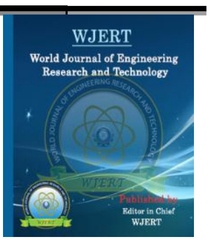
wjert, 2020, Vol. 6, Issue 6, 63-74. *Original Article* **ISSN 2454-695X**

World Journal of Engineering Research and Technology

WJERT

www.wjert.org **Impact Factor Value: 5.924**



MULTI-PLATFORM CHATBOT MODELING AND DEPLOYMENT WITH FRAMEWORK

Dr. M. Anusha* and Thamaraiselvi J.

[1]Assistant Professor, PG & Research Department of Computer Science, National College (Autonomous), Tamilnadu, India.

[2]M. Phil Scholar, PG & Research Department of Computer Science, National College (Autonomous), Tamilnadu, India.

Article Received on 24/08/2020 **Article Revised on 14/09/2020** **Article Accepted on 04/10/2020**

***Corresponding Author**
Dr. M. Anusha
Assistant Professor, PG & Research Department of Computer Science, National College (Autonomous), Tamilnadu, India.

ABSTRACT

Chatbots turns into a difficult undertaking that requires ability in an assortment of specialized spaces, going from characteristic language preparing to a profound comprehension of the APIs of the focused-on texting stages and outsider administrations to be incorporated. Chatbot (and voice bot) applications are progressively received in different spaces, for example, internet business or client administrations as an immediate correspondence channel among organizations and end- clients. Various systems have been created to facilitate their definition and arrangement. While these structures are productive to plan straightforward chatbot applications, they despite everything require propelled specialized information to characterize complex associations and are hard to advance alongside the organization needs (for example it is regularly difficult to change the NL motor supplier). What's more, the arrangement of a chatbot application for the most part requires a profound comprehension of the focused-on stages, particularly back-end associations, expanding the turn of events and support costs. In this paper, we present the Xatkit system. Xatkit handles these issues by giving a lot of Domain Specific Languages to characterize chatbots (and voice bots and bots by and large) in a stage autonomous way. Xatkit additionally accompanies a runtime motor that consequently sends the visit bot application and deals with the characterized discussion rationale over the foundation of decision. Xatkit measured engineering encourages the different development of any of its segments. Xatkit is open source and completely accessible on the web.

KEYWORDS: Modeling, Domain Specific Language (DSL), Chatot Design (CD), Chatbot Deployment (CD), Chatbot Voice (CV).

INTRODUCTION

Instant messaging platforms have been widely adopted as one of the main technologies to communicate and exchange information. The largest parts of them offer built-in maintain for integrating chatbot applications, which are automated conversational agents capable of interacting with users of the platform.[1] Chatbots have established functional in a variety of contexts to computerize tasks and get better the user experience, such as automated customer services, education and e-commerce. However, regardless of frequent platforms have recently emerged for creating chatbots their building and deployment remains a highly technical task. Chatbots are also increasingly used to facilitate software engineering activities like automating deployment tasks, assigning software bugs and issues, repairing build failures, scheduling tasks like sending reminders, integrating communication channels, or for customer support.[2] In this context, we explored the use of chatbots for domain modelling in previous work. Modelling chatbots can be embedded within social networks to support collaboration between different stakeholders in a natural way, and enable the active participation of non-technical stakeholders in model creation.

The extensive attention and command for chatbot request has emphasize the require to rapidly build multifaceted chatbots supporting NL processing (NLP), custom knowledge base definition, and complex action responses including external service composition.[3] However, the development of chatbots is challenging as it requires expertise in several technical domains, ranging from NLP to a deep understanding of the API of the targeted instant messaging platforms and third-party services to be integrated Intents are defined via training phrases. These phrases may include parameters of a certain type (e.g., numbers, days of the week, countries). The parameter types are called entities. Most platforms come with predefined sets of entities and permit defining new ones. Some platforms permit structuring the conversation as an expected flow of intents. For this purpose, a common mechanism is providing intents with a context that stores information gathered from phrase parameters, and whose values are required to trigger the intent. In addition, there is normally the possibility to have a fallback intent, to be used when the bot does not understand the user input.[4]

This work aims to tackle all these issues by raising the level of abstraction at what chatbots are defined. To this purpose, we introduce Xatkit, a narrative model-based chatbot development structure that aim to tackle this question using Model Driven Engineering (MDE) techniques: domain precise languages, display place self-regulating bot definitions, and runtime interpretation. Indeed, Xatkit embeds a dedicated chatbot-specific modeling language to specify user intentions, computable actions and callable services, combining them in rich conversation flows. Conversations can either be started by a user awakening Xatkit or by an external event that prompts a reaction from Xatkit (e.g. alerting a user that some event of interest _red on an external service the bot is subscribed to).[5] The resulting chatbot definition is independent of the intent recognition provider (which can be configured as part of the available Xatkit options) and frees the designer from the technical density of production with messaging and backend platform as Xatkit can be deployed through the Xatkit runtime constituent on them without the stage several additional steps. Xatkit is the effect of a collaboration work between the Open University of Catalonia and the Berger-Levrault company who is interested in adapting chatbots as part of its citizen portal service offering.[6]

Related Works

In,[7] Sara Pérez-Soler, Esther Guerra, Juan de Lara et al presents Modelling is worn in premature phases of software and organization expansion to argue and explore troubles, understand domains, evaluate alternatives and comprehend their implications. In these surroundings, reproduction is intrinsically collaborative as it involves stakeholders with different backgrounds and expertise, who cooperate to build solutions based on consensus. However, modeling tools typically provide unwieldy diagrammatic editors that may hamper the active involvement of domain experts and lack mechanisms to ease decision-making. In.[8] tackle these issues, we embed modeling within social networks, so that the interface for modeling is natural language which a chatbot interprets to derive an appropriate domain model. Social networks have spontaneous built-in conversation mechanisms, while the exercise of usual words lowers the admittance barrier to modeling for domain experts. Moreover, we facilitate the choice among modeling alternatives using soft consensus decision-making. This advance is support by our implement SOCIO, which mechanism on community networks like Telegram. In[9] Oscar Diaz, Felipe M. Villoria et al presents Blogs can be used as a conduit for customer opinions and, in so doing, building communities around products. We attempt to realize this vision by building blogs out of product catalogues. Unfortunately, the immaturity of blog engines makes this Endeavour risky. This paper presents a model-driven approach to face this drawback. This implies the introduction of (meta) models: the catalogue model, based on the typical Open Catalog Format, and blog representation, that elaborate on the employ of blogs as conduits for virtual communities. Blog models end up being realized through blog engines. Purposely, we center on two types of engines: a hosted blog stage and a standalone blog proposal, both in Blojsom. However, the necessitate of principles in a broad and continually developing blog-engine freedom hinders together the portability and the maintainability of the solution. Hence, we alternative to the idea of "abstract platform" as a technique to leave from the peculiarities of precise blog engines. In addition, the document events the recycle gains brought by MDE in comparison with the physical coding of blogs. This jeopardizes migration and reuse which, in turn, hinders the fulfillment of the cost-effective mandate. In[10] Davide Falessi, Natalia Juristo, Claes Wohlin, Burak Turhan, Jürgen Münch et al presents Controlled experiments are an important empirical method to generate and validate theories. Many software engineering experiments are conducted with students. It is often claimed that the use of students as participants in experiments comes at the cost of low external validity while using professionals does not. We believe a deeper understanding is needed on the external validity of software engineering experiments conducted with students or with professionals. We aim to gain insight about the pros and cons of using students and professionals in experiments. In[11] performed an unconventional, focus group approach and a follow-up survey. First, during a meeting at ISERN 2014, 65 experimental researchers, counting the seven authors, argued and discuss the employ of students in experiment with an open brain. Afterwards, we revisited the topic and elicited experts' opinions to foster discussions. Then we resulting 14 statements and request the ISERN attendees exclude the author, to offer their stage of conformity with the statements. Finally, we analyzed the researchers' opinions and used the findings to further discuss the statements. Our survey results showed that, in general, the respondents disagreed with us about the drawbacks of professionals. In[12] John Hutchinson, Jon Whittle, Mark Rouncefield et al presents to deal with the comparative absence of empirical studies of model driven engineering (MDE) in two diverse but complementary ways. First, we present an analysis of a huge online investigation of MDE deployment and experience that provides some rough quantitative measures of MDE practices in industry. Second, we supplement these figures with qualitative data obtained from some semi-structured, in-depth interviews with MDE practitioners, and, in particular, through describing the practices of four marketable organizations as they adopt a representation ambitious engineering approach to their software progress practices. In[13] in-depth semi-structured interviewing, we invited practitioners to replicate on their experiences and chosen four to employ as exemplars or case studies. In documenting some particulars of their attempts to deploy model driven practices, we recognize a number of factors, in exacting the importance of multifaceted organizational, managerial and social factors – as opposite to easy technological factors – that emerge to authority the comparative success, or failure, of the endeavor. In[14] David Kavaler, Sasha Sirovica, Vincent Hellendoorn, Raul Aranovich, Vladimir Filkov et al presents Modern software development is increasingly collaborative. Open Source Software is the bellwether; they support dynamic teams, with tools for code sharing, communication, and concern tracking. The success of an OSS project is dependent on team communication. E.g., in issue discussions, individuals rely on rhetoric to conflict their situation, but also maintain technological relevancy,[15] Rhetoric and scientific tongue are on opposite ends of a language complexity spectrum: the former is stylistically natural; the latter is terse and concise. Issue negotiations embody this duality, as developers employ rhetoric to exemplify technical issues. The approach combine in several discussions can define collection culture and influence performance, e.g., issue resolution times might be longer if conversation is imprecise. In,[16] GitHub, we studied question discussions to understand whether project-specific language differences exist, and to what extent users match to a language norm. We built project specific and overall GitHub language models to learn the consequence of perceived language complexity on manifold responses. We find that experienced users be traditional to project-specific language norms, accepted individuals use generally GitHub language rather than project-specific language, and conformance to project specific language norms diminish concern resolution times

Problem Definition

Chatbots are computer programs with a textual or voice interface, based on natural language. They are specifically designed to make user interaction as natural as possible, and have received extensive attention from academia and industry in recent years. Chatbots not only enable a faster and more natural way to access information, but they will become a key factor in the process of humanizing machines in the near future. Usability is defined as the degree to which a program can be used to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a specified context of use. Usability is a critical aspect in interactive software systems and so it is essential to incorporate usability in chatbots, to improve user experience. Chatbots are become pervasive and are used in many areas, such as bookings of all sorts of services, to obtain medical advice and for online shopping.

The numerous exercise and benefits of chatbots explicate their tough enlargement in terms of users, approval and saving resources. It is expected that the number of users will grow in the US by 23.1%. Although the market is still beginning to take shape it is estimated that the market size will expand massively. Many universities and commercial companies have put into use chatbots interacting with mature systems. At the commercial level, Face book messenger already has more than 300,000 chatbots in use. This makes downloading and installing new apps unnecessary, and the use of smart phones allows for personalization possibilities

Proposed System

This employment endeavor to attempt these issues by raising the altitude of abstraction at what chatbots are defined. To this attitude, we establish Xatkit, a narrative model-based chatbot improvement structure that aims to tackle this issue using Model Driven Engineering (MDE) techniques: domain specific languages, platform independent bot definitions, and runtime interpretation. Indeed, Xatkit embeds a dedicated chatbot-specific modeling language to specify user intentions, computable actions and callable services, combining them in rich conversation flows. Conversations can either be started by a user awakening Xatkit or by an external event that prompts a reaction from Xatkit (e.g. alerting a user that some event of interest fired on an external service the bot is subscribed to). The resulting chatbot definition[3] is independent of the intent recognition provider (which can be configured as part of the available Xatkit options) and frees the designer from the technical involvedness of production with messaging and backend platform as Xatkit can be organize through the Xatkit runtime component on them without performing any additional steps.

Chatbot

Chatbots are software agents that communicate with end-users via text-based conversations. Depending on the scenery of discussion, chatbots are separated into open- and closed-domain. Open-domain chatbots can contribute in a free form conversation with a user, having no specific goal defined. Closed-domain chatbots are building for serving a user to achieve precise objective. The approach describe in this dissertation is used to realize closed-domain chatbots. In distinguish to desktop, mobile and web applications, chatbots do not provide graphical user interface. Instead, users are communicating with chatbots through conversational user interface or integration with obtainable messaging applications.

Architecture Model

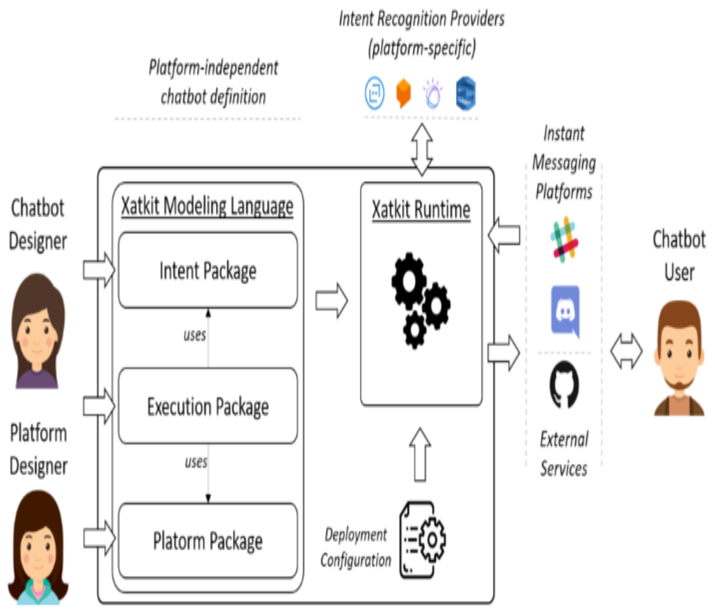


Figure Conversation using chatbot architecture.

Modules

- Import and load the data file
- Preprocess data
- Create training and testing data
- Build the model
- Predict the response

Import and Load The Data File

We introduce the essential packages for our chatbot and initialize the variables we will utilize in our scheme.

Preprocess Data

When working with text data, we need to perform various preprocessing on the data before we make a machine learning or a deep learning model. Tokenizing is the most basic and first thing you can do on text data. Tokenizing is the process of breaking the whole text into small parts like words. Here we iterate through the patterns and tokenize the sentence using nltk. word_tokenize function and append each word in the words list. We also create a list of classes for our tags. Now we will lemmatize each word and remove duplicate words from the list. Lemmatizing is the process of converting a word into its lemma form and then creating a pickle file to store the objects which we will use while predicting.

Create Training And Testing Data

Now, we will create the training data in which we will provide the input and the output. Our contribution will be the prototype and output will be the class our contribution pattern belongs to. But the computer doesn't understand text so we will convert text into numbers.

Build The Model

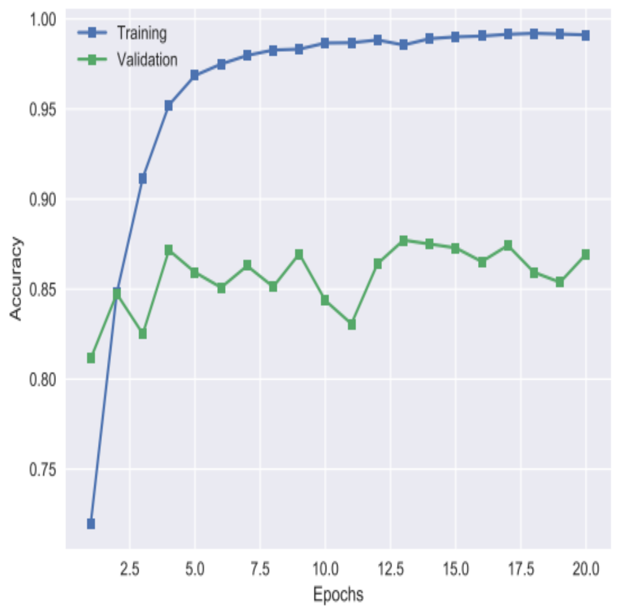
We have our training data ready, now we will build a deep neural network that has 3 layers. We use the Keras sequential API for this. After preparation the representation for 200 epochs, we achieved 100% accuracy on our reproduction.

Predict The Response (Graphical User Interface)

Now to predict the sentences and get a response from the user to let us create a new file 'chatapp.py'. We will weigh the trained model and then exercise a graphical user interface that will suppose the response from the bot. The model will only tell us the class it belongs to, so we will implement some functions which will identify the class and then retrieve us a random response from the list of responses. To predict the class, we will need to provide input in the same way as we did while training. So, we will generate some function that will execute text preprocessing and then predict the division. We will obtain the input message from the user and then employ the assistant function we have fashioned to acquire the response from the bot and display it on the GUI. Here is the full source code for the GUI.

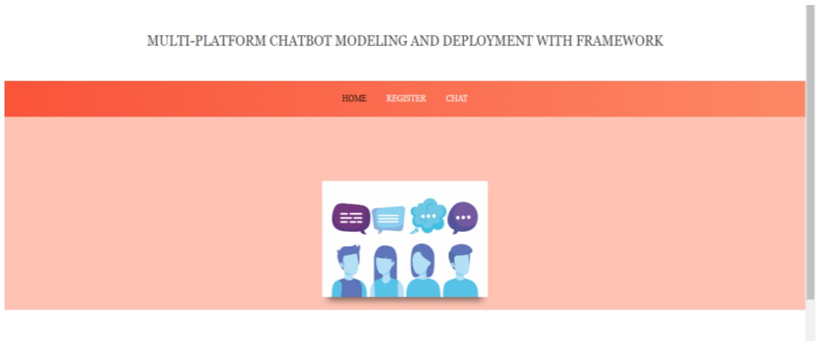
RESULT AND DISCUSSION

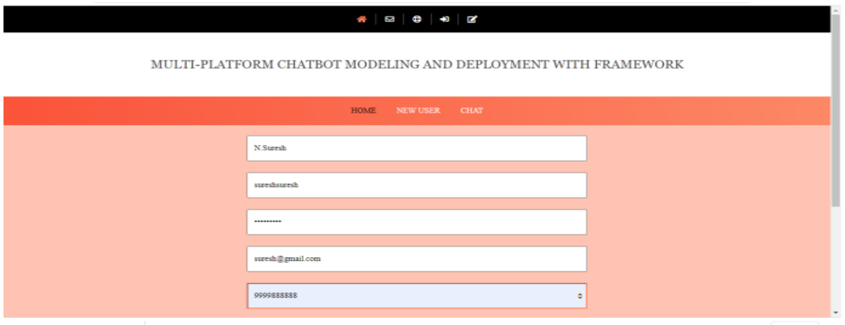
Traditional customer service often emphasizes users' informational needs; however, we found that over 40% of user requests on Twitter are emotional and they are not intended to seek specific information. This reveals a new paradigm of customer service interactions. One elucidation is that, evaluate with profession the 1-800 number or writing an email, social media significantly lower the charge of contribution and allows additional users to freely contribute to their experiences with brands. Also, sharing emotions with public is considered as one of the main motivations for using social media. Future study can scrutinize how touching requests are connected with users' inspiration in the context of social medium



Deep learning-based scheme accomplish comparable performance as human agents in conduct touching requests, which represent a important segment of user requirements on social media. This finding opens new possibilities for integrating chatbots with human agents to support customer service on social media. For example, an automated technique can be designed to separate emotional and informational requests, and thus emotional requests can be routed to deep learning chatbots. The response speed can be greatly improved. Deep learning outperformed IR in all the measures. This is principally since of deep learning, as a statistical-based approach is much enhanced at management unnoticed statistics and thus more elastic than keyword investigate approaches. For instance, given a reference reply to the request "my flight is delayed" and one to "my order is cancelled", a deep learning based system is able to generalize the reply in both scenarios and provide meaningful replies to unseen questions such as "my flight is cancelled", for which the most appropriate replies can hardly be retrieved from limited requests/topics available in the training data.

OUTPUT RESULT


MULTI-PLATFORM CHATBOT MODELING AND DEPLOYMENT WITH FRAMEWORK


MULTI-PLATFORM CHATBOT MODELING AND DEPLOYMENT WITH FRAMEWORK

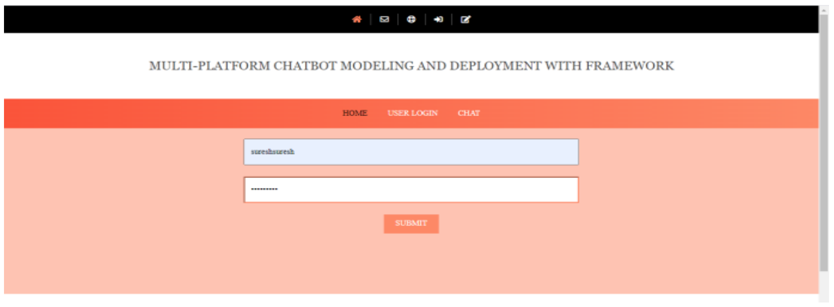
N.Suresh

sureshsuresh

----

suresh@gmail.com

9999888888


MULTI-PLATFORM CHATBOT MODELING AND DEPLOYMENT WITH FRAMEWORK

sureshsuresh

CONCLUSION

Xatkit, a multi-channel and multiplatform chatbot modeling framework. Xatkit proposes a set of domain-specific languages to decouple the chatbot definition from the technical details of the platform-specific aspects where the bot is going to be deployed. This increases the reusability of the chatbot and facilitates its redeployment when the needs of the company change, including the possibility of evolving the NLU engine used during the text analysis phase. Moreover, the runtime component can be easily extended to support additional platform-specific actions and events beyond those already shipped with the current version of Xatkit. For instance, some platforms like Alexa or Trello have been recently added by external contributors to the core Xatkit team. Xatkit is prepared to be used in real-case circumstances. But it has still plenty of room for improvements. At the language level we plan to improve the variability of the bot specification, moving towards a product-line approach that enables companies to create and quickly update several versions of the same bot (e.g. to create localized versions of the bot for each branch of the company). At the framework level, we plan to work on the integration of chatbot generators, able to create partial bot specifications from existing data sources within the company (e.g. FAQs or user guides). We also plan to study the combination of sentiment analysis and behavioral design patterns to create more likeable and effective chatbots.

REFERENCE

1. N. T. Thomas, ``An e-business chatbot using AIML and LSA," in Proc. Int. Conf. Adv. Computing, Commun. Informat. (ICACCI), Sep. 2016; 2740_2742.
2. Dr.M.Anusha "Feature selection using k-means genetic algorithm for multi-objective optimization" Procedia Computer Science, 2015; 1074-1080.
3. V. Subrahmanian, A. Azaria, S. Durst, V. Kagan, A. Galstyan, K. Lerman, L. Zhu, E. Ferrara, A. Flammini, and F. Menczer, ``The DARPA Twitter bot challenge," Computer, 2016; 49(6): 38_46.
4. G. Inc, The Road to Enterprise AI. Pune, Maharashtra: RAGE Frameworks, 2017.
5. P. Jackson and I. Moulinier, Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization, vol. 5. Amsterdam, The Netherlands: John Benjamins, 2007.
6. B. Nardi, S. Whittaker, and E. Bradner, ``Interaction and outeraction: Instant messaging in action," in Proc. 3rd CSCW Conf., 2000; 79_88.
7. R. Grinter and L. Palen, ``Instant messaging in teen life," in Proc. 5th CSCW Conf., 2002; 21_30.
8. L. C. Klopfenstein, S. Delpriori, S. Malatini, and A. Bogliolo, ``The rise of bots: A survey of conversational interfaces, patterns, and paradigms," in Proc. Conf. Designing Interact. Syst. (DIS), 2017; 555_565.
9. A. Xu, Z. Liu, Y. Guo, V. Sinha, and R. Akkiraju, ``A new chatbot for customer service on social media," in Proc. CHI Conf. Human Factors Comput. Syst. (CHI), 2017; 3506_3510.
10. A. Kerly, P. Hall, and S. Bull, ``Bringing chatbots into education: Towards natural language negotiation of open learner models," Knowl.-Based Syst., 2007; 20(2): 177_185.
11. Dr. P.S.S Akilashri, "Multi-Objective Optimization of Metal Parameters for Surface. Roughness and Response Analysis ", International Journal Of Modern Engineering Research (IJMER), ISSN: 2249-6645.
12. M. Brambilla, M. Dosmi, and P. Fraternali, ``Model-driven engineering of service orchestrations," in Proc. IEEE Congr. Services, Los Angeles, CA, USA, Jul. 2009; 562_569. doi: 10.1109/SERVICES-I.2009.94.
13. G. Daniel, J. Cabot, L. Deruelle, and M. Derras, ``Multi-platform chatbot modeling and deployment with the jarvis framework," in Advanced Information Systems Engineering (Lecture Notes in Computer Science), vol. 11483, P. Giorgini and B. Weber, Eds. Rome, Italy: Springer, Jun. 2019; 177_193. doi: 10.1007/978-3-030-21290-2_12.

14. J. Masche and N.-T. Le, ``A review of technologies for conversational systems," in Proc. 5th ICCSAMA Conf. Springer, 2017; 212_225. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319- 61911-8_19.
15. (2018). DialogFlow Website. [Online]. Available: https://dialog_ow.com/ [14] (2018). Watson Assistant Website. [Online]. Available: https://www.ibm. com/watson/ai-assistant/.
16. J.Pereira and O. Díaz, ``Chatbot dimensions that matter: Lessons from the trenches," in Proc. 18th ICWE Conf. Springer, 2018; 129_135. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319- 91662-0_9.