https://doi.org/10.2352/EI.2022.34.3.MOBMU-205

Chatbot Integrated with Machine Learning Deployed in the Cloud and Performance Evaluation

Ganesh Reddy Gunnam, Devasena Inupakutika, Rahul Mundlamuri, Sahak Kaghyan and David Akopian; Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, USA

Abstract

Recently human-machine digital assistants gained popularity and commonly used in question-and-answer applications and similar consumer-supporting domains. A class of more sophisticated digital assistants employing longer dialogs follow the trend, and there are several commercial platforms supporting their prototyping such as Google DialogFlow, Manychat, Chatfuel, Amazon Lex, etc. This paper explores cloud deployment of chatbots systems and their performance assessment methodologies. The performance measures includes system response delays and natural language processing capabilities. A case study platform supporting so-called deep-logic chatbots with long cycling capabilities is implemented and used for the assessment. To enable human-like conversations with a chatbot, huge training data, complex natural language understanding models are required and need to be adjusted and trained continuously. We explore implementation formats supporting auto scaling, and uninterrupted availability. In particular, we employ an architecture consisting of separate dialog management, authentication, and Natural Language Understanding (NLU) services. Finally, we present a performance evaluation of such loosely coupled chatbot system.

Introduction

A chatbot is an artificial intelligence (AI) software that can simulate a conversation with a user in natural language through messaging applications, websites, and mobile apps. A chatbot is often described as one of the most advanced and promising expressions of interaction between humans and machines [1].

However, from a technological point of view, a chatbot only represents the natural evolution of a question answering system leveraging natural language processing (NLP). Formulating responses to questions in natural language is one of the most typical examples of NLP applied in various enterprises' end-user applications [2]. Chatbots, also commonly known as dialog agents, receive requests from users either through speech (spoken language) or direct text input and output either a textual or vocal response (through speech synthesis) [3]. Lately, many industries and technology giants have been incorporating NLP into the chatbots for more human-like conversations, with the rise in artificial intelligence solutions. In this regard, the transformers [4] have become the to-go architecture for NLP, outperforming the convolutional and recurrent neural networks. As much as there has been an emergence in chatbot development frameworks [5], such as Dialogflow, ManyChat, Chatfuel, and Wit.ai, among others, so has been the rapid increase in NLU engines [6] by popular cloud service providers (Google, Amazon, Microsoft, to name a few). These NLU services classify user inputs into entities and intents.

Despite the presence of overall chatbot frameworks with some consisting of built-in NLU integrations such as Dialogflow and some facilitating the integration of external NLU services to their framework such as ManyChat, the challenges to deploy a functional integrated chatbot-NLU system in real-time exist. Some of these problems include: the methodology to establish communication between chatbot core (dialog management unit that drives the conversation flow and decides the subsequent steps (see figure 1), the decision whether to use popular NLU frameworks (whose underlying prediction models are unknown such as Google Dialogflow or Amazon Lex) or a bespoke NLU model, and finally, the performance aspects of aforementioned units of the chatbot system architecture.

Therefore, in this work, we initiate the investigation of the technical performance aspects (above mentioned third challenge) of the chatbot-NLU integrated architecture. Towards the realization of this goal, we developed a proof-of-concept rule-based chatbot service, and setup communication with a transformers-based NLU prediction service through the microservices architecture [7]. Microservice architecture bots are known to provide flexible conversational interface to extract the service information, service API documentation, building and testing results, the health status, service usage analysis, and service dependency graphs by connecting multiple tools [8]. Furthermore, to optimize the performance and measure response times, cloud services are utilized and compared with the on-premises implementation. Specifically, we evaluated the performance of a microservices-based chatbot application with NLU along with Facebook messenger as a communication channel. The communication response times and NLU prediction accuracy are reported.

The remainder of this paper is organized as follows. Section 2 covers background of exisiting NLP services for chatbots, highlights the performance challenges faced by the chatbot-NLU integration in the microservices architecture, and briefly covers the aspects we address in this paper. In Section 3 we provide NLP integration methodology for a rule-based chatbot core service. In Section 4 we provide the two experimental settings covering the communication with NLU service with and without Facebook Messenger channel. In Section 5 we present the performance results and provide concluding remarks in section 6.

Background

This section first covers existing NLP services available for easier integration with chatbot and enable more human-like conversations followed by the existing performance challenges in microservices-based NLP integration.

NLP services for chatbots

Many services exist that provide building blocks to realize an intelligent conversational agent with NLP capabilities. Some of these services include IBM Watson [9], Wit.AI [10], and Dialogflow [11]. These services enable NLP integration in chatbots by providing built-in cloud-based cognitive services and natural language interfaces for applications with the capabilities to convert user's natural text into structured data. Such NLP services let the developers integrate to different social media channels such as FB Messenger, WhatsApp, Slack, among others with the token. It is however the developer's responsibility to handle the coordination between chatbot interface, integration between the chatbot core and third-party services, and additional overhead of the overall integration's maintenance, scalability, and extensibility. To that end, the microservices-based architecture for system design becomes a solution that provides the flexibility of integrating heterogeneous services with less overhead of communication and complexity as well as loose coupling of different functionality (such as in this chatbot-NLU integration work) to avoid single point of failure. Microservices are small, autonomous services that work together and can be executed independently.

Performance challenges in NLP integration

Implementing microservice-based applications in containerized [12] cloud is motivated by easy deployment, fast migration, and higher scalability resulting in lowering infrastructure and maintenance costs. All the services in microservice architecture are independent from each other and the only form of communication between services is through their published interfaces such as endpoints [13]. Although microservices are supposed to be faster, they add a layer of complexity that poses considerable performance problems. The major ones include: (1) the choice between synchronous and asynchronous calls to involved services. The asynchronous requests allow for more concurrent work within individual services, and more efficient requests for the integrated application. However, it is important that the receiving service can fulfill such asynchronous requests within a decent time window, and scale to accommodate the continuous load of requests [6]. (2) Handling third-party requests: Although microservices are communicating with one another efficiently, sometimes the limitations of a third-party service or API can cause significant issues for an application. With the increasing growth of third-party services and APIs within applications, to avoid application failure and for the continued service of these services and APIs, appropriate measures must be taken into consideration for integrated services. Owing to the aforementioned challenges, in this paper we present the performance assessment of chatbot-NLU services integration that consists of third-party services in the architecture. This work leveraged and integrated best functionalities of the available frameworks (such as [14]) for realization of human-like chatbot with NLU integration. Further, performance is measured through system response delay and NLU service prediction performance.

NLP Integration Methodology

This section presents NLP integration methodology along with the integrated system architecture, covers the experimental setup for the collection of response time and model prediction accuracy as performance metrics. Figure 1 presents the overall system architecture consisting of a chatbot integrated with NLU service. The figure shows the communication sequence in the integrated system. Each service in the architecture communicates via corresponding APIs. The chatbot-user communication flow is as follows:

- The front-end user interface constitutes of messaging channels such as SMS (Twilio), Facebook Messenger, WhatsApp, among others. These channels are used by an end-user to communicate with the chatbot and its backend.

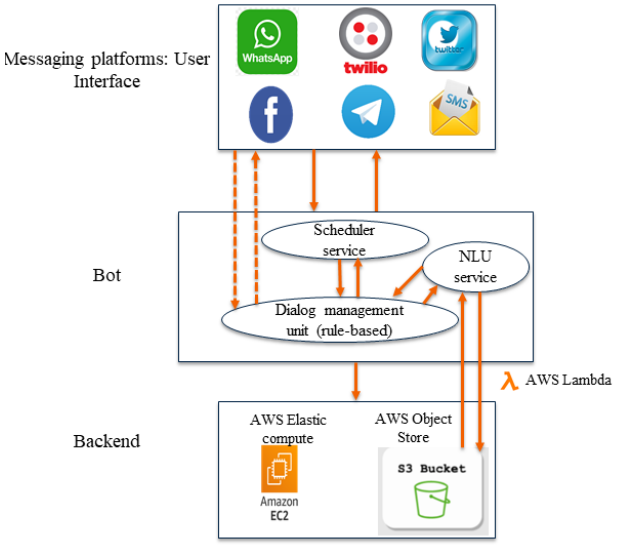

Figure 1. *Chatbot-NLU Integrated System Architecture.*

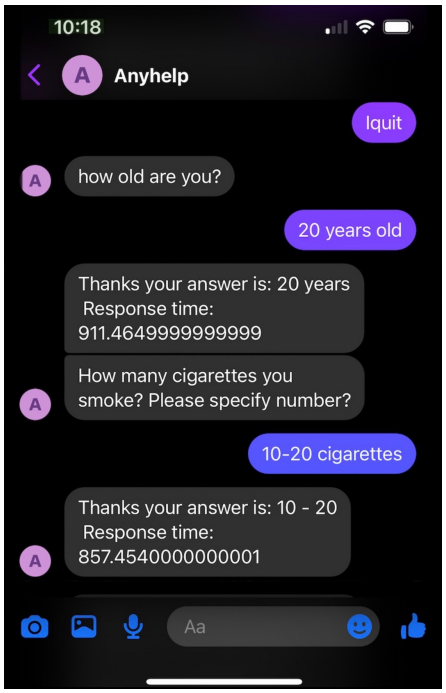

Figure 2. *Chatbot Communication via Facebook Messenger channel with NLU.*

- Second is the bot layer consisting of scheduler service that periodically checks for pending or timed messages ready to be sent, core dialog management unit that performs rule-based decisions on user's input, and a natural language understanding (NLU) service that handles user's free-text necessary to take appropriate action or response. The core unit asynchronously processes every incoming message, communicates with the NLU service when needed for advanced processing or pre-processing of free-text so as to be recognized by the rule-based processor in core and take correct actions.
- The integrated system is built on Amazon Web Services backend. The core unit is deployed on the elastic cloud compute instance and the NLU utilized AWS object store S3 for trained models to communicate via AWS Lambda functions.

Experimental Setting and Performance metrics

In this paper, we have used two types of experimental setups deatils of which are explained in the following sections.

1. **NLU Model as a service**
   We deployed NLU trained model in the cloud and exposed it as application programming interface (API) endpoint. This API will take questions, evaluate them based on the context, and gives the appropriate answer back. We utilize a customized script to send API requests to the NLU service. A pre-trained language model based on the Bidirectional encoder representation for transformers (BERT) is uploaded as an Amazon Web Services (AWS) S3 object.

2. **Facebook and NLU communication**
   In this setup, we have integrated our NLU service with Chatbot which is deployed in the cloud. Additionally, we have created a Facebook page and enabled the messenger. By using webhook, we have connected this Facebook page with our NLU integrated chatbot. So any user who communicates with our chatbot using this page is able to receive a response from the NLU service if the the message is not recognized by the chatbot. Figure 2 shows an example conversation and a rule-based chatbot communication with the NLU service.

As for the performance metrics, we utilize average response times for NLU communication and prediction performance accuracy for NLU model.

Response Time

Response Time is defined as the time taken to receive response for a given request. In the first evaluation regarding direct communication with NLU service, we took response time as the time taken to get response from the service. We took an average of 25 sent requests for consistency. The average response is calculated as:

$$Average\ Response\ Time = \frac{Sum\ of\ All\ Response\ Times}{Number\ of\ Iterations} \quad (1)$$

For the second evaluation, we sent the message to chatbot from the Facebook (FB) messenger. The messenger sends request to NLU service and gets the response back to the user. In this case, the response time is the summation of NLU service response time and the Facebook messenger delay. Furthermore, we repeated the experiment 25 times and calculated the average response time.

NLP Prediction Accuracy: F-score

We use F-score as the accuracy of a model on a dataset. It is the harmonic mean of model's precision and recall where precision is the fraction of relevant examples out of all retrieved examples and recall is the fraction of retrieved examples among all relevant examples. A standard F1-score (F-score of 1) is expressed as:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (2)$$

Results

This section presents the performance measures obtained from the chatbot-NLU communication under the two experimental settings as described in the experiment setting section.

NLU Response Times

As mentioned before, we experimented two different scenarios. For the first scenario, we calculated the NLU response time without any messaging platform. Figure 3 shows the response times over 25 communication instances. The highest response time was 869 milliseconds and the lowest was 694 milliseconds. Thus, the average response time was approximately 760 milliseconds. Whereas, for the second experiment that includes the messenger platform (Facebook), the highest time was observed to be 1038 milliseconds and 747 milliseconds was the lowest with 897 milliseconds average response time. From the above two calculations, we observed that the average delay caused by the Facebook messenger is around 137 milliseconds which seems reasonable. Figure 2 shows the chatbot communication using Facebook messenger and the response times of each message exchange.

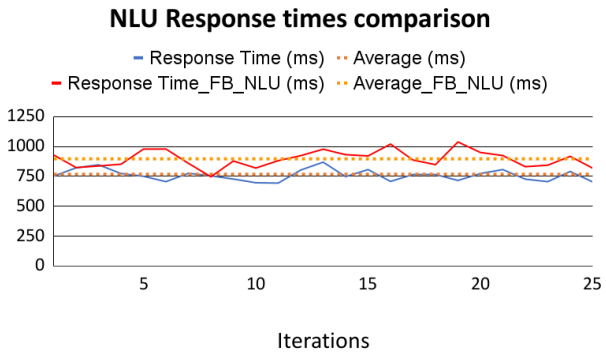

Figure 3. *NLU Response times comparison.*

NLU Prediction Accuracy: F-score

Given the microservices-based NLU integration with the chatbot, the average response times are dependent on the prediction from the NLU model. Thus, the correctness of predictions is important for proper actions from the chatbot service to user's requests. We evaluate the correctness of the NLU service prediction based on the accuracy (F1-score) of the trained model itself. Table 1 shows the F1-score of the NLU model with average response times of the NLU service. The pre-trained BERT model was fine-tuned on the evaluation dataset that consists of keywords and examples from the intervention protocol [15] provided for the

case-study chatbot from healthcare professionals. F1-score is then computed over the individual words in the prediction against the ones in the true answer. Thus, the basis of F1-score is the number of shared words between the prediction and the truth. We achieved an F1-score of 0.94 for the fine-tuned model that was uploaded to the S3 object store and released as a prediction service.

Table 1: NLU prediction performance and Average response times

Model	Parameters	F1_score	Average response time On prem _NLU	Average response time FB Channel _NLU
NLU model service (BERT transformer)	pre-trained model on massive dataset available publicly and fine-tuned on the evaluation corpora: all tokens-768 last layer	0.94	767.8 ms	896.73 ms

Conclusions

In this paper, we presented a rule-based chatbot integrated with the NLU service, both deployed in the cloud. We demonstrate the microservices-based integration architecture and successful communication between the involved services overcoming the performance challenges typically faced by such architecture. To validate the chatbot and NLU performance, we conducted experiments for measuring system response times and NLU model's accuracy. We observed that the presented architecture manages both standard (integration with FB messenger) and bespoke (integration with NLU model deployed as S3 object store service) third-party services. The integrated application decently avoids significant issues that are typical of such architecture by maintaining the communication delays well under the limits and delivering a decent prediction performance (with F1-score of 0.94) by the NLU model.

References

[1] Nicola Dragoni et al. "Microservices: yesterday, today, and tomorrow". In: *Present and ulterior software engineering* (2017), pp. 195–216.

[2] Anmol Kumari, Komal Gupta, and Deepika Rawat. "TERRABOT—A Multitasking AI". In: *Machine Intelligence and Smart Systems*. Springer, 2021, pp. 443–456.

[3] Jan-Gerrit Harms et al. "Approaches for dialog management in conversational agents". In: *IEEE Internet Computing* 23.2 (2018), pp. 13–22.

[4] Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems 30* (2017), pp. 5998–6008.

[5] Sahak Kaghyan et al. "Review of Interactive Communication Systems for Business-to-Business (B2B) Services." In: *Electronic Imaging, Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications* 11 (2018), pp. 117-1-117–11.

[6] D Inupakutika et al. "Integration of NLP and Speech-to-text Applications with Chatbots". In: *Electronic Imaging* 2021.3 (2021), pp. 35–1.

[7] Surya Roca et al. "Microservice chatbot architecture for chronic patient support". In: *Journal of Biomedical Informatics* 102 (2020), p. 103305.

[8] Chun-Ting Lin, Shang-Pin Ma, and Yu-Wen Huang. "MSABot: A chatbot framework for assisting in the development and operation of microservice-based systems". In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 2020, pp. 36–40.

[9] *IBM Watson*. `https://www.ibm.com/watson`. Accessed: 2022-02-05.

[10] *Facebook Wit.ai*. `https://wit.ai/`. Accessed: 2022-02-05.

[11] *Google Dialogflow*. `https://cloud.google.com/dialogflow`. Accessed: 2022-02-05.

[12] Dirk Merkel et al. "Docker: lightweight linux containers for consistent development and deployment". In: *Linux journal* 2014.239 (2014), p. 2.

[13] *Designing interservice communication for microservices*. `https://docs.microsoft.com/en-us/azure/architecture/microservices/design/interservice-communication`. Accessed: 2022-02-02.

[14] *Bots for Workplace*. `https://developers.facebook.com/docs/workplace/integrations/custom-integrations/bots/`. Accessed: 2022-02-02.

[15] P Chalela et al. "Development of a bilingual Facebook Messenger chat to promote smoking cessation among young adults". In: *ANNALS OF BEHAVIORAL MEDICINE* 54 (2020), S52–S52.

Author Biography

Ganesh Reddy Gunnam is a Ph. D. candidate in the Department of Electrical Engineering at the University of Texas at San Antonio (UTSA). He received his MS degree in electrical engineering from the University of Texas Rio Grande Valley (UTRGV) in 2017. He is pursuing his studies in the Smart Deep-logic Chatbot Design, Development and Performance Assessment Methodology. His research interests include Chatbot cloud deployment, virtualization and cloud computing. For correspondence. Email: Ganeshreddy.Gunnam@utsa.edu.

Devasena Inupakutika is working with Samsung Semiconductor Inc. She received Ph.D. in the Department of Electrical Engineering at the University of Texas at San Antonio (UTSA). Her research is in the development and performance

analysis of systems and methods for enhancing mobility. Her research interests include web and mobile application development, cloud-IoT integration and deep learning based wireless LAN indoor positioning systems. For correspondence. Email: Devasena.inupakutika@my.utsa.edu.

Rahul Mundlamuri received a bachelor's degree in electronics and communications engineering from JNT University, India, in 2014, and a master's degree from the University of Houston, Houston, TX, USA, in 2016. He is currently pursuing the Ph.D. degree in electrical engineering with The University of Texas at San Antonio, TX. From 2017 to 2018, he was a Data Engineer with VRN Technologies, Austin, TX. His current research interest includes neural network-based localization services and data processing. For correspondence. Email: Rahul.mundlamuri473@gmail.com.

Sahak Kaghyan is a Postdoctoral Research Scientist at the University of Texas at San Antonio (UTSA). He received his Ph.D. in Computer Science from Russian- Armenian University in 2014. His research interests include Full Stack Web Development, Mobile application development, Conversational AI design and development, Machine Learning and Software Engineering.

David Akopian is a Professor and Associate Dean of Research at the University of Texas at San Antonio (UTSA). Prior to joining UTSA, he was a specialist with Nokia from 1999 to 2003. From 1993 to 1999, he was a staff member at the Tampere University of Technology, Finland, where he received his Ph.D. degree in 1997. His current research interests include signal processing algorithms for communication and navigation receiver