A Chatbot Extended with Argumentation

Stefano Bistarelli[1], Francesco Santini[1] and Carlo Taticchi[1]

[1]*Department of Mathematics and Computer Science, University of Perugia*

Abstract

We present an argumentative chatbot, whose aim is to work in favor of (*pro*) or against (*con*) a given topic. Information is internally represented as a *Bipolar Argumentation Framework*, and semantics are computed with the purpose to select the argument to propose to the end-user. We use *DialogFlow* as the support framework to develop chatbots.

Keywords

Chatbot, Bipolar Argumentation, DialogFlow

1. Introduction

Chatbots (or simply bots) are software whose main aim is to simulate human conversations, usually in the form of an exchange of text messages. In recent years, this kind of application has made a great leap forward, also thanks to the significant advances in the field of Artificial Intelligence. Two are the most exploited technologies for implementing a chatbot: *Natural Language Processing* (*NLP*) and *Artificial Neural Networks* (*ANN*). NLP allows for automatically process written or spoken information, by operating lexical, grammatical, syntactic and semantic analyses. On the other hand, ANNs are trained on large amount of data to correctly classify information in input. Bots are widely used to streamline the communication process between the user and the offered services. Examples of real-world applications are represented by customer service, home banking, smart homes and management of IoT devices [1].

Although chatbots built starting from machine learning techniques are able to respond to requests in input, the user is not always willing to accept the provided answers, especially if they are not part of an explicit logical process which the user can follow and understand. In this work, we take a first step towards the development of a chatbot that, instead of exploiting algorithms based on matching and ANN data, uses logic deduction rules to generate its answers. In particular, we use Abstract Argumentation [2] to handle the information exchanged during the conversation and to allow our chatbot for simulating a debate between two people who either want to make a decision together (*pro* bot) or discuss to affirm their theses (*con* bot). The two people (or agents) enact an argumentation-based dialogue [3] in which knowledge can be exchanged, for instance, trough a concurrent argumentation language [4, 5, 6].

The paper is structured as follows: Sect. 2 illustrates components and features of the proposed chatbot; Sect. 3 reports some relate work; finally, Sect. 4 concludes the paper with final remarks.

AI^3*: 5th Workshop on Advances In Argumentation In Artificial Intelligence, Milano, Italy*

stefano.bistarelli@unipg.it (S. Bistarelli); francesco.santini@unipg.it (F. Santini); carlo.taticchi@unipg.it (C. Taticchi)

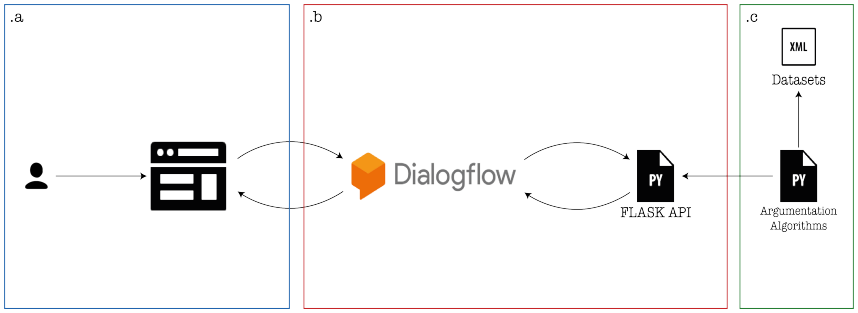

Figure 1: The three module in which the chatbot is structured.

2. Extending Chatbots with Argumentation

There exist several platforms that help the development of chatbots: some examples by the big tech players are Wit.ai[1], IBM Watson Assistant[2], Azure Bot Service[3], and Amazon Lex[4]. The framework we chose to develop our chatbot is DialogFlow[5], a chatbot and virtual assistant development-platform by Google. Each agent developed with DialogFlow is composed of one or more **intents** which manage the interaction with the user. Each intent consists of four basic elements: *i)* a set of **training phrases** that contains examples of what a user can (semantically) say and is used to activate individual intents; *ii)* **actions** which define the behaviour of the agent and trigger predetermined operations; *iii)* a set of **parameters** that can be extracted from user input to generate specific responses; *iv)* **responses** to be proposed to the user. Response messages, then, can be of two types.

We use *Bipolar Argumentation Frameworks* (*BAF*) [7], where a framework is defined by a triplet $\langle \mathcal{A}, \mathcal{R}_{def}, \mathcal{R}_{supp} \rangle$: $\mathcal{R}_{def}$ collects defeat relations, while $\mathcal{R}_{supp}$ collects support between two arguments. To compute extensions on BAFs we used Aspartix [8],[6] which returns the set of accepted arguments as, for example, $\{in(a), in(b)\}, \{in(a)\}$, etc. We describe the components of our chatbot with the help of Fig. 1, also providing insights of the implemented functionalities. The bot is divided into three modules: *i)* the user interface module, *ii)* the DialogFlow module, and *iii)* the module related to the algorithms we implemented.

The first module includes a web interface: the user interface, shown in Fig. 2, provides a direct interaction with the chatbot. Before starting the conversation, the user can adjust a number of settings as follows. First, a topic can be selected among those supported by the bot. Then, it is possible to choose the argumentation semantics that the bot will use to formulate its answers. Finally, the user decides whether the bot should produce responses in favour of her claims or against them. The conversation begins by entering a sentence in the text box panel. The chatbot replies with an answer, chosen according to the specified settings, after which the user can

[1]Wit.ai: https://wit.ai.
[2]IBM Watson Assistant: https://www.ibm.com/cloud/watson-assistant/.
[3]Azure Bot Service: https://azure.microsoft.com/en-us/services/bot-service/.
[4]Amazon Lex: https://aws.amazon.com/lex/.
[5]DialogFlow: https://cloud.google.com/dialogflow/docs/.
[6]Aspartix Webpage: https://www.dbai.tuwien.ac.at/proj/argumentation/systempage/.

enter a new argument. Each input is followed by a response and, with every interaction of this exchange between user and bot, the web interface also displays a representation of the BAF used to compute the answer. In this BAF, the argument matched to the user input is highlighted in green, while the argument corresponding to the bot output is highlighted in grey.

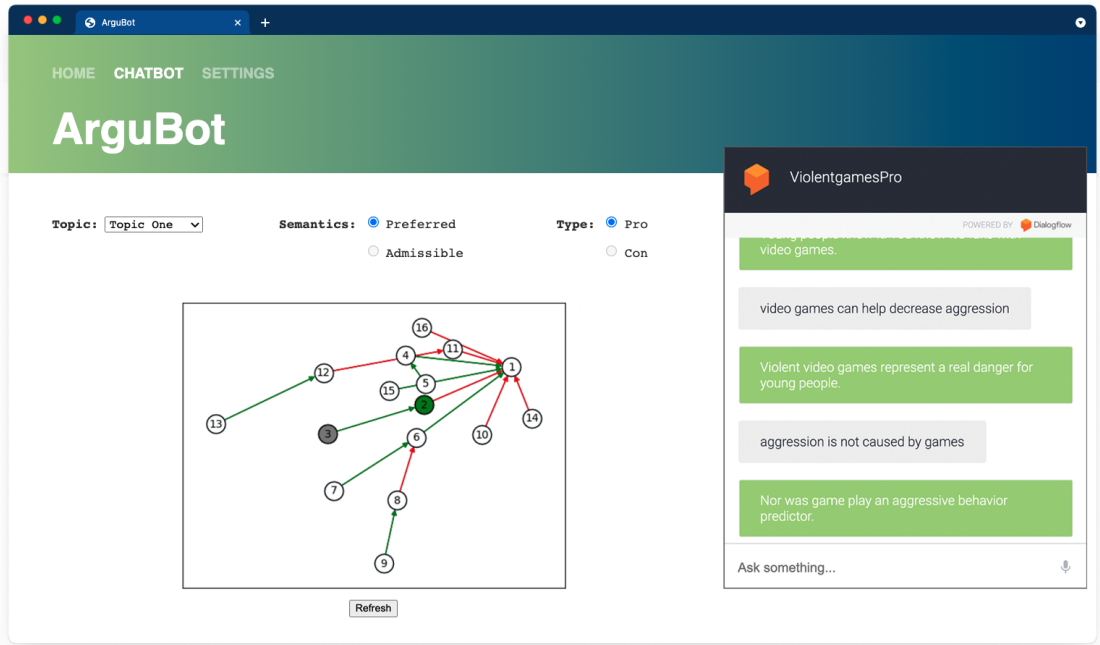

Figure 2: The Chatbot interface.

The second module of the chatbot, which in Fig. 1 is indicated by the central red box, is formed exclusively by DialogFlow. The Google development framework in our project is used only as a means to perform NLP operations. The third and last module of the bot includes all the algorithms we use to compute the extensions and the reinstatement labelling of the obtained AF. In Fig. 1 this module is represented by the green box on the right.

2.1. How the Chatbot Works

Now that we have described its components, we will see how the chatbot processes the sentences entered by a user and how to give the answers. Each time a user enters a sentence in the text area of our interface the system sends it to the DialogFlow module. This module takes the user's sentence and through the features of NLP performs the association between the sentence and an intent that forms the bot. An intent is the key element that makes up a chatbot developed with DialogFlow as it is able to categorise the sentences entered by a user. In this work we have created an intent for each argument that makes up a topic. Once the user's phrase is associated with an intent, DialogFlow sends a request to our API service. These APIs allow for obtaining an answer by extracting the most fitting argument from the underlying BAF. For this purpose,

two different procedures can be used, depending on whether the chatbot acts *pro* or *con* the topic.

In case the user is talking with the *pro* bot, the response argument is selected among those which are acceptable together with the user argument, according to the selected semantics. On the other hand, if the user is talking with the *con* bot, the response argument will be selected among those which attack the user argument. In both the cases described above, once this answer topic is selected, one of the reply sentences that is associated with the argument is selected. Such sentences are associated when the BAF chatbot is created. Finally, the API sends such sentence back to the request previously made by DialogFlow (Fig. 1b), which will return it to the screen as a response to the user.

3. Related Work

Most of related work [9, 10, 11] deals with persuasive argumentative chatbots designed by A. Hunter and co-authors. These software tools engage in persuasive dialogues, with the purpose to make the user change their current stance on a subject. To accomplish this, the chatbot needs a method to select persuasive counterarguments, or switch topic and present a new argument.

In [12] the authors develop a conversational agent able to explain issues related to global warming: the approach relies on textual entailment to identify the best answer for a statement conveyed by a human agent. The work in [13] adopts an argumentative-based chatbot to provide recommendations and explanations to help stroke patients in self-managing their condition, in collaboration with healthcare professionals. Same objectives are shared with the work in [14].

In [15] approach to dialogue management using chatbots in combination with social practices and argumentation theory. The use of social practice theory provides a way to contextualise the dialogue and thus providing all kinds of background information that can be subsequently used to interpret the input of the student and also to guide the conversation.

4. Conclusions and Future Work

In this paper we have described the whole procedure we followed in order to create a chatbot that uses solutions based on Abstract Argumentation as a method to generate answers to be given to the user interacting with the chatbot.

We started by studying the basic concepts of argumentation and applied them to our database, writing procedures for selecting topics as answers. Finally, we designed and implemented a web interface with which users can interact with the chatbot.

As future work we will consider the possibility to add arguments and relations (i.e., supports and attacks) dynamically to the argumentation graph directly during the conversation with the user. The current implementation is endowed with a centralised store that serves as a knowledge base for the agents. We would like to devise a model in which the store is, instead, distributed. Moreover, we plan to extend ConArgLib [16] to deal with support relations and integrate it with the chatbot, instead of relying on Aspartix. For this purpose, a meta-argumentation approach such as the one used in [17] could be considered.

References

[1] A. Shevat, Designing bots: Creating conversational experiences, O'Reilly Media, Inc., 2017.

[2] P. M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artif. Intell 77 (1995) 321–357.

[3] H. Prakken, Historical overview of formal argumentation, FLAP 4 (2017).

[4] S. Bistarelli, C. Taticchi, Towards an implementation of a concurrent language for argumentation, in: AIxIA 2020, volume 12414 of *LNCS*, Springer, 2020, pp. 154–171.

[5] S. Bistarelli, C. Taticchi, A concurrent language for argumentation, in: AI^3@AIxIA 2020, volume 2777 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020, pp. 75–89.

[6] S. Bistarelli, C. Taticchi, Introducing a tool for concurrent argumentation, in: JELIA 2021, volume 12678 of *LNCS*, Springer, 2021, pp. 18–24.

[7] C. Cayrol, M. Lagasquie-Schiex, On the acceptability of arguments in bipolar argumentation frameworks, in: ECSQARU 2005, volume 3571 of *LNCS*, Springer, 2005, pp. 378–389.

[8] W. Dvorák, S. A. Gaggl, A. Rapberger, J. P. Wallner, S. Woltran, The ASPARTIX system suite, in: COMMA 2020, volume 326 of *FAIA*, IOS Press, 2020, pp. 461–462.

[9] L. A. Chalaguine, A. Hunter, A persuasive chatbot using a crowd-sourced argument graph and concerns, in: COMMA 2020, volume 326 of *FAIA*, IOS Press, 2020, pp. 9–20.

[10] L. A. Chalaguine, A. Hunter, H. W. W. Potts, F. Hamilton, Impact of argument type and concerns in argumentation with a chatbot, in: ICTAI 2019, IEEE, 2019, pp. 1557–1562.

[11] E. Hadoux, A. Hunter, Comfort or safety? gathering and using the concerns of a participant for better persuasion, Argument Comput. 10 (2019) 113–147.

[12] D. Toniuc, A. Groza, Climebot: An argumentative agent for climate change, in: ICCP 2017, IEEE, 2017, pp. 63–70.

[13] N. Kökciyan, I. Sassoon, E. Sklar, S. Modgil, S. Parsons, Applying metalevel argumentation frameworks to support medical decision making, IEEE Intell. Syst. 36 (2021) 64–71.

[14] I. Sassoon, N. Kökciyan, E. Sklar, S. Parsons, Explainable argumentation for wellness consultation, in: EXTRAAMAS 2019, volume 11763 of *LNCS*, Springer, 2019, pp. 186–202.

[15] F. Dignum, F. Bex, Creating dialogues using argumentation and social practices, in: INSCI 2017, volume 10750 of *LNCS*, Springer, 2017, pp. 223–235.

[16] S. Bistarelli, F. Rossi, F. Santini, Conarglib: an argumentation library with support to search strategies and parallel search, J Exp Theor Artif Intell 0 (2020) 1–28.

[17] G. Alfano, A. Cohen, S. Gottifredi, S. Greco, F. Parisi, G. R. Simari, Dynamics in abstract argumentation frameworks with recursive attack and support relations, in: ECAI 2020, volume 325 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2020, pp. 577–584.