Integration of generative LLMs into the new generation of chatbots to enhance human-computer interaction

Guillermo Vicente-Oliva†
Departamento de Informática
Universidad de Valladolid
Valladolid, Spain
guillermo.vicente.oliva@uva.es

David Escudero-Mancebo
Departamento de Informática
Universidad de Valladolid
Valladolid, Spain
descuder@infor.uva.es

César González-Ferreras
Departamento de Informática
Universidad de Valladolid
Valladolid, Spain
cesargf@uva.es

Valentín Cardeñoso-Payo
Departamento de Informática
Universidad de Valladolid
Valladolid, Spain
valentin.cardenoso@uva.es

ABSTRACT

Most conventional chatbots rely on strategies that extract information from databases and use predefined templates to generate responses, which poses a significant limitation in maintaining natural, rich, and contextually adapted dialogues. This study examines the enhancement of chatbots through the integration of application programming interfaces (APIs) from large pretrained language models (LLMs), focusing particularly on the GPT architecture. First, the conventional architectural paradigm of chatbots is described, followed by a description of the integration of GPT-based components. As a proof of concept, this enhanced architecture is implemented in a controlled environment, evaluating coherence, contextual relevance, and adaptability. Results, based on user opinions, indicate a significant improvement in the quality of interactions with the enhanced chatbot compared to its conventional counterpart. In conclusion, the integration of LLM APIs, in this case GPT, represents a notable advancement in dialogue systems, offering more contextual and adaptive responses. This study anticipates a relevant leap in chatbot technology, suggesting a paradigm shift towards more humanized and effective human-computer interactions in the coming years.

1 Introduction

Chatbots are defined as software agents designed to simulate conversations with human users through textual or voice interfaces [10]. In the current state of information technologies, chatbots have found a niche market with various applications, ranging from customer service [16] to personal assistance [9] [4], political campaigns [12]. Despite their widespread use, conventional chatbots face significant limitations, particularly in their ability to maintain sufficiently rich and varied dialogues that adapt to the conversation context with a satisfactory level of naturalness [18]. Traditionally, these systems have relied on strategies to extract information from databases and use predefined templates, limiting their ability to adapt to the complexities of natural language and offer relevant and personalized responses.

Considering this fundamental limitation, there is a need to explore more advanced approaches that can navigate the high variety of human language with greater flexibility. This study presents some key insights to improve the functionality of chatbots through the integration of Language Model APIs, with a specific focus on the GPT (Generative Pretrained Transformer) architecture. The choice of GPT as the central axis is based on its demonstrated ability to understand and generate text dynamically, offering a more natural and fluid approach in automated interactions. The challenge lies in presenting how to include the API within already developed systems or those following the conventional structure of pre-established dialogues.

This article describes the implementation of a conversational system on the Dialogflow platform [6] using the capabilities of generative artificial intelligence, specifically GPT, to improve intent detection offered by the platform, as well as to offer other language generation functionalities. We explain how the integration of advanced technologies like GPT can overcome traditional barriers associated with chatbots, thus facilitating progress towards more natural, intuitive, and effective interactions. The timing of this article is opportune because GPT technology is still in its infancy, but its widespread adoption looms as a potential mean to enhance the quality and relevance of responses, potentially marking a step towards systems that can interact in a more human and adaptive manner.

The rest of the article is structured as follows: first, the research is contextualized in the state of the art on chatbots and the use of LLMs in service-oriented dialogue systems; then, the methodology adopted to integrate GPT into the chatbot architecture is presented, along with a description of the proof-of-concept implementation, and an evaluation of the results obtained. Through this analysis, we aim not only to highlight the effectiveness of integrating LLMs in improving chatbots but also to explore the implications of these advancements for the future of human-computer interaction.

2 State of the art

2.1 Chatbots creation technology

The creation of conversational agents that emulate human communication is at the root of computing motivations since Turing posed the question in the seminal article on artificial intelligence *Computer Machinery and Intelligence* [20], asking whether computers could communicate indistinguishably from humans. Since then, chatbots, as programs that communicate with humans or other chatbots giving the impression of not being automatons, have undergone significant evolution. The earliest systems, dating back to the 1960s with the pioneer ELIZA, relied on the pattern-response technique, where responses were generated through the identification of keywords and the application of pre-established rules to simulate human conversations. Despite its simplicity, ELIZA demonstrated machines' ability to emulate human interactions in limited contexts, laying the groundwork for future research in the field [22].

Over time, chatbot construction technology has evolved from rule-based systems to more complex models using natural language processing (NLP) and machine learning techniques. Despite these improvements and their application in dozens of commercial services [1] [25], conventional chatbots still have significant limitations, particularly in their ability to understand and process natural language effectively. Rule-based techniques and pattern recognition systems, although useful for applications with a limited and predictable scope, often fail in situations requiring deep contextual understanding or the management of unstructured dialogues. This inability to handle the ambiguity and variety of human language limits chatbots' effectiveness in providing coherent and contextual responses, resulting in frustrating user experiences in complex scenarios [24] [13] .

2.2 Natural language generation with GPT

The transition towards artificial intelligence-based systems and language models like GPT represents a significant advancement in overcoming chatbots' limitations. These models have demonstrated a remarkable ability to generate coherent responses that take into account the conversation context. To achieve this, GPT systems are based on pretrained models with a large amount of text, allowing them to capture a wide range of linguistic nuances [2][21]. This evolution has brought chatbots closer to truly natural interaction with users, marking a milestone in the development of more advanced and efficient automated dialogue systems.

GPT technology has proven to be highly effective in understanding and generating natural language. This technology is based on Transformers [15], which combine the encoder-decoder architecture [14] with the attention mechanism [21]. One of the most successful GPT products applied to language is chatGPT, which has continuously evolved since its first version launched in 2022, using a methodology of unsupervised pretraining followed by supervised fine-tuning. The use of models with millions of parameters, combined with specialized fine-tuning processes, has made the application an undeniable success [19].

The impact of GPT and transformer models in general on natural language generation is profound, offering more powerful and flexible tools for developers and data scientists [2]. These tools have enormous potential in improving chatbot programming and are poised to become a staple in dialogue system development kits. Despite this potential, the integration of GPT into conventional chatbot architectures is still under development.

3 Description of the software architecture

Figure 1 depicts the general architecture of a standard conversational system. The user generates a written or spoken expression (using the computer's microphone or phone), and the system interprets the input sequence by extracting the semantic content of the transaction, referred to as an intent or communicative intention. To extract semantic values, explicitly declared formal grammars can be employed, but nowadays, it is more common to train automatic systems using example phrases. When the system identifies the intention of the input expression, an event or series of events is triggered, each associated with a set of actions to be executed by the dialogue manager [17]. Taking the example phrase *I want to read a science fiction book of less than 250 pages*, the intent, or user's intention, would be to get a book recommendation, while the genre (*science fiction*) and length (*less than 250 pages*) would be semantic values.

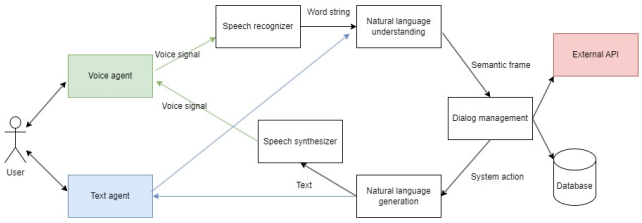

Figure 1: General solution architecture (inspired in [11]).

The extracted semantic value or values from the input expression can be used to assign values to parameters in the interaction context. These values are used as input parameters for actions to access a database that provides relevant variables for the interaction. In response, actions return values that are used by natural language generation systems to craft responses.

Until recently, the usual way to generate responses was to use templates where variable elements were replaced by the values returned by actions. Nowadays, automatic systems trained with labeled dialogue corpora are also commonly used. The logic of the interaction between the user and the system is generated from a detailed description of the intents and entities or slots of semantic frames [3]. In some cases, flowcharts are also used to plan and design the dialogue with the user.

The first applications of GPT trained on LLMs within dialogue systems are in generating responses using natural language [5]. These systems allow more flexible responses using the terms resulting from the actions invoked in the dialogue manager. When the GPT system is powerful enough, its API can be used to collect information that replaces or complements the information traditionally collected from the database. When the system is configured in chat mode, simulating a conversation with the user, it could be used directly to replace the dialogue manager. In this work, we explore the first of these options on a use case that will serve to assess the potential of its use.

4 Proof of concept

A system has been developed as a proof of concept that recommends book readings. The system provides users with reading recommendations using key terms such as the author, genre, number of pages in the book, year of publication, and the target audience: children, young adults, or adults. Additionally, given a title, the system can offer a synopsis of the book or additional information related to the key terms.

4.1 Prototype description

The prototype has been developed using Dialogflow, which is part of the Google Cloud platform and allows the design of conversational interfaces, facilitating their integration into various devices and applications. Specifically, the ES version is used, a simpler version suitable for designing conversational agents of medium complexity [6]. Figure 2 depicts the Dialogflow software architecture with its main components.

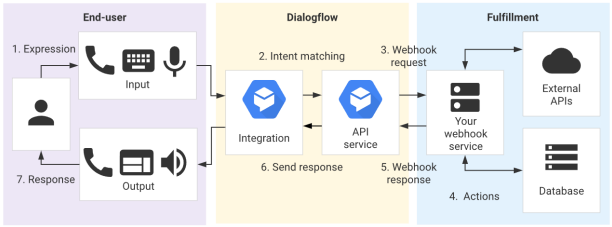

Figure 2: Dialogflow architecture [7]

To date, the Dialogflow ES version lacks functionalities to utilize LLMs, although these have already been integrated into its more advanced version, Dialogflow CX. Therefore, a development effort is required to access the capabilities offered by these generative models. In this proof of concept, we will adopt this approach to enhance the capabilities of intent detection and response generation provided by the platform. The general architecture of our system consists of two main elements: the Dialogflow platform and an API that allows responding to system requests, also known as a webhook service. This service can, in turn, call other APIs or include integration with databases, and in our case, it will handle the connection with the OpenAI API. This webhook API should expose a POST operation, which will be used in all calls made by Dialogflow to the service. This system leverages the capabilities of GPT to classify the user's intent and extract the terms present in their request, such as the book title, author, or desired genre for recommendations. Recommendation generation and synopsis are carried out through calls to GPT, and the information is stored in a MongoDB database to avoid unnecessary queries to the model. Additionally, the relationship between the Dialogflow session and the last referred book is kept in memory, allowing the user to request synopses or information about the book without needing to re-enter the title. This helps optimize resources and improve the user experience. The final response is generated from a series of predefined templates, which are complemented with information returned by GPT.

To obtain the intent through the LLM, it is necessary to create the desired intents in the Dialogflow platform. Dialogflow functions as an authoring tool that allows setting up the dialogue interactively by establishing intents and associated events. For each intent, it is necessary to define an event. This event will allow the webhook service to access the intent through it, via a fulfillment request response, for which it will be necessary to activate this option in the fallback intent offered by the system. In the example case, three intents have been defined, one for each of the functionalities offered by the system (GetInfo, GetSynopsis, and RecommendBook), as well as two additional intents to facilitate the management of the key terms that the user may use for obtaining recommendations. Additionally, six entities have been defined to model the parameters of the user's query.

After configuring all this, the code responsible for handling the intent is created. This code should accept the system request and extract both the current intent and the user request. Then, a message is sent to GPT requesting it to classify the user's intention into a series of categories, including an extra category for classifying requests that do not match the previously defined ones. Furthermore, GPT can be requested to extract relevant parameters from the user's request, preferably in a predefined JSON format to facilitate handling of the response (see figure 3). When the system is ready to recognize the intent, the process is triggered when a user makes a request. As there is no intent configured with training phrases, Dialogflow activates the fallback, which calls the webhook API via fulfillment. In the code, the current intent (in this case, fallback) and the user request are obtained, and a call to GPT (version 3.5 Turbo) is made to classify the user's intent. Once classified, an event response is sent to Dialogflow, where the event matches the user's intention, and all relevant parameters, if any, are included.

```
Your task is to classify the user's intent based on their message.
There are 3 possible intents: to obtain information, to get a synopsis, or to
get a recommendation about a book. You should return a response in JSON format,
with the following structure:
{
  "intent": INTENT
  "book: BOOK
  "filters: FILTERS
}
where INTENT will take one of these values:
 - information if the user wants to obtain information
 - synopsis if the user wants a synopsis or summary
 - recommendation if the user wants a recommendation
 - fallback if the intent is unclear or does not correspond to any of the above
BOOK will be the title of the book if included in the message. Otherwise, its
value will be an empty string.
FILTERS will only be considered if it is a recommendation and will follow this
structure:
{
  "genre": literary genre the user is interested in, empty if not specified
  "author": author the user is interested in
  "theme": theme the user is interested in
  "minPages": minimum number of pages
  "maxPages": maximum number of paged
  "lang": desired language, in ISO 639 format
  "langLevel": desired language level
  "similarBook": similar book the user is interested in
  "age": target audience age of the book
}
```

Figure 3: Intent classification prompt using GPT.

The response generation is carried out similarly to obtaining the intent, using fulfillment so that Dialogflow calls the webhook service, from which queries to GPT are made. These calls include some fields of interest, such as those specified below [8]:

- session: session identifier in Dialogflow.
- queryResult: this field includes the original user request (queryText) as well as the information about the different Dialogflow parameters (parameters).
- intent: intent information, as its name, accessible at displayName.

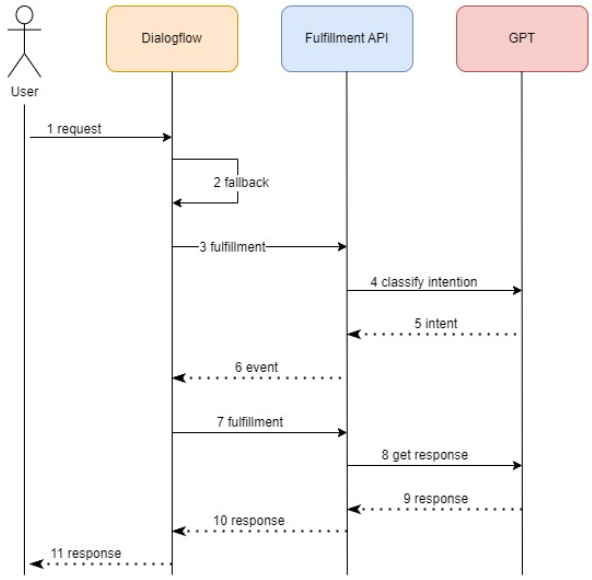

Figure 4: Sequence of obtaining a response. Figure 5 details the contents transmitted at each step.

When generating responses with GPT, it's possible to obtain complete responses directly to offer them to the end user, or parameterized responses, where the assistant is asked to respond using a specific structure, such as JSON. Then, this data is processed to craft the final response, allowing for greater control and avoiding unexpected responses that may affect the user experience.

Figure 4 shows the sequence of events that are part of the flow, from the moment when the user makes a request until the system generates a response. The flow begins with the user's request, for example*, I want to read a science fiction book of less than 250 pages* (1 in the figure). Dialogflow does not detect any intent (as no training phrases have been configured for any). The fallback intent is triggered (2). A fulfillment request is made for the fallback intent (3). The fulfillment API makes a request to GPT, asking it to classify the user's intent and extract the parameters from it (4). GPT responds with the detected intent and the parameters of the request (5). The API responds with an event response containing the intent and parameters obtained by GPT (6). Dialogflow makes a new fulfillment request, this time for the recognized intent, in the example case, "recommendation" (7). The API requests a response from GPT (8). GPT responds to the request (9). The API composes the final response from the JSON obtained through GPT and sends the result to Dialogflow (10). Dialogflow displays the response to

Figure 5: Information transmitted in the different steps represented in Figure 4. In black are the data sent, in blue is a description, the number refers to the identifier of the step in Figure 4.

the user (11). Figure 5 shows examples of the information transmitted in these steps.

4.2 System evaluation

Once the system is implemented, a usability study is conducted, in which five participants perform transactions in a controlled laboratory environment, in person. A series of tasks related to searching for reading recommendations are proposed, and observations related to user behavior are noted. At the end of the session, users are asked to fill out an evaluation survey regarding the tested system. The survey consists of six questions focusing on personal perception of ease of use, system speed, interaction comfort, and satisfaction with the obtained responses. In the questionnaire, users unanimously rate the system as easy to use, with quick interaction, although they also state that they did not feel like they were talking to a person. Four of the users were satisfied with the recommendations obtained, although only one found the information reported in the additional information sheets useful.

From the observations made during the usability tests, it can be highlighted that the system is capable of detecting the user's intent from relatively ambiguous phrases ("what is the book about"), but there are cases where the system cannot recognize the intention or recognizes it incorrectly. There is a high repetition in the recommendations. The system is able to complete the information structure in most cases, but it can also generate unexpected results.

5 Discussion and conclusions

The usual use of dialogue systems with commercial platforms like Dialogflow involves defining a set of expected training phrases to classify the user's intent within a set of predefined intents. This process requires domain knowledge that leads to the declaration of an appropriate set of phrases. User-defined entities must also be considered beforehand and limited to a finite list or regular expression, which can make their detection challenging in certain cases. The closed set of template phrases and limited entities restricts user expressiveness and leads to issues in service usage. In contrast, as demonstrated by the described system, the use of GPT provides flexibility in classifying user intent without the need for training phrases, simply by listing available options and describing desired entities in natural language.

The use of GPT not only facilitates the detection of intents and entities but can also be used to entirely or partially eliminate them. In this case, the responsibility for classifying intent and necessary parameters would fall on the model, which with the appropriate instructions can become a personalized assistant. Following the example of book recommendation, the implementation could be done by sending the user input directly to GPT and requesting it to simulate being a book recommendation system and respond to the user's request. Future work includes contrasting this option, but the tests conducted in the proof of concept indicate that this option may entail risks due to the lack of control over possible responses.

It is important to highlight that our approach goes beyond simply creating an interface for interacting with the large language model (LLM) of GPT. Unlike a generic implementation that could be limited to sending prompts to an LLM and returning its responses, as it would be facilitated by DialogFlow CX in its generative mode, our system is designed to extract and handle meaningful variables within the context of a dialogue. This capability enables a more functional integration with the dialogue process, enabling the precise identification of intents and entities, as well as the adaptation of LLM responses to the specific needs of the context. For example, the direct use of ChatGPT for acquiring products like for example sneakers (with high variety and multiple characteristics) might result in an inefficient and possibly sterile exchange, given the open nature of its responses. In contrast, our approach allows for defining the dialogue flow in a way that effectively uses the LLM to identify key intents and entities, ensuring a goal-oriented and relevant interaction. Therefore, our work should not be merely seen as the creation of a front-end but as the development of an integrated solution that harnesses the power of GPT to significantly enhance automated interaction, providing a smarter and more contextual framework for automated dialogue. Our proposal relies on intent classification and response retrieval, which can be independently implemented complementing the functionalities offered by the dialogue platform. Thus, GPT is employed to detect the intent and subsequently obtain responses through the usual flows offered by the platform.

While our study provides valuable insights into the potential enhancements offered by integrating ChatGPT into conventional chatbot architectures, we acknowledge the need for a more rigorous evaluation methodology, as highlighted by [23] and similar works. Metrics for evaluating chatbots, particularly those aimed at optimizing user experience, require careful consideration and adoption. Additionally, our evaluation solely with users was limited by a smaller sample size and lack of detailed user descriptions. As such, future work should aim to incorporate more sophisticated metrics and methodologies, including larger participant pools and clearer user descriptions, to provide a more comprehensive assessment of the proposed enhancements. Furthermore, conducting comparative tests directly with GPT to evaluate the effectiveness of the proposed approach would be beneficial and is an avenue for future research.

In conclusion, using an LLM-based system within a dialogue system not only anticipates improving the quality and relevance of responses generated by chatbots but also signals a paradigm shift in automated dialogue technology, marking the transition towards systems that can interact more humanly and adaptively.

ACKNOWLEDGMENTS

This work has been carried out within the framework of project PID2021-126315OB-I00 funded by MCIN / AEI / 10.13039/501100011033 / FEDER, EU.

REFERENCES

[1] Eleni Adamopoulou and Lefteris Moussiades. 2020. Chatbots: History, technology, and applications. *Machine Learning with Applications* 2, (2020), 100006.

[2] Maria Teresa Baldassarre, Danilo Caivano, Berenice Fernandez Nieto, Domenico Gigante, and Azzurra Ragone. 2023. The social impact of generative ai: An analysis on chatgpt. In *Proceedings of the 2023 ACM Conference on Information Technology for Social Good*, 2023. 363–373.

[3] Oscar Corcho and Asunción Gómez-Pérez. 2000. A roadmap to ontology specification languages. In *International Conference on Knowledge Engineering and Knowledge Management*, 2000. 80–96.

[4] Irina Dokukina and Julia Gumanova. 2020. The rise of chatbots – new personal assistants in foreign language learning. *Procedia Comput Sci* 169, (2020), 542–546. https://doi.org/https://doi.org/10.1016/j.procs.2020.02.212

[5] Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research* 61, (2018), 65–170.

[6] Google. 2024. Dialogflow docs. *https://cloud.google.com/dialogflow/es/docs*.

[7] Google. 2024. Fulfillment flow. *https://cloud.google.com/dialogflow/es/docs/images/fulfillment-flow.svg*.

[8] Google. 2024. Fulfillment webhook. *https://cloud.google.com/dialogflow/es/docs/fulfillment-webhook?hl=es-419#webhook_request*.

[9] Zhuoyan Han. 2023. The applications of chatbot. *Highlights in Science, Engineering and Technology* 57, (July 2023), 258–266. https://doi.org/10.54097/hset.v57i.10011

[10] Daniel Jurafsky and James Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*.

[11] Candace Kamm, Shrikanth Narayanan, Dawn Dutton, and Russell Ritenour. 1997. Evaluating spoken dialog systems for telecommunication services. In *Fifth European Conference on Speech Communication and Technology*, 1997. .

[12] Yunju Kim and Heejun Lee. 2023. The Rise of Chatbots in Political Campaigns: The Effects of Conversational Agents on Voting Intention. *Int J Hum Comput Interact* 39, 20 (2023), 3984–3995. https://doi.org/10.1080/10447318.2022.2108669

[13] Lorenz Cuno Klopfenstein, Saverio Delpriori, Silvia Malatini, and Alessandro Bogliolo. 2017. The rise of bots: A survey of conversational interfaces, patterns, and paradigms. In *Proceedings of the 2017 conference on designing interactive systems*, 2017. 555–565.

[14] Sascha Lange and Martin Riedmiller. 2010. Deep auto-encoder neural networks in reinforcement learning. In *The 2010 international joint conference on neural networks (IJCNN)*, 2010. 1–8.

[15] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2022. A survey of transformers. *AI Open* (2022).

[16] Dragoș Florentin Mariciuc and others. 2023. A Bibliometric Analysis of Publications on Customer Service Chatbots. *Management Dynamics in the Knowledge Economy* 11, 1 (2023), 48–62.

[17] M McTear, Z Callejas, and D Griol. 2016. The conversational interface: Talking to smart devices: Springer international publishing. *Doi: https://doi. org/10.1007/978-3-319-32967-3* (2016).

[18] Helly Raval. 2020. Limitations of existing chatbot with analytical survey to enhance the functionality using emerging technology. *International Journal of Research and Analytical Reviews (IJRAR)* 7, 2 (2020).

[19] Denis Rothman. 2021. *Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more*. Packt Publishing Ltd.

[20] Alan M Turing. 2009. *Computing machinery and intelligence*. Springer.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Adv Neural Inf Process Syst* 30, (2017).

[22] Joseph Weizenbaum. 1966. ELIZA—a computer program for the study of natural language communication between man and machine. *Commun ACM* 9, 1 (1966), 36–45.

[23] Oscar Jimenez Flores y Juan Jimenez Flores y Yoselin Gutiérrez Rojas y Víctor Jimenez Flores. 2018. MÉTRICAS DE EVALUACIÓN PARA CHATBOTS, ORIENTADAS A OPTIMIZAR LA EXPERIENCIA DE SU USO EN LAS REDES SOCIALES. *REVISTA CIENCIA Y TECNOLOGÍA - Para el Desarrollo - UJCM* 4, 0 (2018), 185–191. https://doi.org/10.37260/rctd.v4i0.134

[24] Shubin Yu, Ji (Jill) Xiong, and Hao Shen. 2024. The rise of chatbots: The effect of using chatbot agents on consumers' responses to request rejection. *Journal of Consumer Psychology* 34, 1 (2024), 35–48. https://doi.org/https://doi.org/10.1002/jcpy.1330

[25] Tomáš Zemčík. 2019. A Brief History of Chatbots. *DEStech Transactions on Computer Science and Engineering* (February 2019), 14–18. https://doi.org/10.12783/dtcse/aicae2019/31439