A chatbot system to support visually impaired students: a pilot experiment

Leonardo Padilha[1], **Alinne C. Correa Souza**[1], **Francisco Carlos M. Souza**[1]

[1]Coordenação de Engenharia de Software
Universidade Tecnológica Federal do Paraná – Dois Vizinhos – PR – Brasil

leonardopadilha@alunos.utfpr.edu.br

{alinnesouza, franciscosouza}@utfpr.edu.br

Abstract. *Students with visual impairment need for different methods to assist their studies such as read tools and conversation bots (chatbots). Assistive technologies can be incorporated as a training or support tool and specifically in the educational field as complementary material. This paper proposes a chatbot as a study that supports an intelligent system for visually impaired students. Finally, the paper reports a pilot experiment regarding the effectiveness, time of response, and speech output of the proposed system in two scenarios of the software engineering area. Overall, the system proved to be tolerant of typing errors and returned quick responses by 0,9 seconds on average. Our results also demonstrate the efficiency in output with speech employing a library called gTTS.*

1. Introduction

The learning of blind students is challenging since requires direct and instant support from the professors during the discipline. Different capabilities and needs of the blind students imply the necessity of technologies that would assist and complement their learning and feedback related to software engineering contents such as programming, system modeling using Unified Modeling Language (UML), and database for instance. Assistive technology is a great ally in the process of teaching visually impaired students. Among the existing technologies, it is important to emphasize the use of chatbots. Computer-based chatbots are getting to be distinctly famous as an intuitive and successful open framework between humans and machines. A chatbot is a conventional agent that is able to interact with users in a given subject by using natural language [Huang et al. 2007].

Chatbots for educational purposes are interesting because they offer many advantages in a learning environment as it allows students to interact when necessary, facilitates the identification of information and provides a different form of assimilating content. Additionally, it may be used as a content platform, where professors can insert additional material. In this paper, we present a chabot system called EducationaL Intelligent chatbOT (ELIOT). ELIOT aims at supporting and improving the visually impaired students learning of Software Engineering (SE). Our system is based on two elements: *(i)* software engineering concepts; and *(ii)* artificial intelligence techniques, which should allow the interaction with these students through text and speech. It is important to highlight that ELIOT was created with a focus on SE due to the need to assist the learning of a blind

student in the Software Engineering course of a University of Southern Brazil. However, the knowledge base can be adapted to any area or content.

Overall, the contributions of the present work can be summarized into the following points: *(i)* sets out a chatbot interface to support the learning of visually impaired students; *(ii)* interaction through textual or speech; *(iii)* provides a different form of assimilating content; and *(iv)* makes a pilot study to evaluate the interaction and effectiveness of chatbot system; The remainder of this paper is organized as follows. Section 2 addresses the related work, which is not exhaustive but only the most relevant ones. Section 3 outlines the ELIOT Chatbot system and describes each of its phases. Section 4 presents a pilot case study to evaluate the Chatbot. Section 5 discusses the results of the Chatbot, and finally, section 6 presents the conclusions and topics for further investigation.

2. Related Work

We were able to find several studies that present the chatbots application in different contexts, as health, games, and business. However, few studies [Alves 2017], [Pereira 2016], [Benotti et al. 2014], [Silveira and Thiry 2010], [Batista et al. 2009], [Inoue and Vinciguerra 2009], explore specifically the use of chatbots to support the computer science education, but not for visually impaired students.

Alves [Alves 2017] presents Laura as a chatbot to answer Java questions. Data mining techniques were used to extract knowledge from the StackOverflow online discussion forum to serve as Laura's knowledge base. Pereira [Pereira 2016] describes the design and implementation of a Telegram bot, @dawebot, for training computer science students in any subject using Multiple Choice Question (MCQ) quizzes. The authors in [Benotti et al. 2014] addresses the educational software tool based on a chatbot that helps students learn basic Computer Science concepts such as variables, conditionals, and finite state, among others. In order to increase student engagement was used a gamified, inquiry-oriented, structured task was where students program a chatbot and get automated progress feedback.

Silveira and Thiry [Silveira and Thiry 2010] presents a chatbot to help the learning of concepts in the software measurement. The chatbot was developed for the web platform using the Java programming language and AIML technology. Batista et al. [Batista et al. 2009] address multi-agent systems models and a chatbot based on the middleware agent-layer framework. This chatbot was implemented to be utilized in virtual educational environments to provide an interface to guide the students in the explanation of the FAQs about the Java programming language. Inoue and Vinciguerra [Inoue and Vinciguerra 2009] propose a distributed chatbot called NICOLE that aims to aid in the learning of a programming language. The chatbot intelligence is based on Fuzzy Logic techniques.

In our study, we developed a chatbot for visually impaired students. The difference between our chatbot and the others is related to input, output, and knowledge database. It gets the input text or speech format from the user, and the output of this application will be given in both formats too. The knowledge database can be extended to any domain since the system provides to professors with the possibility of entering information related to its content. Therefore, the knowledge database can be modified and improved according to the professors' needs.

3. ELIOT - A Chatbot System

In this study, we propose a chatbot system to aid the teaching of Software Engineering for blind students called ELIOT. The system idea is to provide a mechanism to the professor's which they can integrate knowledge into the system. This system is being defined in a project, specially designed for software engineering blind students, but it can be applied to any knowledge area. As shown in Figure 1, ELIOT consists of four sequential steps, namely: (i) training; (ii) input question; (iii) processing; and (iv) response. It is important to highlight that to development ELIOT, we performed requirements specification using User stories, and created use case and activities diagrams.

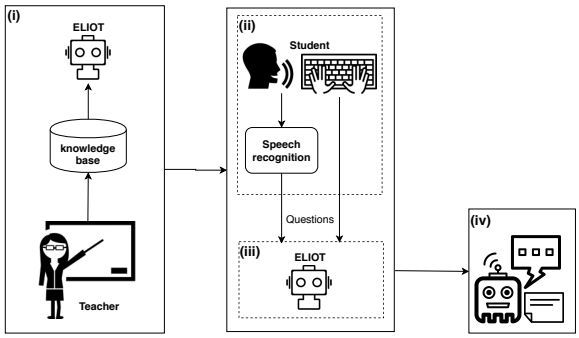


Figure 1. ELIOT - a chatbot system.

3.1. Step 1: training

The first step focuses on creating a database containing information on a specific subject in software engineering. This phase aims at providing a template to the professor that inserts its discipline for the bot to learn. For the chatbot training, we create a database using two areas in software engineering, which are: software requirements and software modeling. The database is composed of phrases, questions, and answer written in Brazilian Portuguese, which are divided into three categories: *(i)* greetings; *(ii)* general conversation; and *(iii)* specific conversation. The first category consists of greetings commonly used to start a conversation. The second category represents conversations about any general topic, such as daily routines, activities, etc. Finally, the last category addresses specific contents about a given area.

In this paper, the chatbot system is developed through a Python library called Chatterbot that implement a dialog engine and chatbots concepts. The Chatterbot utilizes different techniques to make a conversation from a statement, the larger the database, the greater the accuracy of a response to a specific context. Furthermore, to get the inputs, we use a natural text by keyboard from two interfaces (terminal and web interface) and we use speech input, which employed speech recognition and artificial intelligence techniques.

Speech recognition techniques have been widely investigated over the past years. However, this topic is still a challenge due to the subtleties of each language. Most modern speech recognition systems use machine learning or neural networks to classify the speech correctly. Therefore, in this system, we use a trained neural network from google cloud API, since, is considered the best mechanism to speech recognition.

3.2. Steps 2 e 3: input question and processing

The second module (processing) determines the logic for chatterbot generates the responses based on user input, the module works by using methods called logic adapters. The methods select a response for a question by searching into the database for the closest matching known statement according to the input. After, from the known statements, it chooses a response from the selection of responses to that statement or question and returns it to the user. In this paper, we utilized a popular method to compare the proximity of the data in the process data, called the Jaccard index and also known as the Jaccard similarity coefficient. Jaccard similarity is often used for comparing similarity, dissimilarity, and distance in datasets [Albatineh and Niewiadomska-Bugaj 2011].

3.3. Step 4: conversion text to speech and response

Lastly, the four step allows the chatbot to return a response based on the previous step as we can see in Figure 2. The library provides some text methods to send the output, such as string, terminal output, emails engine, etc. For our chatbot system, we obtain the responses as a string and manipulate them using a speech synthesizer to convert text to speech.

Figure 2. ELIOT - web interface

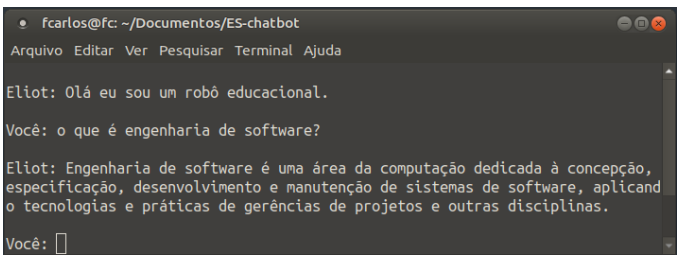


Figure 3. ELIOT - terminal interface

This conversion was performed utilizing a Python library and CLI tool to Translate's text-to-speech API called Google Text-to-Speech (*gTTS*) [1]. The response into a

[1]https://pypi.org/project/gTTS/

string is passed as a parameter to the *gTTS* function, and then it is converted and reproduced as speech. In addition, for comparison purposes, we use the *pyttsx* library [2] to convert text to speech, as well. The main difference between the two libraries is that gTTS needs the internet to perform the conversion since it employs Google technologies to work.

The iteration with ELIOT can be made through a command terminal or an interface web which the student writes a question or statement and the chatbot returns a response in text and speech as shown in Figure 2 and Figure 3. For the terminal interface, we utilized the python libraries for the development of the chatbot and a Linux terminal. Whereas, the web interface was developed using a python web framework called flask[3], HTML, javascript, and the same python libraries for chatbot development.

4. Experimental Study

In this study, we conducted a pilot experiment to assess the effectiveness ELIOT to answer Software Engineering questions for visually impaired students. The methodology used to conduct this experiment is based on Wohlin [Wohlin et al. 2012].

4.1. Research Questions and Goal Definition

We used the Goal-Question-Metric (GQM) model [Basili and Weiss 1986] to set out the objectives of the experiment can be summarized as follows: *"Analyse **ELIOT system** for the purpose of **evaluation** with respect to **response quality, response time and speech output effectiveness** from the point of view of **experimenters** in the context of **the software engineering area**."*

According to the GQM presented above, we defined four research questions:

RQ_1**: How effective is the ELIOT system to answer questions through text?** In answer to RQ_1, the effectiveness of the ELIOT system was evaluated by the number of correct answers from questions inserted by the professor.

RQ_2**: How effective is the ELIOT system to answer questions through speech?** For RQ_2, we analyzed the number of correct answers get by Eliot applying the phrases/questions asked by professor through speech.

RQ_3**: How quick is the ELIOT system to answer questions of software engineering?** To answer RQ_3 we analyze the average time of ELIOT answer's. We created five questions in both interfaces (terminal and web).

RQ_4**: How effective is the speech output in the ELIOT system to answer questions of software engineering?** In answer to RQ_4, we analyze the speech output in the ELIOT system for the questions asked. For this, we compare two distinct libraries as gTTS and pyttsx. The speech outputs of both libraries were assessed by professors of the software engineering area.

4.2. Experiment Design

This pilot study comprises four different experiments (E = e_1; e_2; e_3; e_4). For all experiments, the phrases/questions were created in the Brazilian Portuguese language. In

[2]https://pyttsx.readthedocs.io/en/latest/

[3]http://flask.pocoo.org/

addition, all experiments were conducted through the Linux time utility on a laptop with Intel Core i7 1.8 GHz CPU, 8GB memory in the Ubuntu 18.04 operating system.

The first and second experiment (e_1; e_2) are related to RQ_1 and RQ_2, respectively. Both experiments aim to evaluate the effectiveness of the ELIOT system to answer questions through text and speech. For the e_1, one question with five different typing errors was defined according to software engineering content. For the e_2, three different phrases/questions were specified for each category, on a total of nine questions. Each phrases/questions the experiment was performed 10 times. These phrases/questions are presented in Table 1.

Table 1. Brazilian Portuguese Phrases/Questions for the experiment e_2.

Phrases/ Questions	Greetings	General Conversation	Specific Conversation
RQ1	Oi.	E você?	Você precisa identificar os elementos que compõem o diagrama de classe.
RQ2	Olá.	Bom vê-lo.	Classe, atributo, método e relacionamento.
RQ3	Tudo bem?	Eu me chamo Eliot	Uma classe é representada por um retângulo dividido em três partes, a primeira contém o nome da classe, a segunda os atributos e a terceira os métodos.

The third experiment (e_3) aims to answer RQ_3. In this experiment, we assessed the quickness of the ELIOT system to answer questions of software engineering. For this experiment, we selected one question of each category used in the experiment e_2, more specifically phrases/questions $RQ3$. In addition, we compare the time of the answers obtained in two different scenarios: *(i)* terminal interface and *(ii)* web interface. Finally, the last experiment (e_4) is related to RQ_4. In this experiment, ten volunteers evaluated the effectiveness of speech output in the ELIOT system to answer questions. These volunteers created six questions, being two for each category. In addition, we compare the evaluation performed by volunteers using two libraries: *(i) gTTS* and *(ii) pyttsx*.

All volunteers were professors from three different universities and were distributed among the courses of Software Engineering, Computer Science and Information System. There were 4 females and 6 males. It is important to highlight we accomplish this pilot experiment with Professors for two reasons: *(i)* they will manipulate the database through the insertion of knowledge; and *(ii)* the role of Professors in the teaching-learning process to visual impairment students. The volunteers had to evaluate the speech output effectiveness in the ELIOT system using the Likert scale [Likert 1932], which consisted of five criteria (5 - Extremely, 4 - Very, 3 - Moderately, 2 - Slightly, 1 - Not a bit).

4.3. Experiment setup

The procedure followed during the execution of the experiment consists of the following steps:

1. **Analyzing of ELIOT system effectiveness:** we evaluated the number of correct answers from:*(i)* questions with typing errors, and *(ii)* questions spoken by Eliot.

2. **Computing the time response:** the average of time response taken by ELIOT system from each question was computed according to three categories (greeting, conversation, and suggestion). The time was computed 30 times for each interface (web and terminal).
3. **Description of ELIOT:** we performed a description with explanations and examples of questions that can be asked by volunteers in the ELIOT system.
4. **Interaction volunteer with ELIOT system:** the volunteers had access to the ELIOT system in which they performed the questions. It is worth highlighting that the volunteers had no contact with the knowledge base of the system.
5. **Comparison between the *pyttsx* and *gTTS* libraries:** we compared the *pyttsx* with *gTTS* libraries to assess the effectiveness of the speech output in the ELIOT system. The speech outputs of both libraries were assessed by 10 professors $P = (p_1, p_2, p_3, ..., p_{10})$ of the software engineering area.

5. Results and Discussion

In this study, we conducted a pilot experiment to assess the effectiveness ELIOT to answer Software Engineering questions for visually impaired students. The results are shown following in separate subsections according to each research question.

5.1. Effectiveness of the ELIOT system to answer questions through text (RQ_1)

The effectiveness of the ELIOT system was evaluated by the number of correct answers from questions inserted by the professor. The answers generated by the ELIOT system are shown in Figure 4. The results indicate that the ELIOT system is effective once it generates the same answer for question with typing errors. As Figure 4 shows, we executed ELIOT with different typing errors to analyze its output. The results prove that even with extremes errors, the proposed system can return the correct response. Therefore, we observed that this type of error does not affect ELIOT system performance. Thus, this system can be seen as a complement to the learning of visually impaired students through text input.

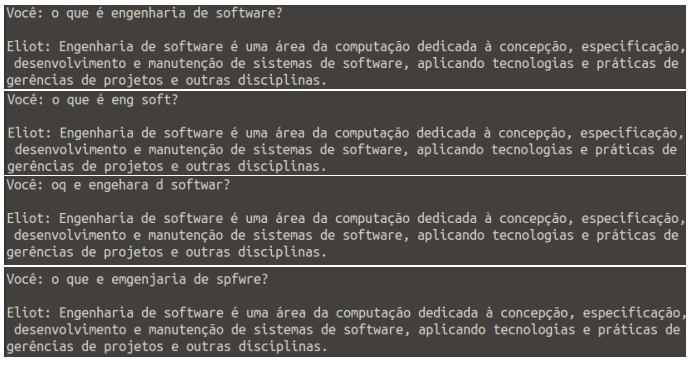


Figure 4. The same answer was generated by ELIOT system for the question with five different typing errors in Brazilian Portuguese.

5.2. Effectiveness of the ELIOT system to answer questions through speech (RQ_2)

In this RQ, we analyzed the number of correct answers obtained by the ELIOT applying the phrases/questions asked by the professor through speech. The number of correct answers is shown in Figure 5. We found that in the Greetings category, on average, 96% was

achieved using $RQ1$, $RQ2$, and $RQ3$ phrases/questions. For the General Conversation, 88.67% was achieved with $RQ1$, $RQ2$ and $RQ3$. And, for Specific Conversation, 82.67% was achieved with $RQ1$, $RQ2$ and $RQ3$. We noticed that the more specific the topic is the more difficult it is for the system to recognize the speech input correctly. This may have occurred due to features of language or the small number of samples in the database. The results indicate that the ELIOT system is effective, once answers questions through speech.

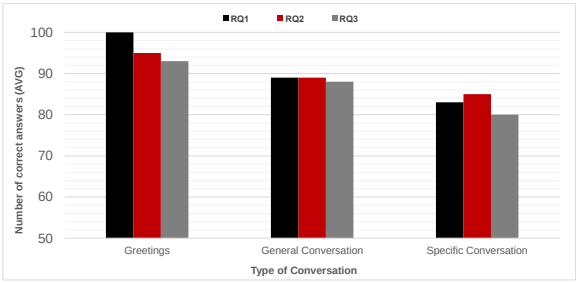


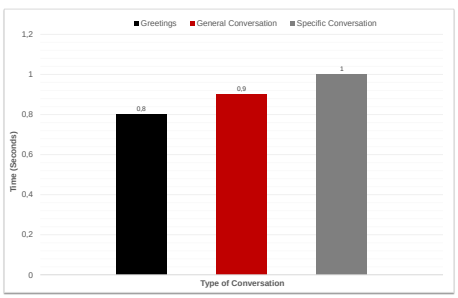
Figure 5. Number of correct answers get by Eliot from the phrases/questions asked by professor trough speech

5.3. Quickness of the ELIOT system to answer questions (RQ_3)

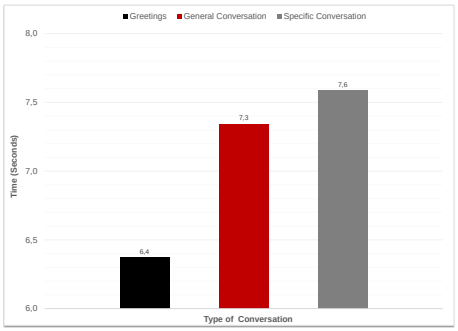
In this research question, we computed the average time response taken by the ELIOT system for the questions according to interface (terminal and web) and three categories (greeting, general conversation, specific conversation). Figures 6a and 6b present the time response of ELIOT system for terminal and web interface, respectively.

We observed that for all categories that the terminal interface time (Figure 6a) is better than the web interface. For the greeting category, the time response using the terminal interface was 5,6 seconds faster than the web interface. The general conversation category was 6,4 seconds faster and the specific category was 6,6 seconds. In addition, it is important to highlight that regardless of the type of interface used, the response time is proportional to the degree of complexity of the conversation.

Therefore, the ELIOT system is able to generate the responses quickly once its average response time in the terminal interface for the three categories is 0.9 seconds. Furthermore, it is natural to expect that the time use of the terminal interface is less than



(a) ELIOT responses in terminal interface



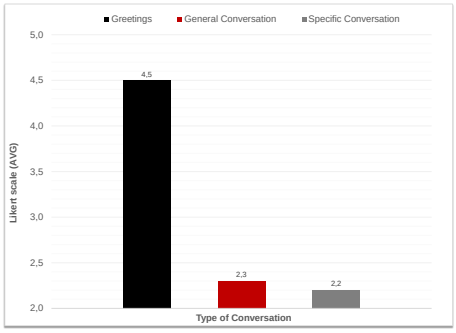
(b) ELIOT responses in web interface

Figure 6. Response time of the ELIOT system.

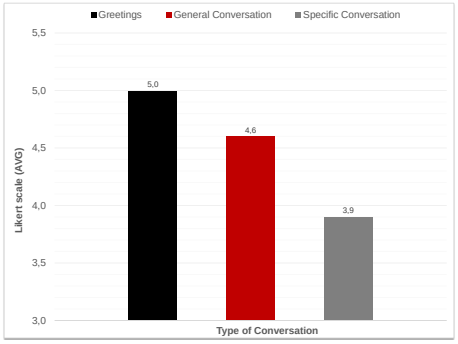
the web interface. It occurs because the source code is accessed faster by an operational system than a web host or local server. Finally, we note the need to use techniques to optimize web solutions that enable a chatbot that include these features and return answers quickly.

5.4. Effectiveness of speech output in the ELIOT system (RQ_4)

In this research question, a comparison was made between the *pyttsx* and *gTTS* libraries to evaluate the effectiveness of the speech output in the ELIOT system. Figure 7a and Figure 7b presents the results of volunteers evaluation for *pyttsx* and *gTTS* libraries, respectively. According to the results presented in Figure 7a and Figure 7b, we observed that the *gTTS* library obtained a better evaluation compared with the *pyttsx* library. For all categories the *gTTS* library is better than *pyttsx* library. One hundred percent of the subjects assessed that using the *gTTS* library for greeting category is "extremely" effective. 92% evaluated that the questions of general conversation category are "very" effective, and 78% evaluated that the questions of specific conversation category are "moderately" effective.



(a) ELIOT speech output using *pyttsx* library.



(b) ELIOT speech output using *gTTS* library

Figure 7. Speech output in the ELIOT system using libraries based on volunteers opinion.

On the other hand, the *pyttsx* library did not get as good ratings as the *gTTS* library. After using the *pyttsx* library, 90% of the subjects evaluated the questions of the greeting category as "very" effective. 46% assessed that the questions of the general conversation category are "slightly" effective, and 44% assessed that the questions of specific conversation category are "slightly" effective. The *pyttsx* library is a speech synthesizer platform that supports the native text-to-speech libraries of Windows and Linux at least. However, it has no good conversion to some words in certain languages, such as Brazilian-Portuguese. Therefore, according to the results, the speech output in the ELIOT system is effective using the *gTTS* library.

6. Final Remarks

In general, students with any special needs have faced many obstacles to achieve space and recognition in the job market. However, unfortunately, very few students can realize this desire which makes this research an essential step in helping these people. Assistive technology has increasingly become a fundamental instrument to aid interaction. It aims to provide people with special needs more independence, quality of life, and social inclusion by improving their skills and learning. In this paper, we propose a chatbot to

support and improve the teaching of visually impaired students. Herein, the idea is not to replace screen reading tools used by visually impaired people. We aim for a system as an additional tool and a mechanism for teachers to embed content and provide it to the students.

Thus, our results suggest that our chatbot system is promising for the following reasons: *(i)* it assists the learning; *(ii)* returns a response to students through textual/speech questions; and *(iii)* provides a different form of assimilating content for visually impaired students. Our work opens space for improving ELIOT system. More specifically: (*i*) devise new similarity metrics; (*ii*) expand the knowledge database , (*iii*) optimize the chatbot in web platform; (*iv*) test others technologies of text-to-speech conversion; and (*v*) evaluation with the visually impaired students using the ELIOT system.

References

Albatineh, A. N. and Niewiadomska-Bugaj, M. (2011). Correcting jaccard and other similarity indices for chance agreement in cluster analysis. *Advances in Data Analysis and Classification*, 5(3):179–200.

Alves, M. F. S. (2017). Laura: Um chatterbot para responder perguntas sobre java. Master's thesis, Universidade Federal do Ceara, Quixada, Brazil.

Basili, V. and Weiss, D. (1986). A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, 10(6):728–738.

Batista, A. F. M., Marietto, M. G. B., Barbosa, G. C. O; França, R. S., and Kobayashi, G. (2009). Multi-agent systems to build a computational middleware: A chatterbot case study. In *Proceedings of the 4th International Conference for Internet Technology and Secured Transactions, (ICITST)*, pages 1–2.

Benotti, L., Martínez, M. C., and Schapachnik, F. (2014). Engaging high school students using chatbots. In *Proceedings of the Conference on Innovation Technology in Computer Science Education*, ITiCSE '14, pages 63–68.

Huang, J., Zhou, M., and Yang, D. (2007). Extracting chatbot knowledge from online discussion forums. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*.

Inoue, P. N. and Vinciguerra, D. (2009). Chatterbot para auxiliar no aprendizado de uma linguagem de programacao. *Anuario da Producao de Iniciacao Cientifica Discente*, XII(13):273–283.

Likert, R. (1932). A technique for the measurement of attitudes. *Journal Archives of Psychology*, 22(140):1–55.

Pereira, J. (2016). Leveraging chatbots to improve self-guided learning through conversational quizzes. In *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*, TEEM '16, pages 911–918.

Silveira, J. L. and Thiry (2010). Desenvolvimento de um chatterbot para Área de medição de software. In *Computer on the Beach*, pages 87–91.

Wohlin, C., Runeson, P., Host, M., Ohlsson, M. C., Regnell, B., and Wesslen, A. (2012). *Experimentation in Software Engineering: An Introduction*. Springer-Verlag Berlin Heidelberg, 1st. edition.