respective negative sentiment, such as 'Happiness x Sadness' and 'Easy to use x Difficult to use'. We presented these lists to the participants to identify which feelings arose during the use of the OER-Chat; Figure 4 illustrates the results achieved.

In total, we have identified 4 negative emotions (Dissatisfaction, Irritation, Disappointment (2x), and Frustration). However, we have noticed 8 positive

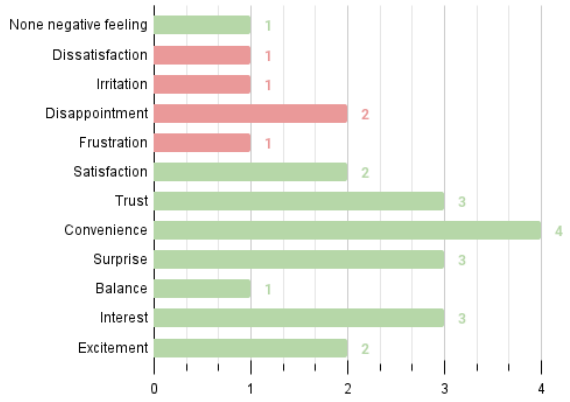

Figure 4. Positive (green) and Negative (red) feelings

emotions (None negative feeling, Satisfaction (2x), Trust (3x), Convenience (4x), Surprise (3x), Balance, Interest (3x), and Excitement (2x)). Negative feelings were not repeated, except for Disappointment, which was reported by two participants. Meanwhile, positive sentiments were repeated many times. Sometimes reaching 3 or 4 participants.

In addition, we also analyzed how these emotions emerged for each participant. In Figure 5 we present the entire list. As shown, except for participant P1, who presented only positive feelings, all other participants listed 1 or 2 positive feelings for each negative feeling. The findings demonstrate that users, for the most part, experienced more positive sensations than negative ones while using the OER-Chat.

5.2. User experience

We also identified more technical and objective elements in the feedback we received. We collected phrases from participants that described their user experience after using the chatbot. For instance, one participant (P4) stated that: *It is practical. The chat is intuitive and fast*. Another participant (P1) highlighted that: *Overall, after using the OER-Chat, I had a positive experience*. In Table 3, we present sentences describing positive and negative experiences.

As can be seen, most sentences about positive experience refer to aspects related to the user's journey itself. Participants expressed satisfaction upon completing their use of the chatbot. On the other hand, negative feelings arise from technical frustrations, such as the implementation of certain features. An example of this is the criticism that emerged when users clicked on an OER and it opened in the same tab. Instead, participants preferred it to open in a new tab.

Table 3. Positive and Negative Experiences

How would you describe your experience after using the Chatbot?
(+) Positive
• Very good
• At first, there was the interest and expectation of interacting in a more natural way through Chatbot
• I found [it] practical
• The chat format is intuitive and fast
• Generally, after using chatbot, I had a positive experience
• [...] pleasant and efficient
• It is intuitive and made me feel more comfortable
• Chatbot met my expectations
(-) Negative
• The repetition of the results, despite the change in interaction commands, reduced initial feelings.
• I got a little annoyed that when you click on the links I am redirected instead of opening a new tab. It causes the inspection of several resources simultaneously to be laborious unnecessaryly.
• Just a little discontent with the results
• Unfortunately, it was below expectation, as the results were quite limited and, compared to the searches performed at EduCapes, the effectiveness fell short of the expected.

5.3. Usability

Usability also emerged from the feedback and comments received. This category represents the ease of use of the chatbot. We identified 17 comments regarding the chatbot's usability, which are listed below and organized into 5 distinct categories:

1. Design: The chatbot has a nice design
   - *Elegant*
   - *Pleasant*
   - *Very good*
2. Absence of errors: The chatbot did not show any errors during use
   - *No errors*
   - *Regarding irrelevant errors or results, I found nothing very significant*
   - *I do not think so*
3. Ease to use: The chatbot is ease to use
   - *It's easy to use (2x)*
   - *(The OER-Chat is) objective*
   - *I was impressed with the ease of use*
4. Responsive: The chatbot is responsible for smartphones/tablets
   - *Use on mobile devices is still viable, which is a plus*
5. Simplicity: The chatbot is simple to use
   - *Simple (2x)*
   - *Simple, "light"*

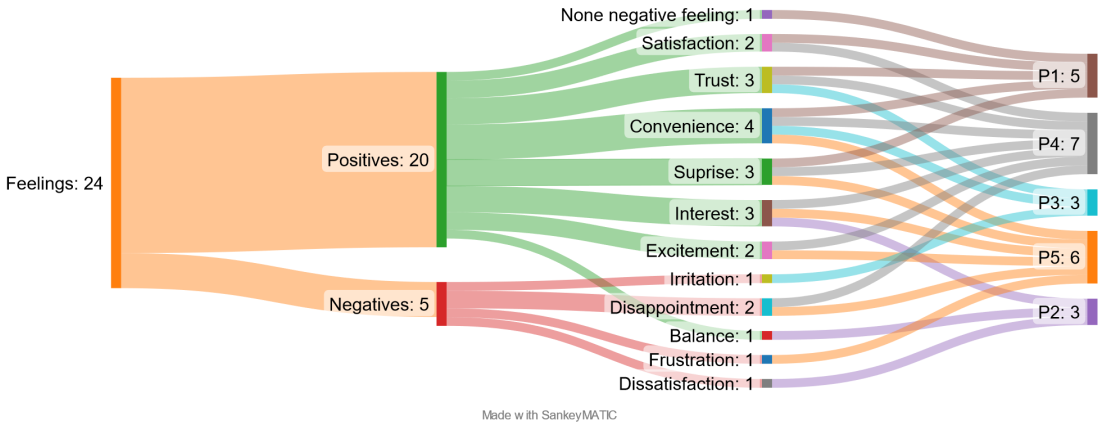

Figure 5. Feelings from each participant

- *I found it simple and practical*
- *Chatbot design is clean and straight to the point*
- *I found no difficulties when using chatbot*

5.4. Adherence of Results

Another point highlighted by users is the adherence of results after sending a command to the OER-Chat, we identified 26 comments about the adherence of results. Many of them hilighting positives and/or negatives aspects, which were classified into five categories:

1. Accurate: results were very accurate
   - *Very precise*
   - *Results were very accurate about what I sought*
2. Fast: results were very fast
   - *Returned the results at an impressive speed*
   - *The results appear very fast*
   - *Very low response time*
   - *Rapidness*
3. Relevant: results were very relevant
   - *Relevant to what I sought*
   - *Compared to the results of EduCapes repository, I thought the OER-Chat were more relevant*
   - *The results presented by Chatbot were quite relevant*
   - *It's adhering*
   - *Some materials have weird names, like "bus travel", but it is to teach programming. I think there the problem is with the material and the merit is from Chatbot that included himself having a generic title.*
   - *Very relevant results*
4. Repeated: results were repeated
   - *Repetition of results (2x)*
   - *At first it was to try to calibrate the interaction text for different results, until I realized that they repeated themselves and only the similarity rate showed variations*
   - *Results were quite limited*
5. Lack of adherence: results were not adherent
   - *Very similar to questions that are similar but present great differences. I asked "how to keep students motivated when teaching introductory programming?" And also "Which programming language to teach first?" And the results were two very similar lists, with little mentions directly.*
   - *Much less adherence than in the searches performed in EduCapes*
   - *Only two results seemed more adherent, but when accessing the area, I realized that the content of only one of them partially answered.*
   - *Unnecessary results have still appeared.*
   - *Results out of context with some frequency*
   - *But regarding motivation, I thought the results were less adherent*
   - *Irrelevant results prevailed*
   - *The effectiveness fell short of expected*
   - *Yes*

5.5. Functionalities

The last category that emerged from the participants' comments were ideas and suggestions for new features. These comments were grouped in two groups:

1. New tab: Two participants suggested opening a new tab when clicking on a resource
    - *It would be more practical if he opened a new tab*
    - *The fact that the links of the areas open on the same chatbot page can cause an interruption in the conversation, as the solution does not store the search history*
2. Probability: One participant commented that the probability calculator could be improved
    - *Some results pointed to probabilities higher than expected rates, in my opinion*

As can be seen, these comments represent aspects of future improvement derived from inherited intentions.

5.6. Deepening the discussions

We conducted a brief interview with the participants in order to identify the points of criticisms and understand negative feelings that emerged. Next, we summarize the main points observed.

Personal preferences: Personal preferences may have interfered with the validation process. For example, one participant stated, *"I think the only thing that irritated me in the chat was the behavior when clicking on the links. It keeps them in the same tab instead of opening another one. If it returns 5 results that I like, I have to click, view, go back, click, view, go back instead of opening all 5 tabs at once"*. This comment came from a participant who is working in the software industry (P3). He has experience with programming, and as a consequence, the usability of the system carries greater weight for him. That's why the feeling of "Irritation" arose. However, this irritation is not due to OER-Chat flaws, such as the absence of relevant results, but rather a technical aspect that may be addressed in the future with a new feature.

"Semantic" problems: Participants used more words to describe negative feelings than positive feelings. However, after the interviews, we noticed that the participants elaborated more on the problems so that they will be fixed in a future version. When they identified a positive feeling, they described it in a generic sentence, such as *"It's easy to use"* or *"It's Elegant and Pleasant"*. They did not elaborate further on those ideas because the description of this feeling was sufficient. Using the chatbot itself was a positive experience, as described by the P1 participant.

However, when a negative feeling arose, it was necessary to elaborate on this emotion. Participant P2 confirmed this perception: *"I had some feelings of frustration when using the chatbot. I detailed this in my feedback to point out these flaws and suggest future improvements. Well, I had immense interest in using the chatbot for education, and my criticisms are aimed at making it better in future versions"*. Thus, we realized that criticisms were not aspects to make the chatbot unfeasible, but suggestions for future improvements.

Is OER-Chat an option to Chat GPT? No. The purpose of OER-Chat is to provide OER for users according their demand. For this, several resources are stored and processed by the chatbot before providing a response. Furthermore, the focus of OER-Chat is to provide academic resources such as slides, ebooks, videos and courses that can be downloaded, reused and edited by users. Finally, OER-Chat also values the author of the resource, presenting his/her name and identification. Chat-GPT and other similar assistants focus on providing a textual response to each user input. Thus, the resource itself and the authors are not detailed.

While Chat-GPT and similar tools typically employ closed architectures, denying users access to models, content, data sources, and processing mechanisms, our proposal features an open architecture. This allows for user-driven extensions and improvements. The models, data, and content are publicly accessible and auditable by fellow researchers, promoting an organic and collaborative approach to educational development.

Complex questions: Furthermore, some users have posed intricate queries through OER-Chat, like "How can I maintain student motivation in introductory programming instruction?" While the chatbot can offer OER to assist with such inquiries, we acknowledge that addressing them may require simplification of the question or a thorough review of the provided results.

Qualitative perspective: We applied a qualitative paradigm to evaluate how users respond after using OER-Chat. Therefore, a real-world scenario was adopted instead of a controlled environment. The OER-Chat was validated by participants coming from various backgrounds including developers, teachers, and general users interested in programming and open education. As observed, all participants acknowledged the significance of OER-Chat. We collected data to identify patterns, subjects, and relationships, emphasizing the subjective interpretations

of each user who accessed the chatbot. We observed convergence among the participants, which is highly significant considering that the participants evaluated the OER-Chat individually, without any interaction among themselves.

Contributions to Open Education: OER-Chat aims to democratize the teaching and learning of introductory programming while supporting content producers. Instead of providing textual responses, OER-Chat suggests OER to users. This proposal supports content creators who produce OER and individuals who intend to use such resources. This initiative can facilitate the dissemination of open education and promote the reuse of educational material.

6. Conclusion and Future Work

This study presented an open chatbot to support the reuse of OER for introductory programming. The evaluation was carried out in a real-world scenario involving five participants with diverse perspectives on programming. While the number of participants is a potential limitation, it's worth noting that they came from various backgrounds and all acknowledged the effectiveness of OER-Chat. We observed a convergence of opinions among these participants, which is significant given that they individually assessed OER-Chat without any interaction among themselves. As a future work, we intend to use the OER-Chat for remixing resources and conduct new evaluations.

Acknowledgment

This work was supported by grant #2019/26871-4, São Paulo Research Foundation (FAPESP), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, and CNPq.

References

Carreira, G., Silva, L., Mendes, A. J., & Oliveira, H. G. (2022). Pyo, a chatbot assistant for introductory programming students. *Int.l Symposium on Computers in Education*, 1–6.

Dahiya, M. (2017). A tool of conversation: Chatbot. *International Journal of Computer Sciences and Engineering*, *5*(5), 158–161.

Deus, W. S. d., & Barbosa, E. F. (2020). An exploratory study on the availability of open educational resources to support the teaching and learning of programming. *IEEE Frontiers in Education Conference*, 1–9.

Herrera, A., Yaguachi, L., & Piedra, N. (2019). Building conversational interface for customer support applied to open campus an open online course provider. *International Conference on Advanced Learning Technologies*, 11–13.

Holotescu, C. (2016). Moocbuddy: A chatbot for personalized learning with moocs. *Int. Conference on Human-Computer Interaction*.

Kabir, R., Kabir, S., & Amin, S. (2015). Isolating informative blocks from large web pages using html tag priority assignment based approach. *Electrical Computer Engineering: An International Journal*, *4*, 61–72. https://doi.org/10.14810/ecij.2015.4305

Lei, L., & Liu, D. (2016). A new medical academic word list: A corpus-based study with enhanced methodology. *Journal English for Academic Purposes*. https://doi.org/https://doi.org/10.1016/j.jeap.2016.01.008

Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. (2019). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*. https://doi.org/10.1109/TE.2018.2864133

Okonkwo, C. W., & Ade-Ibijola, A. (2022). Revision-bot: A chatbot for studying past questions in introductory programming. *IAENG Int. Journal of Computer Science*.

Pandit, N. R. (1996). The creation of theory: A recent application of the grounded theory method. *The qualitative report*, *2*(4), 1–15.

Ray, P. P. (2023). Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*, *3*.

Roesler, D. (2021). When a bug is not a bug: An introduction to the computer science academic vocabulary list. *Journal of English for Academic Purposes*. https://doi.org/https://doi.org/10.1016/j.jeap.2021.101044

Selltiz, C., Jahoda, M., Deutsch, M., & Cook, S. W. (1991). Research methods in social relations.

Tlili, A., Shehata, B., Adarkwah, M. A., Bozkurt, A., Hickey, D. T., Huang, R., & Agyemang, B. (2023). What if the devil is my guardian angel: Chatgpt as a case study of using chatbots in education.

UNESCO. (2019). Recommendation on open educational resources (OER).

Verleger, M., & Pembridge, J. (2018). A pilot study integrating an ai-driven chatbot in an introductory programming course. *IEEE Frontiers in Education Conference*.