MCTK: a Multi-modal Conversational Troubleshooting Kit for supporting users in web applications

Giulio Antonio Abbo
Department of Electronics, Information and Bioengineering, Politecnico di Milano
Milan, Italy
giulioantonio.abbo@polimi.it

Pietro Crovari
Department of Electronics, Information and Bioengineering, Politecnico di Milano
Milan, Italy
pietro.crovari@polimi.it

Sara Pidò
Department of Electronics, Information and Bioengineering, Politecnico di Milano
Milan, Italy
sara.pido@polimi.it

Pietro Pinoli
Department of Electronics, Information and Bioengineering, Politecnico di Milano
Milan, Italy
pietro.pinoli@polimi.it

Franca Garzotto
Department of Electronics, Information and Bioengineering, Politecnico di Milano
Milan, Italy
franca.garzotto@polimi.it

ABSTRACT

Conversational Interfaces for user assistance are becoming persuasive. Today, though, most chatbots are not integrated into the application in which they are placed, but only superimposed, with no communication between the conversational and the graphical interface. We propose Multi-modal Conversational Troubleshooting Kit (MCTK), a Python package to easily integrate a conversational agent for troubleshooting in web applications. MCTK is multi-modal: once the system recognizes the problem the user is encountering, the textual solution in the chat is coupled with visual hints in the GUI. On top of that, MCTK is easy to configure and offers separation of concerns: dialogue designers can work on the conversation without the necessity of modifying the code, and vice versa.

CCS CONCEPTS

• **Human-centered computing** → **Natural language interfaces**; Web-based interaction; **Interaction techniques**.

ACM Reference Format:
Giulio Antonio Abbo, Pietro Crovari, Sara Pidò, Pietro Pinoli, and Franca Garzotto. 2022. MCTK: a Multi-modal Conversational Troubleshooting Kit for supporting users in web applications. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces (AVI 2022), June 6–10, 2022, Frascati, Rome, Italy*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3531073.3534480

1 INTRODUCTION AND BACKGROUND

In the last decades, user assistance – the problem-solving process that helps users of a specific system – has moved more and more from manual-based approaches to conversational troubleshooting methodologies, where the user is not required to read a handbook or a set of frequently asked questions and is instead actively helped by a system expert [5]. The automation of this procedure has led to the rise in popularity of conversational troubleshooting bots. Today, many tools for conversational problem solving are available on the market [2, 7]. These can be categorized into two families, according to the integration paradigm.

The first one comprises all plug-and-play platforms: adopters can configure their chatbots through a GUI and it is automatically deployed into a component inserted in the website. However, these tools are just superimposed on the application: the chatbot is not aware of what is happening on the website, and vice versa.

The second family includes all those conversational frameworks that expose an API for the programmers to interact with. However, these tools are built around the conversation: the implementation of the conversational interface and the design of the dialogue itself are intrinsically entangled; a domain or conversational expert cannot improve the system autonomously and is forced to ask a developer for support [9].

When the complexity of the tasks arises, an effective troubleshooting could benefit from multi-modal assistance, juxtaposing the textual messages with visual hints on the graphical user interface [4, 6, 8]. For this reason, we propose MCTK, Multi-modal Conversational Troubleshooting Kit, a prototype system that allows to easily design and integrate a conversational agent into a web application. When MCTK is integrated into a web application, the user can ask a question through the chat, and the system displays a textual solution in the chat together with visual hints in the graphical interface to guide the user.

To the best of our knowledge, MCTK is the first conversational troubleshooting toolkit designed to be multi-modal, extensible, and to follow the separation of concerns: the conversation design is detached from the implementation, meaning that a dialogue designer can change the behaviour of the system without interventions on the code, and, in the same way, interventions on the code are not linked with the behaviour's configuration.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
AVI 2022, June 6–10, 2022, Frascati, Rome, Italy
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9719-3/22/06.
https://doi.org/10.1145/3531073.3534480

Our work paves the ground for a new generation of conversational agents, able to actively support the user in their tasks, providing intuitive guidance even in complex settings.

2 MCTK:MULTI-MODAL CONVERSATIONAL TROUBLESHOOTING KIT

2.1 Design Requirements

MCTK is a framework to provide multi-modal conversational troubleshooting. Embedding MCTK into a website allows users to describe the issue they encountered to a chatbot, and receive multimodal suggestions – text messages in the chat and visual hints in the GUI – on how to solve it accordingly.

Contrarily to most used chatbot frameworks, we want MCTK to natively support *multi-modality*: if a user poses a question, the response should not only provide the answer but also point out graphically which elements of the graphical interface are related to the specific question [3].

We aim at *separation of concerns*: the deployment of MCTK and its configuration must be loosely coupled, such that the conversation designer can modify the chatbot without any programming knowledge required.

Finally, we require *extensibility*, both for the introduction of new types of issues and the inclusion of new interface items in the troubleshooting system.

2.2 How it works

We can describe the action of MCTK in four steps, as shown in Figure 1. (1) The system receives an issue description from the interface and the set of functionalities (modules) that are active on the screen. (2) An external natural language understating engine is trained with sample utterances to extract the problem type from the user sentences [1]. (3) This information is used to identify the problem that is causing the issue, and then retrieve the list of possible solutions. (4) These solutions are finally communicated to the users, through a sentence in the chat, that guides them through the resolution and some visual hints that suggest where to operate on the interface.

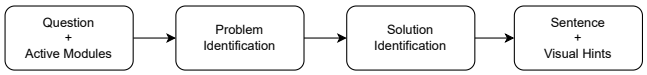

Figure 1: Operative workflow of MCTK

The system is built around the Configuration Table, a data structure representing connections between problems and solutions in a specific field of application; its structure is represented in Table 1.

Each column represents a problem type and each row is a parameter of a module; a module can span multiple lines, one for each of its parameters. If a specific parameter is linked to a problem type, the corresponding cell will contain an identifier corresponding to an utterance, contained in a separate table. This structure balances maintainability, relevant when reading and updating the table, with extensibility, which is important when expanding the table with new problem types and solutions.

MCTK is currently implemented as a Python package. To use it, it is sufficient to initialize it with the Configuration Table data.

Table 1: Structure of the Configuration Table.

Module	Parameter	problem1	problem2	problemN
moduleA	param1	utternace1	utterance2	
moduleA	param2	utternace3		utterance4
moduleB	param3			utterance5

Then, when provided with the user question and the context data, it returns the response and the information to update the graphical interface. The developer can display this information as preferred, or use some ready-to-use frontend components that speed up the integration.

A preliminary assessment with 3 developers and 3 conversation designers shows that both successfully used MCTK in autonomy. In addition, we extended a simple tool for clustering analysis with MCTK, to explore its troubleshooting ability, as shown in Figure 2. The chatbot was able to help users in improving their analysis for all 10 users who tried the application, even if they had never done a clustering analysis before. Users especially appreciated the multi-modality of the answers and the support received from the chatbot.

Figure 2: MCTK in action. When a user describes a problem, MCTK replies with a textual suggestion and and providing visual hints on the interface

3 CONCLUSION

We propose MCTK, a framework that allows multi-modal troubleshooting: guiding through the conversation and highlighting the relevant parts of the graphical interface. Users can describe their issues using natural language, MCTK identifies the underlying problem and proposes (textually and graphically) an executable solution.

MCTK aims at being easy to configure, maintain, and extend, as it is based on two simple tables that do not require specific knowledge to be edited. Further study will assess rigorously the effectiveness of the kit and its ease of use. Despite MCTK being still a prototype, we aim at distributing it as an open-source Python package to be easily integrated into any web application.

ACKNOWLEDGMENTS

This research is partially funded by EIT (European Institute of Technologies) under contract EIT Digital 22148 “Include”

REFERENCES

[1] Kate Acomb, Jonathan Bloom, Krishna Dayanidhi, Phillip Hunter, Peter Krogh, Esther Levin, and Roberto Pieraccini. 2007. Technical Support Dialog Systems: Issues, Problems, and Solutions. In *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies (NAACL-HLT-Dialog '07)*. Association for Computational Linguistics, USA, 25–31.

[2] Rodrigo Bavaresco, Diórgenes Silveira, Eduardo Reis, Jorge Barbosa, Rodrigo Righi, Cristiano Costa, Rodolfo Antunes, Marcio Gomes, Clauter Gatti, Mariangela Vanzin, Saint Clair Junior, Elton Silva, and Carlos Moreira. 2020. Conversational Agents in Business: A Systematic Literature Review and Future Research Directions. *Computer Science Review* 36 (May 2020), 100239. https://doi.org/10.1016/j.cosrev.2020.100239

[3] Abbas Mehrabi Boshrabadi and Reza Biria. 2014. The efficacy of multimodal vs. print-based texts for teaching reading comprehension skills to iranian high school third graders. 5 (Jan. 2014), 17.

[4] Pietro Crovari, Sara Pidó, Franca Garzotto, and Stefano Ceri. 2021. Show, Don't Tell. Reflections on the Design of Multi-modal Conversational Interfaces. In *Chatbot Research and Design (Lecture Notes in Computer Science)*, Asbjørn Følstad, Theo Araujo, Symeon Papadopoulos, Effie L.-C. Law, Ewa Luger, Morten Goodwin, and Petter Bae Brandtzaeg (Eds.). Springer International Publishing, Cham, 64–77. https://doi.org/10.1007/978-3-030-68288-0_5

[5] David Heckerman, John S Breese, and Koos Rommelse. 1995. Decision-theoretic troubleshooting. *Commun. ACM* 38, 3 (1995), 49–57.

[6] Dominic W. Massaro. 2004. A Framework for Evaluating Multimodal Integration by Humans and a Role for Embodied Conversational Agents. In *Proceedings of the 6th International Conference on Multimodal Interfaces - ICMI '04*. ACM Press, State College, PA, USA, 24. https://doi.org/10.1145/1027933.1027939

[7] Emi Moriuchi, V Myles Landers, Deborah Colton, and Neil Hair. 2021. Engagement with chatbots versus augmented reality interactive technology in e-commerce. *Journal of Strategic Marketing* 29, 5 (2021), 375–389.

[8] Sharon Oviatt, Rachel Coulston, and Rebecca Lunsford. 2004. When Do We Interact Multimodally? Cognitive Load and Multimodal Communication Patterns. *Proceedings of the 6th international conference on Multimodal interfaces* (2004), 8.

[9] Donya Rooein, Devis Bianchini, Francesco Leotta, Massimo Mecella, Paolo Paolini, and Barbara Pernici. 2021. aCHAT-WF: Generating Conversational Agents for Teaching Business Process Models. *Software and Systems Modeling* (Oct. 2021). https://doi.org/10.1007/s10270-021-00925-7