LIA: A Mathbot Assistant for First-grade Students

Lefteris Moussiades and George Zografos

***Abstract* — Nowadays, chatbots are helpful in many areas, including education. This article presents a chatbot that aims to help first-grade students get acquainted with arithmetic operations. The architecture we propose has a general character, scalability and can build other closed-domain chatbots. The results of an initial pilot study are satisfactory and encourage us to continue the research.**

***Index Terms* — chatbot, educational software, mathbot, natural language interface.**

I. Introduction

Conversational agents or chatbots are software encapsulating a natural language interface. Scopus search with the terms "chatbot", "conversation agent" or " conversational interface" reveals the increased interest that chatbots attract in the research community in recent years. Relevant publications jumped from almost zero in the 1990s to around 200 per year in the mid-2000s; furthermore, they exceeded 1000 per year by 2019 [1].

Indeed, chatbots appear helpful in many areas of activity, including customer service [2], e-commerce [3]-[5], food industry [6], [7], health sector [8]–[11] and robotics [12]–[14]. However, our interest is focused on Educational applications, where chatbots provide educational content and personal assistance to learners [15] or help them with administration tasks [16]. In this context, we decided to experiment with a chatbot specifically designed to help first-grade primary school students to learn the basic mathematical operations, i.e., addition, subtraction, multiplication and division.

Therefore, we search for a chatbot designed explicitly for first-grade primary school students. However, we did not manage to find a suitable one. Some so-called mathbots are traditional software that does not use a natural language interface [17], [18]. Anish Kumar Nayak et al. specializes in transforming problems formulated with text into equations and solving equations [19]. Another mathbot is aimed at 4th graders [20]. Another approach requires a complicated procedure to set up a local installation, making it impractical for use from first-grader [21]. Work in [22] does not implement a voice communication channel, which is essential for interaction with young students. Finally, the best choice for our knowledge is the MathBot presented in [23].

However, this and other chatbots mentioned above do not support the Greek language, which is a requirement for our objective.

Therefore, we decided to design our Mathbot, which we call LIA from Learn Interactively Arithmetic. LIA can be found at https://thesis.cs.ihu.gr/gezozra/mathbot/LIAv4/lia.php.

This article provides an overview of the technology that developers can use for such a chatbot. We justify our choices regarding the appropriate for our purpose technology and proceed to the design. We present the architectural design and give all the essential details about the user intentions identified by the LIA. Our contribution goes beyond the limits of LIA as the proposed architecture can be used to build other closed-domain chatbots in education and beyond. Finally, we conduct an initial pilot study to evaluate the LIA.

The rest of the paper is structured as follows: What technology we chose to implement the LIA and why it is explained in section 2. In section 3, we present the objective of the LIA, the strategy it follows to achieve its objective while we are giving some definitions necessary for understanding the rest of the article. Next, in section 4, we suggest an architecture explaining all the essential details. Section 5 describes the user interface, while in section 6, we present an initial pilot study results. Finally, in section 7, we discuss the findings and announce the new directions of our research.

II. Technology

The essential operation of a chatbot is identifying the user intent [24]. The chatbot receives a user utterance, based on which it tries to determine what the user intention is. In addition, the context of the discussion may be used in the user intent identification when necessary [25]. The context is formed by previously given user utterances and the chatbot's answers. The process of intent identification often includes entity recognition [26]. An entity is one or more words that carry vital information for intent identification. For example, the utterance: "Can we discuss it in half an hour?" contains a timing which further identifies the user's intention to chat with the bot. Timings, locality information, and names are some entity types often used.

Depending on the Knowledge domain, chatbots are discriminated in generic, cross-domain and closed-domain ones [1]. Generic chatbots can discuss any topic, cross-domain chatbots discuss topics from more than one domain, and closed-domain chatbots discuss themes from a particular field. A mathbot is a closed-domain chatbot. A closed-domain chatbot should incorporate a strategy that will direct

Submitted on July 11, 2021.
Published on August 01, 2021.
Lefteris Moussiades, Department of Computer Science, International Hellenic University, Greece.
(e-mail: lmous@cs.ihu.gr)

George Zografos, Department of Computer Science, International Hellenic University, Greece.
(e-mail: gzozra@cs.ihu.gr)

the dialogue towards specific goals.

A variety of technologies can be utilized to implement a chatbot. One option is the use of a chatbot development platform. Commercial platforms like [27], [28], [10] offer many advantages, but they require an operational cost. LIA, however, should be distributed and used free of charge by young students. There is also a variety of open-source platforms like RASA [29], Botkit [30], Chatterbot [31], and many others. Open source platforms could be a choice for our current design of the mathbot. Still, it is unclear if they would help future development when the chatbot becomes more complicated [32].

Therefore, we decided to develop the LIA from scratch. For this purpose, it is possible to use a variety of different approaches. The user intent is usually identified either by a neural network or using an appropriate pattern matching language. In the first case, the user utterance words are coded into vectors, which feed the neural network [33]. Various neural networks are used as classifiers, including linear SVM algorithms to Recurrent Neural Networks [34] and Sequence to Sequence models [24]. However, these approaches need extensive training sets, which are not available in our case. In general, for closed-domain chatbots, the use of a scripting language seems to be preferable over a neural network approach as it is challenging to find suitable training data for a specific purpose [1].

So, we decided to design LIA by combining a pattern matching language with a strategy that will direct the dialogue to LIA's specific goals.

We evaluated three pattern matching languages, the AIML, Rivescript and Chatscript. Although there are pros and cons to all three, we chose the Chatscript because its support of concepts gives it more expressive power than its competitors [35].

More specifically, the following technologies have been utilized in the current implementation of LIA:

- A Chatscript server as a backend dialogue flow engine [36].
- Google's Web Speech API for Automatic Speech Recognition (ASR) [37], which is only available in newer versions of Chrome and Edge browsers at the time of writing.
- The Responsive Voice API [38] for Text to Speech (TTS) conversions.
- The PHP programming language [39] for the communication between the frontend and the Chatscript engine.
- The Javascript programming language [40] for the frontend implementation.

However, the design of LIA allows its implementation on any platform that supports ASR and TTS and at least one programming language that supports regular expressions, e.g., Java, Javascript, C #, C objective. Therefore, LIA can easily be implemented and constitute the natural dialogues' interface in various physical robots' platforms.

III. Objective, Strategy and Essential Definitions

The *Objective* of LIA is to help the student practice the four basic arithmetic operations. It guides the student to solve quizzes on addition, subtraction, multiplication and division. The tutor determines a series of exercises that they wish to be solved by the students.

Our S*trategy* on the dialogues' formulation aims at the student's solution to all the provided quizzes. At the same time, the student can express himself to facilitate his practice through the interaction with the LIA.

The *Session* is the dialogue flow initiated when a student enters their name and ends when all quizzes have been answered, or the student raises a *Withdrawal* intent. A session can be conducted in different periods. Thus, a student can solve part of the exercises and the rest at another time. The student is identified by the name they enter and the IP address from which they enter. Although this student identification process is not entirely reliable, we preferred it over the registration and login process. The latter is complex and requires an email address that young students rarely have.

We define an *R-concept* as a set of phrases (including one-word expressions or symbols). The phrases in an R-concept may be synonymous but not necessarily. We denote an R-concept as c(p), where p is a phrase representing the specific R-concept. For example, c(modify) indicates a set that may include the words alter, change, adapt, adjust. Correspondingly, c(operation type) may consist of addition, subtraction, multiplication, division and the symbols +, -, x, ÷. Additionally, an R-concept, denoted as c(number), is defined to represent a number.

The *Rules* combines patterns and R-concepts and are used to identify the student's intention. We denote a Rule based on the R-concepts it includes. For example, a rule containing c(p1) and c(p2) is denoted as [c(p1), c(p2)].

We call *Conversation state*, each phase in which the LIA awaits a response from the student. We distinguish four conversation states:

- *Operation selection*: LIA has motivated the student to select an operation.
- *Pending quiz*: LIA has provided the student with a question to solve.
- *Help*: LIA has presented a help table and expects the solution to the pending question.
- *Break time*: During a break.

The *Context* keeps session history. Helpful information can be obtained from the context, including the student id, the Conversation state, the student evaluation record, or the pending question if there is one.

A *Confirmation-action* is a message from LIA to the student, which accepts only "yes" or "no" as a response.

IV. Architecture

Here, we first present the overall architecture of LIA and then analyze each part in detail.

As Fig. 1 shows, the student's phrase is directed to the Intent Identification Module (IIM). If the phrase contains a specific rule, it is successfully identified. If the phrase contains R-concepts that do not constitute a particular rule, the Ambiguity Handling Unit (AHU) undertakes to clarify the user's intention. If the intent is still not recognized, the user phrase updates the database of unidentified intents. An event generator intervenes in the whole process while the context is taken into account. If the intent has been identified, its ID is

sent to the Answer Generation Module (AGM); otherwise, IIM sends to AGM an unidentified intent.

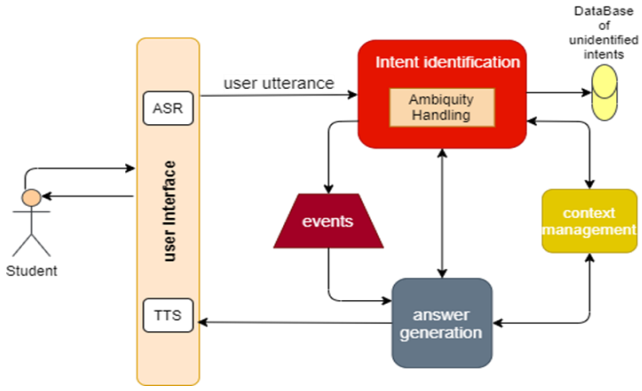

Fig. 1. Architecture Overview of the LIA.

A. *Intent Identification Module*

Here we propose an intent identification module based on the following principles:

- Identifiable intents should be determined as a first step of the design process.
- The user must have at least one way of expressing his intention clearly.
- LIA should recognize every identifiable intent at each conversation state.

LIA identifies the following intents:

1. *Operation-Selection*: The student chooses the operation with which they wish to practice. The intent is identified by Rule [c(select), c(operation type)], i.e., when student phrase contains both R-concepts.
2. *Change-Operation*: The student asks to replace the current operation without specifying the replacer type. The intent is identified by Rule [c(change), c(operation)].
3. *Change-Operands*: The student asks to replace the operands of the current operation. The intent is identified by Rule [c(change), c(operands)].
4. *Help-Request*: The student asks for help on the current question. The intent is identified by Rule [c(help)].
5. *Evaluation-Request*: The student asks to be informed about his score. Identification Rule [c(score)].
6. *Answer-Submission*: The student answers a quiz. The identification rule does not include R-concepts. It is identified based on the presence of one number in the student utterance.
7. *Inability-to-Resolve*: The student declares inability to solve the current quiz. Identification Rule [c(refusal), c(resolve)]
8. *Break-Request*: The student asks for a break. Identification Rule is [c(break)].
9. *Withdrawal*: The student declares his intention to leave. The identification Rule is [c(goodbye)]".
10. *Question*: The student asks for the result of an operation. The intent is identified by Rule[c(number, c(number, c(operation symbol))], i.e. there need two numbers and one operation symbol.
11. In addition to the intents mentioned above, the Intent Identification Module output may also be the *Unidentified-Intent*.

B. *Ambiguity Handling Unit*

The Ambiguity Handling Unit is internal to IIM. Similarly to IIM, the AHU consists of a set of rules. Any student utterance that has not already been identified is forwarded to AHU. We explain the AHU operation with an example.

LIA will assign the student phrase to the intent Change-Operation if it contains the R-concepts c(change) and c(operation), while if it has c(change) and c(operands), it will assign the phrase to the intent Change-Operands. But what if the phrase contains c (change) and none of the c(operation) or c(operands)?

In this case, both Change-Operation and Change-Operands are possible intentions of the student. Then, AHU intervenes and sends to the student confirmation-actions for each possible intent. If it receives the confirmation, the intent is identified, and the confirmation process is interrupted without checking the remaining intents.

If all candidate intents are checked, and none is confirmed, the result is Unidentified-Intent. In this case, IIM updates the database of unidentified intents with the corresponding student utterance. The development team uses utterances in the unidentified intents database to improve LIA further.

C. *Event Generator*

Event Generator (EG) may produce the following events:

- *Break-termination:* It is activated from the timer that the break request sets.
- *Prolonged-inactivity:* It is activated by a timer that counts the idle period, i.e., when the student has not answered the LIA's last question.
- *Quiz-end:* There are no more questions. All questions have been asked.

D. *Answer Generation Module*

The AGM receives input from the IIM, the Context Management Module (CMM) and the EG and outputs a response to the user.

LIA recognizes ten intents, 11 including the *Unidentified-intent*. As we stated, LIA should identify any intent at any conversation state. Therefore, at this level, we discriminate 11×4 cases that may require a distinct response. Additionally, events require that AGM will produce a message to the user.

The event *Prolonged-inactivity* may occur at any conversation state except Break time.

The event *Quiz-end* may occur after receiving an answer to a pending question or when a Withdrawal intent is recognized. The event *Break-termination* may occur only at Break time.

Therefore, cases that may require a distinct response from LIA are 11×4+3+1+1=49. Responses are produced by Response-handlers. Given an intent or event and the Conversation state, AGM selects an appropriate Response-handler. A Response-handler implements a strategy suitable to answer to a particular intent or event at a Conversation state.

For example, a student raises a *Break-Request* while at Conversation-state *Pending-question*. The corresponding response-handler tries to postpone the break until a predefined time limit, which the tutor defines as a system parameter. If the student accepts the postponement, LIA

resends the pending question; otherwise, a break is initiated. The communication between student and response-handler is carried out exclusively with confirmation-actions to prevent the introduction of new intents and restricts the conversation until it is clarified whether LIA will approve the break.

Each Response-Handler implements a different strategy that should be compatible with the global LIA's strategy. Response-Handlers can access the session history as needed. Let consider another example. Assume a student raises an *Inability-to-Resolve* intent while at conversation state *pending-question*. The corresponding *Response-Handler* provides a help table and motivates the student to resolve the question.

However, if the student raises an I*nability-to-Resolve* intent again, the conversation state has changed to *Help*, and a different intent-handler is invoked. It provides the student with the solution and advances the quiz to the next question. A Response-handler encapsulates an algorithm specifically designed for a given intent and a given conversation state. Firstly, the algorithms of response-handlers calculate a response-set, which is a set of alternate phrases having the same meaning.

The final answer to the student is a phrase selected randomly from a response set. In this way, LIA gives the impression of a more natural dialogue.

V. User Interface

In all interactive applications, user interface design is critical to user acceptance of the application. Of course, this also applies to the LIA, even more so, as it is addressed to primary school students. Fig. 2 shows the main view of the LIA.

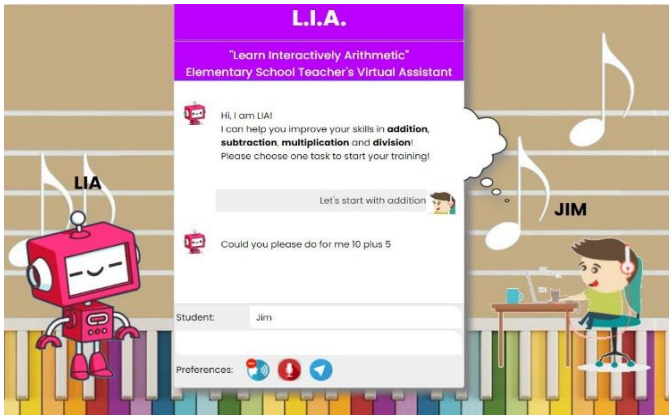

Fig. 2. Front-End UI. of the LIA.

On the right side is a student's image communicating with a computer. A moving robot is displayed on the left side. Between the student image and the moving robot lies the dialogue box. The robot is labelled as LIA. Simultaneously, as soon as the student gives his name, it appears as a tag in the student's image. With this design, we seek to identify the student with the corresponding image and create the student's perception that they are talking to a robot.

Dialogues can be text-based, but this is not a practical approach for elementary school students. For this reason, ASR and TTS interfaces controlled by voice (bottom left) and microphone icons are supported. These icons can be turned on or off at any time. If the voice icon is enabled, then the LIA sends its messages both in writing and orally. If disabled, only in writing. If the microphone icon is activated, every time the LIA expects a student's response, it automatically starts the microphone. The automatic activation is maintained for a short time or until the student completes his phrase. However, if the student does not speak and the microphone is turned off, the student can turn it on at any time by simply clicking on the microphone icon. In this way, a voice communication between the student and the LIA is achieved seamlessly.

VI. Pilot Study

During the quarantine due to Covid, we made the LIA available to 5 students to evaluate it. After training the students, we asked them to interact with the LIA until they solve all the quizzes that LIA will suggest within two days. Then, analyzing the log files, we found the following:

- Most students interacted with LIA extensively. Three students managed to answer all the quizzes and automatically received their evaluation. The fourth solved about the half quizzes, asked for his assessment and then said goodbye. The fifth student left the effort early, simply leaving the LIA session active.
- The main problem faced by the students was the reliability of voice recognition. Analyzing the students' phrases, we concluded that around 35% of phrases resulted from incorrect voice recognition at the beginning of the interaction. But during the interaction, the percentage dropped gradually to 5%. Students gradually learned to read the text produced by the voice recognition system, and displayed in LIA's dialogue interface and not to send it wrong but to repeat their phrase.

In a short interview that followed with each student, four of them stated that they found the LIA useful, and one stated that LIA couldn't help him. Also, everyone complained about the reliability of voice recognition. Finally, a student discovered that he could fool the LIA by taking advantage of the opportunity to ask questions. So, the student asked the question he had received and return the answer to LIA.

VII. Conclusions

We introduced LIA, a chatbot that helps first graders to practice basic arithmetic. In this work, we focused on the selection of appropriate technologies and architectural design. In our opinion, the proposed architecture can be used to design other closed-domain chatbots. The main weakness of our study is the limited experimentation. However, the initial results encourage us to continue.

Note that the more elements in the R-concepts of a Rule, the more phrases are recognized by the Rule. Therefore, an issue that constantly concerns us is the enrichment of the LIA knowledge base through the enrichment of R-concepts. In addition, we intend to add additional Rules, i.e. to increase the number of identifiable intentions. At the same time, we will support complex Rules that will logically combine the R-Concepts, e.g., [c (result) or c (number)]. Finally, we are planning to incorporate AI techniques to identify phrases that otherwise produce unidentified intents.

ACKNOWLEDGMENT

This work is partially supported by the MPhil program "Advanced Technologies in Informatics and Computers", hosted by the Department of Computer Science, International Hellenic University.

REFERENCES

[1] E. Adamopoulou and L. Moussiades, "Chatbots: History, technology, and applications," *Mach. Learn. Appl.*, vol. 2, p. 100006, Dec. 2020, doi: 10.1016/j.mlwa.2020.100006.

[2] F. Johannsen, S. Leist, D. Konadl, and M. Basche, "Comparison of Commercial Chatbot Solutions for Supporting Customer Interaction," *Res. Pap.*, Nov. 2018, [Online]. Available: https://aisel.aisnet.org/ecis2018_rp/158.

[3] S. Gupta, D. Borkar, C. D. Mello, and S. Patil, "An E-Commerce Website based Chatbot," 2015. https://ijcsit.com/docs/Volume%206/vol6issue02/ijcsit20150602125.pdf (accessed Jun. 06, 2021).

[4] T. Nt, "An e-business chatbot using AIML and LSA," presented at the Conference: 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Sep. 2016. doi: 10.1109/ICACCI.2016.7732476.

[5] R. Sankar, "EMPOWERING CHATBOTS WITH BUSINESS INTELLIGENCE BY BIG DATA INTEGRATION," *Int. J. Adv. Res. Comput. Sci.*, vol. 9, pp. 627–631, Feb. 2018, doi: 10.26483/ijarcs.v9i1.5398.

[6] "Domino's - Dom the Pizza Bot," *The Dots*, Jun. 06, 2021. https://the-dots.com/projects/domino-s-dom-the-pizza-bot-290990 (accessed Jun. 06, 2021).

[7] "Whole Foods Bot," *ChatbotGuide.org*, Jun. 06, 2021. https://www.chatbotguide.org/whole-foods-bot (accessed Jun. 06, 2021).

[8] "OneRemission," *KeenEthics*, Jun. 06, 2021. https://keenethics.com/project-one-remission (accessed Jun. 06, 2021).

[9] "Florence - Your health assistant," Jun. 08, 2021. https://www.florence.chat/ (accessed Jun. 08, 2021).

[10] "IBM Watson Health | AI Healthcare Solutions," *IBM Watson Health*, Apr. 30, 2021. https://www.ibm.com/watson-health (accessed Jun. 08, 2021).

[11] A. Palanica, P. Flaschner, A. Thommandram, M. Li, and Y. Fossat, "Physicians' Perceptions of Chatbots in Health Care: Cross-Sectional Web-Based Survey," *J. Med. Internet Res.*, vol. 21, no. 4, Art. no. 4, Apr. 2019, doi: 10.2196/12887.

[12] T. C. Lueth, T. Laengle, G. Herzog, E. Stopp, and U. Rembold, "KANTRA-human-machine interaction for intelligent robots using natural language," in *Proceedings of 1994 3rd IEEE International Workshop on Robot and Human Communication*, Jul. 1994, pp. 106–111. doi: 10.1109/ROMAN.1994.365947.

[13] S. Lauria, G. Bugmann, T. Kyriacou, J. Bos, and A. Klein, "Training personal robots using natural language instruction," *IEEE Intell. Syst.*, vol. 16, no. 5, Art. no. 5, Sep. 2001, doi: 10.1109/MIS.2001.956080.

[14] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein, "Mobile robot programming using natural language," *Robot. Auton. Syst.*, vol. 38, no. 3, Art. no. 3, Mar. 2002, doi: 10.1016/S0921-8890(02)00166-5.

[15] F. Colace, M. D. Santo, M. Lombardi, F. Pascale, A. Pietrosanto, and S. Lemma, "Chatbot for E-Learning: A Case of Study," *Int. J. Mech. Eng. Robot. Res.*, pp. 528–533, 2018, doi: 10.18178/ijmerr.7.5.528-533.

[16] H. T. Hien, P.-N. Cuong, L. N. H. Nam, H. L. T. K. Nhung, and L. D. Thang, "Intelligent Assistants in Higher-Education Environments: The FIT-EBot, a Chatbot for Administrative and Learning Support," in *Proceedings of the Ninth International Symposium on Information and Communication Technology*, New York, NY, USA, Dec. 2018, pp. 69–76. doi: 10.1145/3287921.3287937.

[17] "Get MathBot," *Microsoft Store*, Jun. 08, 2021. https://www.microsoft.com/en-us/p/mathbot/9p47wn62xcxf (accessed Jun. 08, 2021).

[18] "MathsBot.com - Tools for Maths Teachers," Jun. 08, 2021. https://mathsbot.com/ (accessed Jun. 08, 2021).

[19] A. K. Nayak, R. Patwari, and V. Subramanian, "MathBot – A Deep Learning based Elementary School Math Word Problem Solver," p. 7. [Online]. Available: https://www.semanticscholar.org/paper/MathBot-%E2%80%93-A-Deep-Learning-based-Elementary-School-Nayak-Patwari/c1f813eb7ef361949935f38985efd9bce45472cd#references

[20] C. Todorova *et al.*, "Visualizing mathematics with the MathBot: a constructionist activity to explore mathematical concepts through robotics," Aug. 2018.

[21] LaptrinhX, "Discord bot for mathematics," *LaptrinhX*, May 17, 2020. https://laptrinhx.com/discord-bot-for-mathematics-164616255/ (accessed Feb. 24, 2021).

[22] A. A. A. Ahmad and A. Singh, "Creating and Testing of Math Bot," p. 3.

[23] W. Cai *et al.*, "MathBot: A Personalized Conversational Agent for Learning Math," p. 13.

[24] K. Ramesh, S. Ravishankaran, A. Joshi, and K. Chandrasekaran, "A Survey of Design Techniques for Conversational Agents," in *Information, Communication and Computing Technology*, Singapore, 2017, pp. 336–350. doi: 10.1007/978-981-10-6544-6_31.

[25] M. das G. Bruno Marietto *et al.*, "Artificial Intelligence Markup Language: A Brief Tutorial," *Int. J. Comput. Sci. Eng. Surv.*, vol. 4, no. 3, Art. no. 3, Jun. 2013, doi: 10.5121/ijcses.2013.4301.

[26] D. Nadeau and S. Sekine, "A Survey of Named Entity Recognition and Classification," *Lingvisticae Investig.*, vol. 30, Aug. 2007, doi: 10.1075/li.30.1.03nad.

[27] "Dialogflow," Jun. 08, 2021. https://dialogflow.cloud.google.com/#/getStarted (accessed Jun. 08, 2021).

[28] "Wit.ai," Jun. 08, 2021. https://wit.ai/ (accessed Jun. 08, 2021).

[29] T. Bocklisch, J. Faulkner, N. Pawlowski, and A. Nichol, "Rasa: Open Source Language Understanding and Dialogue Management," *ArXiv*, 2017.

[30] "Botkit: Building Blocks for Building Bots," Jun. 08, 2021. https://botkit.ai/ (accessed Jun. 08, 2021).

[31] "About ChatterBot — ChatterBot 1.0.8 documentation," Jun. 08, 2021. https://chatterbot.readthedocs.io/en/stable/# (accessed Jun. 08, 2021).

[32] "Building a Chatbot: Analysis and Limitations of Modern Platforms - DZone AI," *dzone.com*, Feb. 28, 2021. https://dzone.com/articles/building-a-chatbot-analysis-amp-limitations-of-mod (accessed Feb. 28, 2021).

[33] T. Mikolov, I. Sutskever, K. Chen, G. s Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *Adv. Neural Inf. Process. Syst.*, vol. 26, Oct. 2013.

[34] P. Muangkammuen, N. Intiruk, and K. R. Saikaew, "Automated Thai-FAQ Chatbot using RNN-LSTM," in *2018 22nd International Computer Science and Engineering Conference (ICSEC)*, Nov. 2018, pp. 1–4. doi: 10.1109/ICSEC.2018.8712781.

[35] S. Arsovski, I. Muniru, and A. Cheok, *Analysis of the Chatbot Open Source Languages Aiml and Chatscript: A Review*. 2017.

[36] B. Wilcox, *ChatScript/ChatScript*. 2021. Accessed: Jun. 30, 2021. [Online]. Available: https://github.com/ChatScript/ChatScript/blob/5b5f2aa84c2a8697215db4c9a151e0d147a1f204/README.md

[37] "Web Speech API - Web APIs | MDN." https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API (accessed Jun. 30, 2021).

[38] "ResponsiveVoice Text To Speech," *ResponsiveVoice.JS Text to Speech*. https://responsivevoice.org/ (accessed Jun. 30, 2021).

[39] "PHP: What is PHP? - Manual." https://www.php.net/manual/en/intro-whatis.php (accessed Jun. 30, 2021).

[40] "An Introduction to JavaScript." https://javascript.info/intro (accessed Jun. 30, 2021).