A Knowledge-based Methodology for Building a Conversational Chatbot as an Intelligent Tutor

Conference Paper · September 2018

CITATIONS
2

READS
13,601

4 authors:

Xavier Sánchez Díaz
Norwegian University of Science and Technology
17 PUBLICATIONS **54** CITATIONS
SEE PROFILE

Gilberto Ayala-Bastidas
Tecnológico de Monterrey
8 PUBLICATIONS **27** CITATIONS
SEE PROFILE

Pedro Fonseca
Tecnológico de Monterrey
5 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Leonardo Garrido
Tecnológico de Monterrey
73 PUBLICATIONS **588** CITATIONS
SEE PROFILE

A Knowledge-based Methodology for Building a Conversational Chatbot as an Intelligent Tutor

Xavier Sánchez-Díaz, Gilberto Ayala-Bastidas, Pedro Fonseca-Ortiz, and Leonardo Garrido

Tecnológico de Monterrey
Departamento de Computación Regional Norte
Eugenio Garza Sada 2501 Sur, Col. Tecnológico 64849
Monterrey, Mexico
{A01170065,A00819406,A00805772,leonardo.garrido}@itesm.mx

Abstract Chatbots are intelligent agents with which users can hold conversations, usually via text or voice. In recent years, chatbots have become popular in businesses focused on client service. Despite an increasing interest for chatbots in education, clear information on how to design them as intelligent tutors has been scarce. This paper presents a formal methodology for designing and implementing a chatbot as an intelligent tutor for a university level course. The methodology is built upon first-order logic predicates which can be used in different commercially available tools, and focuses on two phases: knowledge abstraction and modeling, and conversation flow. As main result of this research, we propose mathematical definitions to model conversation elements, reasoning processes and conflict resolution to formalize the methodology and make it framework-independent.

Keywords: Chatbots, Knowledge Modeling, Methodology, Conversation Design, Intelligent Tutoring

1 Introduction

Chatbots are computer programs designed to hold conversations with users using natural language [15]. Some of them have human identities and personalities to make the conversation more natural. Ranging from Twitter bots with random responses to more complex counseling service agents, chatbots have become increasingly common in recent years. According to Tsvetkova et al. [16], nearly half of the online interactions between 2007 and 2015 involved a chatbot. They have been documented for use in a variety of contexts, including education [7] and commerce [3].

Chatbots have proven to be useful tools in academic courses, too. One example is the development of Jill Watson, an intelligent tutor developed by Goel et al [5] at Georgia Tech for an artificial intelligence MOOC. Anderson et al. [2] classify intelligent tutors as intelligent computer-aided instruction software which can respond to a student's specific problem-solving strategies. On the same note,

Reyes-González et al. [9] emphasize the importance of an individualized interactive process between the tutor and the student. The effectiveness of Jill Watson showed the potential of chatbots in a massive online class and is a fine example of the new educational era where artificial intelligence may play a major roll. In Goel's words: "it represents an educational technology for supporting learning at a big scale" [5]; sometimes so big that personalized attention from the instructor may result impossible. Using a conversational bot as an intelligent tutor has some other advantages like the fact that it can be available 24/7, giving the student the freedom of learning at their own pace, at any moment and from anywhere with Internet access. Having a chatbot also lightens the work load of the course instructor, as shown in [4].

With so many tools available to develop conversational agents nowadays, the building and deployment of a chatbot may look fairly simple. However, providing the chatbot with suitable information to be able to work as an educational tutor could be difficult. At the time of writing, information on how to design the tutor is scarce and scattered across blog entries and articles which are focused on the chatbot implementation rather than the design and modeling of knowledge. Many of the current development tools (such as Dialogflow and Chatfuel) only handle the implementation phase. The task of abstracting and organizing knowledge is up to the instructor designers. Thus, a methodology for knowledge abstraction and organization is essential, since having a conversation with an instructor is rather different than chatting with a sales agent.

This work proposes a methodology that formally defines and models the chatbot structure as an intelligent tutor. Its aim is to help multidisciplinary teams looking to design and implement chatbots in university courses. The methodology focuses on describing a first-order logic framework which can be implemented on different commercially available tools, and sheds some light on how to represent, expand and maintain the knowledge base.

The rest of the document is formatted as follows. Section 2 reviews human-computer conversation methodologies and available technologies. Section 3 formalizes the chatbots concepts and describes the proposed framework. Implementation and a case study are then discussed in Section 4. Finally, Section 5 presents concluding remarks.

2 A general approach to human-computer conversation

When talking about human-computer conversation, the very first technologies like ELIZA (1966) and ALICE (1995) come to mind due to their importance. ELIZA was created to demonstrate that natural conversation with a computer was possible, but failed to pass the Turing test since its implementation was based on string matching and no context was taken into account for the written responses. ALICE introduced the famous AIML (Artificial Intelligence Markup Language), which uses pattern matching rules to give 'meaning' to the actual words: topic, themes and categories were now taken into account [10].

Since then, chatbots have come a long way. Mobile devices now include simple chatbots which (or perhaps we should say whom) handle simple requests like making phone calls and setting alarms. Furthermore, automation tools have been developed for the creation of more complex chatbots and are now available for commercial use. The ever-increasing presence of machine learning in daily life has made the entire process of 'thinking' a bit more powerful than simple pattern matching, and perhaps the most common examples are those available directly from tech giants like Google, Facebook and Amazon. Despite each framework having its own implementation details and limitations, most tools derive from a general idea—receive raw data, give it meaning, and then act appropriately according to a knowledge base. Fig. 1 illustrates this process for a text-based chatbot.

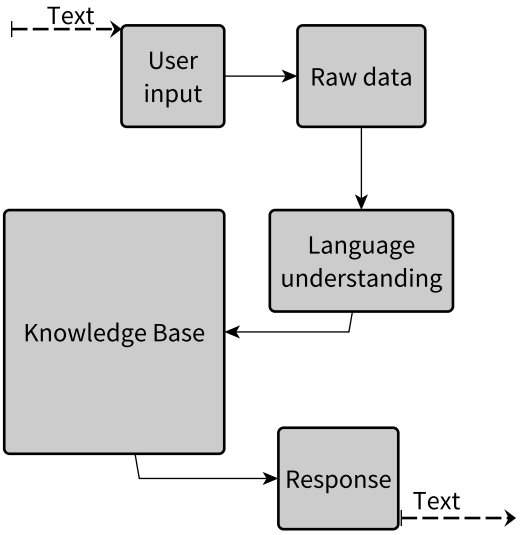

Figure 1. General methodology for a text-based conversational agent

To process natural language, chatbots rely on pattern recognition and bigram identification [14], a procedure which is usually handled by conversation frameworks. However, it is up to the chatbot designers to generate the knowledge base and provide the learning engine with appropriate examples.

3 Proposed Methodology

Our proposal considers two main phases:

Knowledge modeling
This phase determines how knowledge is represented and stored in the knowledge base.

Conversation flow
Both the lexicon used by the tutor and the order in which ideas are presented should be defined in this phase.

This section presents the formal definitions and foundations of the proposed methodology first. Later, each phase is described and contrasted with real-life queries. Finally, since implementation details vary across different conversation frameworks, they are not covered per se in the methodology. Nevertheless, conflict resolution and good implementation practices are broadly presented along a case study.

3.1 Formal definitions and foundations

A chatbot can be described as a conversational agent which gives an appropriate response when prompted with a known query. Formally speaking, a chatbot is a function f of the form $f: Q \to R$, which maps queries $q \in Q$ to responses $r \in R$.

In order to give the appropriate response, the query must be converted from natural language to a given entry in the knowledge base. This process consists on breaking down the user input, i.e. a sentence, to identify key concepts of the conversation. The most notable concepts at play are *entities* and *intents*.

An entity is an abstract object which holds relevance to the user. It can be thought of both as a subject or an object in a conventional sentence: *The quick brown fox jumps over the lazy dog* refers to a *quick brown fox* as a subject, and to the *lazy dog* as the object. Both the subject and the object are entities in this sense and can be grouped together into classes.

Intents, on the other hand, are abstract representations of the user's intentions. Since the user is asking a specific query, there must be something they want to do or know. However, in conventional questions the intent is not always present.

When a person asks, '*Can you tell me what time it is?*' their intention is to *find out the time*. The imperative sentence '*Show me my agenda*', for example, can be rewritten as '*Would you show me my agenda?*' which is a question in which the user wants *their agenda* (the entity) to be *shown* (the intent). Nevertheless, there are some other queries where the intention cannot be extracted from a rearrangement of their words. A user asking, '*Why do we snore?*' wants to know the *reason* of why we *snore* when we sleep. The intention of finding out the *reason* is not in the input text.

Queries as Functions of First-order Logic First-order logic is a branch of the study of reasoning, dealing with inference and 'belief' management using formulas in the form of predicates. It uses truth-functional connectives like $\neg$ (not), $\wedge$ (and), $\vee$ (or) and many others; along the use of functions to describe

the state of a variable which is either true or false [8]. In the context of chatbots, this truth value can be seen as the presence or the absence of a condition to trigger a response if it exists in the knowledge base. At first glance, intents may resemble verbs in a common sentence, but in fact they are relations of variables to a truth state: first-order logic functions. A query, then, has the form:

$$g^n(t_1, t_2, \ldots, t_n) \tag{1}$$

where $g^n \in G$ is an n-ary function symbol in the set of functions G, and t_i is a term in the set of terms T. The set of terms T is the set of known entities and G is the set of known intents in the knowledge base.

For example, in the statement `show(agenda)`, `show()` is a function which receives a single parameter to generate a response. The entity `agenda` is then shown to the user. Under this assumption, `show(flight)` could also be a possible query, when the user wants to know the status of a recently booked flight. If the user wanted to know the `time` right now, the function would look similar: `tell_time(now)`. Another example could be `tell_time(here)`, in case the bot is programmed to handle different time zones or countries. The query '*What's the time in New Zealand?*' could be translated to `tell_time(nz)`, or something along those lines. In the question '*What's the difference between x86 and x64 architectures?*' the intention is to know the `difference` between two entities, the *x86* and the *x64* processor instruction sets. The relation here is associating a pair of entities to a response, which is the difference between these two concepts. Following the function notation used throughout this work, it can be written as `difference(x86, x64)`. The `difference` function is binary. However, n-ary functions may be defined, as stated in Eq. 1.

It is important to consider that questions triggers are not stored in the knowledge base as text. Instead, a first-order predicate with the form defined in Eq. 1 is built for each query. First-order logic has limited expressivity [13] since some concepts which cannot be expressed using this formal system exist. Nevertheless, an adequate delimitation of the agent's reach can guide the knowledge modeling process, speeding up the implementation phase and reducing the training complexity.

3.2 Phase I: Knowledge modeling, extraction and storing

As described in Section 3.1, the sets of queries and responses are to be defined first, and there are several ways to perform that task. To deal with knowledge extraction and representation, for example, Huang et al. [6] use tuples of the form `<input, response>`, which are constructed by ranking the replies of a web forum thread as either 'fascinating', 'acceptable' or 'unsuitable'. Ales et al. [1], on the other hand, focus on automatic emotion detection of news headlines to give meaning to the input using self-organizing maps. Both ideas revolve around a medium-sized knowledge base, with a couple of concepts and queries.

However, a tutor designed for a college courses may deal with hundreds of different concepts and definitions. For this, the input of an expert is recommended,

especially if aiming for accurate pedagogical explanations using an adequate lexicon. The tutor language may be either casual and relaxed, or a bit more abstract and rigorous depending on the context. For example, in the field of mathematics—using abstract constructs and precise notations—the input from an expert is advised, since technology alone does not guarantee that students learn mathematics better than using only a regular textbook [12]. Moreover, problem situations can be represented in several ways, even in natural language. The use of an appropriate lexicon is important for students to incite them to translate everyday situations to mathematical models [11].

In order to comply with the trigger-response approach, knowledge should be separated into 'units', extracted from the expert and then laid down on a knowledge base with certain queries in mind for each knowledge unit. Each of these units represent a single query, a unique combination of functions and parameters that yields a certain response.

Although many data structures exist to store the knowledge base, most chatbot conversation structures are based on trees [15]. Each node in the tree represents a unique response, from a simple greeting to detailed information about previous queries. It is also important to note that in order for the conversation service to determine which response the user is looking for, the similarity between the user input and all known queries must be calculated. This process is usually done by machine learning algorithms using similarity measures between sentences in which each word or character may represent a single dimension, and its accuracy is refined by providing thousands of correctly labeled examples of user inputs. Therefore, it is advised to group knowledge units by similarity of the user input that will trigger them, rather than clustering by topic.

For instance, grouping the examples provided in Section 3.1 by intent, one can see that all three queries using the function `tell_time` have a similar input:

- ***What's*** *the* ***time*** *now?*
- ***What time is*** *it here?*
- ***What is*** *the* ***time*** *in New Zealand?*

The `tell_time` function input is somewhat different from that of the `show` function, in which the phrase *Show me* is predominant. Creating branches according to intents therefore reduces the complexity of the search.

3.3 Phase II: Conversation Flow

Once knowledge is separated into small atomic units, designing how to present them is the next step. An efficient way to do that is the creation of a glossary and a naming convention to keep track of the available queries and manage their trigger order. For instance, for the creation of the intelligent tutor for the introductory mathematics course in our institution, each knowledge unit in the tree was given a unique ID. The ID was generated automatically from abbreviations

of the names of the intents and the entities, with a hyphen separating the intent from the entities, and the entities separated by a plus sign: `def-N` was used to represent the definition of the natural numbers, corresponding to the question '*What is a natural number?*' or '*What is the definition of natural numbers?*'.

Some conversation frameworks allow entities to be grouped into categories, as it is the case of IBM Watson. The glossary, then, may contain entities clustered like the following:

- **Numbers**. Named sets of numbers, like *natural numbers*, *integers*, *rationals*, *reals*, etc.
- **Terms**. Mathematical terms related to the course material, e.g. *infinite decimal expansion* and *numerical representation*.
- **Equations**. Terms and notation specifically related to equations: *linear equations system* and *solution of a linear equation*.
- **Proofs**. Mathematical proofs that require a rigorous and a logical explanation, for example *proof that* $\sqrt{2}$ *is irrational* or *proof that the cardinalities of the naturals and the integers are equal*.
- **Modifiers**. Terms that modify the behavior of the bot. Usually used in conjunction of already defined entities, like *no solution [of something else]* and *positive [something]* or *negative [something]*.
- **Algebraic components**. Terms related directly with algebraic procedures, e.g. *substitute* or *x of t*, referring to the notation $x(t)$.
- **Wrong terms**. This category could be used to encapsulate frequently asked terms that are either mistaken or nonexistent. A fine example is the term *unreal numbers*: since the set of rational numbers and irrational numbers complement each other, unreal numbers *should* be a complement of the reals. This is obviously a wrong assumption.

Entities are then combined with intents to formulate a unique set of conditions that are needed to trigger their response, which were written by the expert considering the pedagogical aspect of the language employed in the answer. In this way, instructors can easily work along knowledge engineers to effectively model the queries and generate the knowledge base.

An example of an abstract representation of the knowledge base is presented Fig. 2, where queries are grouped according to their intents. Starting from the top of the tree (usually a greeting or a welcome message), the conversation framework will determine which branch contains the desired query and start iterating over the contents until it finds it or reaches the end. Chatbot designers are encouraged to include an error notice at the end of a branch in case the query is not found.

This approach works if the tutor is receiving queries in any order. However, more complex conversations can be modeled by branching each of the different intents into smaller pieces of conversation as needed.

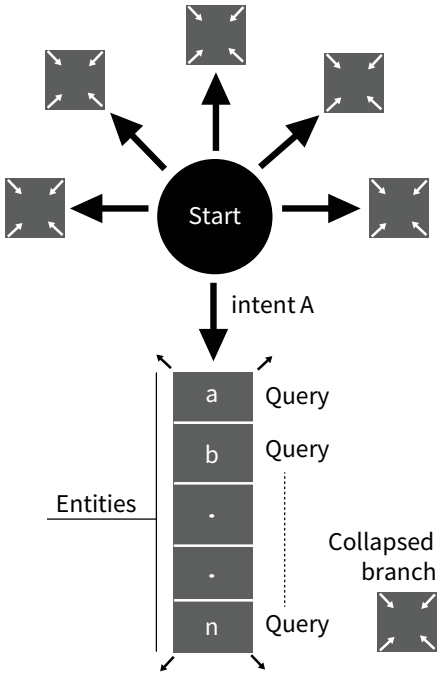

Figure 2. An abstract representation of the knowledge base built by grouping queries by intent.

Table 1. Glossary for the MA-1001 intelligent tutor.

ID	Intent	Entities
`def-N`	Definition	Natural Number
`ex-Q`	Example	Rational number
`notation-Z`	Notation	Integers
`expl-change+uniform`	Explanation	Uniform, Change
`def-obj_rest`	Definition	Object at rest
`ex-fin_dec_exp`	Example	Finite decimal expansion
`subset-Q+R`	Subset	Rational number, Real number
`card-Qc`	Cardinality	Irrational number
`comp_card-N+Q`	Compare cardinality	Natural number, Rational number

4 Implementation: a case study

Following the proposed methodology, two chatbots were designed in our institution: F-1001 and MA-1001, which are the introductory Physics and Mathematics courses respectively. The tutors were implemented in IBM Watson Conversation Service, using JSON structures for the tree, and providing examples and synonyms via comma separated value (CSV) files.

A small sample of the intents and entities of the intelligent tutor of the introductory math course is presented in Table 1.

Despite the fact that branching by intents guides the search process, the conversation flow may require adjustments due to technical reasons of the framework. It is important to consider how the knowledge base is traversed so that the chatbot answers are sound and complete.

In IBM Watson, the knowledge base is stored in a tree. This tree is usually traversed from top to bottom: taking as a starting point the first node (greeting), and then reviewing a branch if its condition is triggered. Then, each of the nodes in that branch is visited sequentially until one triggers: each branch of the tree is represented by an array. If no answer is found, then the default response is returned, which in the case of the tutors in our institution was a failure notice in the form of "*Sorry, I could not understand your request*". In reality, the knowledge base is a tree of arrays, in which each index of the array is a query.

This behavior may be prone to giving incorrect answers if the query nodes are not in the right order. Consider Examples 1 and 2. The former presents a formal definition of the problem and the latter rewords it into an specific use case. A visual representation of the conflict is also shown in Fig. 3

Example 1. Let the sequence of conditions $B = \langle a, a \wedge b \rangle$ be the array where the desired branch is stored, and the queries $q_1 = a \wedge b$ and $q_2 = a$ be the conditions detected by the chatbot. Query q_1 contains both condition a and condition b. Since the first element of B, B_1 is triggered if all its conditions are met, then q_1 will trigger B_1, when it is highly probable that the user wanted to get the response in B_2. However, q_2 will not trigger B_2 since condition b is missing and

not all conditions are met. On the contrary, q_2 will trigger B_1 as the requirements for its activation are all met.

Example 2. Assume a student wants to know the *definition* of *constant velocity*. Since the knowledge base also contains other topics on *velocity*, it would be better to model the query as a binary function, `def(velocity, constant)`, instead of using *constant velocity* as a single entity. However, since there is already a response for the *definition of velocity*, `def(velocity)`, there could be trouble handling the request if `def(velocity)` is found first.

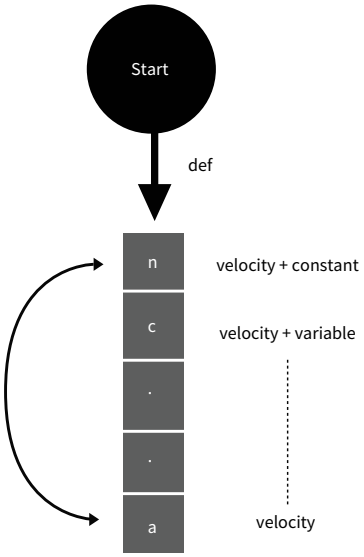

Figure 3. A visual representation of the conflict presented in Ex. 2, which is solved by placing more specific queries first.

To prevent unwanted responses to trigger if a condition is partially met, it is recommended to order the queries in the branch from more specific to less specific, so that a branch B ends up looking as $B = \langle a \wedge b, a \rangle$ and not the other way around. In Example 2, the problem can be solved if the *definition of constant velocity* is found first, and the more general query for the *definition of velocity* is presented last.

Another alternative is to add an additional level of branching and ask for extra information when dealing with similar situations. For instance, the tutor could explicitly ask the user *which of the following definitions on velocity are you interested in?*. However, this approach for conflict resolution is highly discour-

aged, as one may find different *topics* on `velocity` in the `definition` branch as well as in the `examples` branch.

5 Conclusions

Chatbots have become incredibly common nowadays, however there is few information on how to implement them as intelligent tutors for university courses. This work provides a formal methodology for organizing knowledge and arranging it for an appropriate flow to be implemented in commercially available conversation frameworks. The methodology presented focuses on using first-order logic predicates to represent knowledge units extracted from an expert as n-ary functions, which can be later grouped to simplify the search process. The paper also reviewed a case study of the creation of two intelligent tutors using the proposed methodology, which served as an example of conflict resolution when two queries share a trigger condition and are in the same level of the branch. This is just a first step towards formalizing conversational agents in education. Future work should consider modeling of more complex conversation scenarios, for example those requiring additional branching in the knowledge base. Incorporating automated methods for training and response refinement should also be considered. Is also important to develop a methodology to analyze student–chatbot interaction to ensure the chatbot qualifies as an intelligent tutor for higher education.

Acknowledgements

The authors would like to acknowledge the financial support of Writing Lab, TecLabs and Tecnológico de Monterrey, Mexico, for the production of this work.

References

1. Alès, Z., Duplessis, G.D., Şerban, O., Pauchet, A.: A methodology to design human-like embodied conversational agents. In: International Workshop on Human-Agent Interaction Design and Models (HAIDM'12). Valencia, Spain (2012), `https://hal.archives-ouvertes.fr/hal-00927488`
2. Anderson, J.R., Boyle, C.F., Reiser, B.J.: Intelligent tutoring systems. Science **228**(4698), 456–462 (1985). https://doi.org/10.1126/science.228.4698.456, `http://science.sciencemag.org/content/228/4698/456`
3. Angeli, A.D., Johnson, G.I., Coventry, L.: The unfriendly user: Exploring social reactions to chatterbots. In: Proceedings of International Conference on Affective Human Factor Design. pp. 467–474. Asean Academic Press (2001)
4. Dutta, D.: Developing an intelligent chat-bot tool to assist high school students for learning general knowledge subjects. Tech. rep., Georgia Institute of Technology (2017)
5. Goel, A.K., Polepeddi, L.: Jill Watson: A Virtual Teaching Assistant for Online Education. Tech. rep., Georgia Institute of Technology (2016)

6. Huang, J., Zhou, M., Yang, D.: Extracting chatbot knowledge from online discussion forums. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence. pp. 423–428. IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2007)
7. Jia, J.: CSIEC (computer simulator in educational communication): An intelligent web-based teaching system for foreign language learning. CoRR (2003), http://arxiv.org/abs/cs.CY/0312030
8. Makinson, D.: Sets, Logic and Maths for Computing. Springer-Verlag (2012)
9. Reyes-González, Y., Martínez-Sánchez, N., Díaz-Sardiñas, A., Patterson-Peña, M.: Conceptual clustering: a new approach to student modeling in intelligent tutoring systems. Revista Facultad de Ingeniería **0**(87), 70–76 (2018). https://doi.org/10.17533/udea.redin.n87a09, https://aprendeenlinea.udea.edu.co/revistas/index.php/ingenieria/article/view/327565
10. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River, NJ, USA, 3 edn. (2010)
11. Salinas, P., Alanís, J.A., Pulido, R., Santos, F., Escobedo, J.C., Garza, J.L.: Cálculo aplicado: competencias matemáticas a través de contextos (Tomo 1). México: Cengage Learning (2012)
12. Salinas, P., Quintero, E., Sánchez-Díaz, X.: Math and motion: A (coursera) mooc to rethink math assessment. In: Zaphiris, P., Ioannou, A. (eds.) Learning and Collaboration Technologies. pp. 313–324. Springer International Publishing, Cham (2015)
13. Serafini, L.: Expressive power of logical languages. Internet (July 2012), http://iaoa.org/isc2012/docs/expressivity.pdf
14. Setiaji, B., Wibowo, F.W.: Chatbot using a knowledge in database: Human-to-machine conversation modeling. In: 2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS). pp. 72–77. Thailand (January 2016). https://doi.org/10.1109/ISMS.2016.53
15. Shawar, B.A., Atwell, E.: Chatbots: Are they really useful? LDV Forum **22**, 29–49 (2007)
16. Tsvetkova, M., García-Gavilanes, R., Floridi, L., Yasseri, T.: Even good bots fight. PLos ONE **12(2)** (2017), https://arxiv.org/abs/1609.04285