Received November 28, 2020, accepted January 3, 2021, date of publication January 6, 2021, date of current version January 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3049732

Building an Arabic Flight Booking Dialogue System Using a Hybrid Rule-Based and Data Driven Approach

AL-HANOUF AL-AJMI AND NORA AL-TWAIRESH
Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh 11495, Saudi Arabia
Corresponding author: Nora Al-Twairesh (twairesh@ksu.edu.sa)

The authors would like to thank Deanship of scientific research in King Saud University for funding and supporting this research through the initiative of DSR Graduate Students Research Support (GSR).

ABSTRACT Approaches for developing Dialogue Systems (DSs) are typically categorized into rule-based and data-driven. Data-driven DSs require a massive quantity of training data, while rule-based DSs rely on a predefined set of rules and keywords that are to be detected in the user's utterances. The data-driven approaches show more promising results, but owing to a lack of available training data for Arabic task-oriented DSs, development of an Arabic task-oriented DS has typically been conducted using the rule-based approach. In this article, we propose a hybrid rule-based and data-driven approach in a text-based flight booking DS capable of handling customer's utterances. The proposed DS was built through utilizing the Wit.ai natural language interface. The conversation flow was configured using the Wizard of Oz technique, and the DS intents and entities were developed through the use of crowdsourcing training examples. The evaluation results show that the system developed was able to understand user utterances and to self-feed efficiently.

INDEX TERMS Arabic booking bot, dialogue system, flight booking, hybrid approach.

I. INTRODUCTION

Language is widely considered to be the principal marker of human sentience, with conversation or dialogue representing the fundamental arena of language. Dialogue can be defined as a conversation between two or more agents, human or machine [1]. Dialogue typically occurs in two principal forms: human-human or human-computer.

The ideal goal of a human-computer dialogue model is to attain a conversational status comparable to human face-to-face interaction [2]. At present, human-like behavior can be achieved using dialogue systems (DSs). DSs or conversational agents (CAs) are programs that communicate with users in natural language, using text, speech, or both. They are generally categorized as being of two primary types, DSs and chatbots [3], based on how the input is processed and the output is generated. The principal difference between these two kinds of systems lies in what they are designed to model. Chatbots, also known as non-task-oriented systems, model the computer's ability to mimic the conversational (i.e., chat) characteristics of human-human interaction [4]. This kind of DS only focuses on the system's ability to chit-chat rather than to perform a particular task such as booking airline flights. Task-oriented DSs are designed to handle particular tasks through modeling the actual process of dialogue by analyzing and understanding user input, which assists the DS in generating an appropriate response [3].

According a recent survey [5], DSs for English are more developed and technically sophisticated than those tailored for Arabic, with 97% of current DSs being unsuitable for use with Arabic and the majority designed for English alone. The scarcity of Arabic DSs can be attributed to several factors, the first being that English has a far greater number of speakers than Arabic. The second factor relates to the difficulties and challenges associated with Arabic, as it is a Semitic language with rich derivational and inflectional features [5].

DSs can be categorized into rule-based (pattern matching) and data-driven. The rule-based approach, in which a user's utterance is compared with a predefined set of rules, is the oldest approach in DS development. These rules have a set

The associate editor coordinating the review of this manuscript and approving it for publication was Wenge Rong.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

of keywords that fire corresponding rules when they occur in a user's utterance. Data-driven approaches rely on training the DS using training data. Currently, using data-driven approaches in the construction of CAs is enabled by the increasing quantity of social media-generated conversational data and the convenience of massive computing power.

In general, development of an intelligent DS is a still unsolved research problem facing numerous challenges [6]. In previous years, this field had not received significant attention as a result of limited data availability and inadequate artificial intelligence (AI) and natural language processing (NLP) capabilities. DS development became easier with the emergence of DS development frameworks, such as Wit.ai, Dialogflow, Rasa, and IBM Watson Assistant. Consequently, many DSs have been introduced serving various tasks, including customer service [7], user assistance [8], and appointment scheduling.

The principal contributions of this work are as follows:

- We propose a hybrid approach to DS development that can address the challenge of developing a DS with few training data.
- We develop a DS that accomplishes flight booking task using the hybrid approach by utilizing existing DS development frameworks.
- We evaluate the system developed in two stages to determine the DS's ability to self-feed.

The remainder of this article is organized as follows. Section II presents an overview of related DS research. Section III describes the methodology in detail. Section IV explains the proposed DS design and development processes. Section V draws conclusions and suggests some directions for future work.

II. RELATED WORK

A large and ever-growing body of literature has investigated the development of bots and DSs over the last few decades. Recent and advanced DSs are reviewed here to highlight the delayed development of Arabic DSs and the state-of-the-art technologies used to build DSs. Therefore, discussion of these previous studies will be divided between those dealing with Arabic and those dealing with other languages.

A. ARABIC DS DEVELOPMENT

A number of studies have pursued the goal of developing an Arabic-based DS [9], each relying on a different knowledge base and system architecture. Some of these studies [11]–[13] have used the Artificial Linguistic Internet Computer Entity (ALICE) chatbot [10]. As one example of these attempts, Abu Shawar and Atwell [13] presented a machine-learning technique to generate an Arabic DS by retraining ALICE on non-conversational Arabic text. The proposed system accepts user input in Arabic and responds with the appropriate verse from the Quran. In addition, Shawar [12] presented a CA for accessing an Arabic question answer (QA) corpus to retrieve answers for asked questions without requiring sophisticated NLP capabilities. The proposed system was created on a large set of pattern-template matching rules using ALICE. Both these attempts aimed to create new versions of ALICE using different datasets, but the resulting systems do not simulate the nature of human-human conversation. The system in [13] is similar to an information retrieval system, while the one in [12] is more of an interface to a QA system.

Botta is the first ALICE-based chatbot that can conduct conversations using the Egyptian Arabic dialect [11]. Abu Ali and Habash introduced Botta aiming for it to become the Rosie [14] for Arabic, but according to Nielsen [15], the small number of human evaluators could not solve 80% of the system-related problems. However, its being the first Arabic dialect chatbot increases its potential for being used as a baseline for similar chatbots in the future.

In addition, there have been a number of studies [16]–[18] seeking to develop a conversational intelligent tutoring system (CITS). These studies used a number of different techniques not reliant on ALICE software in the development of their systems. For example Abdullah [17], [18], developed as a CITS to teach students aged between 10 to 12 years old the essential topics of Islam, is considered to be one of the most popular Arabic DSs. This CA used supportive texts sourced from Islamic resources. Despite the previous attempts, the proposed system is one of the early Arabic CAs that mimics human behavior effectively by simulating tutoring sessions through conversation.

A considerable amount of effort has been exerted in the development of the ArabChat DS. The first version of this DS [19] aimed to serve the research requirements of an Arabic task-oriented CA, resulting in an Arabic CA capable of handling users' conversations and simulating conversations between machine and human as an information point advisor for the Applied Science University. The issues encountered in the initial version of ArabChat were addressed using the hybrid rule mechanism [20] and the user classification method [21]. The revisions were then integrated into the enhanced version [22].

A DS that aimed to serve hotel customers was presented by Moubaiddin *et al.* [23]. The proposed system interacts with the hotel's customers and generates responses related to hotel services such as reserving a room. The system consists of two principal modules, the parser and the dialogue manager. The parser is based on Government and Binding theory, an approach to universal syntax, and a phrase structure grammar that can be applied to all languages [24].

In [25], a social Arabic chatbot called "Nabiha" was proposed. It serves as an academic counselor for students of the information technology (IT) department at King Saud University (KSU). The aim of Nabiha is to interact socially with students and answer their inquiries on the courses offered in the IT department or any other question related to their academic progress at KSU. Nabiha was developed using pattern matching and the Artificial Intelligence Markup Language.

All of the Arabic DSs mentioned above are text-based and conducted using the rule-based or pattern matching approach. Few have used Short Text Similarity along with the rule-based approach to enhance system functionality. Moreover, it has been reported [17] that owing to Arabic language challenges in building DSs, the pattern matching approach is preferable despite its disadvantages.

B. OTHER LANGUAGE DS DEVELOPMENT

Recently, there has been a greater focus on the development of end-to-end DSs, with several attempts being made using the encoder-decoder framework [26]–[28]. A simple end-to-end approach based on a seq2seq framework was proposed by Vinyals and Le [28]. This approach requires many fewer hand-crafted rules given a large conversational dataset to predict the next sentence based on the previous sentence or sentences in a conversation.

Serban *et al.* [26] developed an end-to-end trainable, open-domain conversational DS based on large dialogue corpora using generative probabilistic models. In particular, the Hierarchical Recurrent Encoder-Decoder architecture proposed in [30] was used to deal with the dialogue domain more effectively and was bootstrapped using pre-trained word embeddings and the model to a larger QA pair corpus.

Li *et al.* [27] introduced a neural reinforcement learning generation method that optimizes long-term rewards to address the short-sighting problem of the responses generated by the neural model. They used Reinforcement Learning (RL) to deliver long-term informative and easily answerable utterances for simulated virtual agents.

These studies have been successful in developing chit-chat DSs using generative models. However, these models might not provide the most applicable option for task-oriented settings.

A number of studies have identified the difficulty in locating sufficiently large datasets for training end-to-end DSs. This is especially the case for task-oriented DSs [31]. Williams *et al.* [32] introduced a hybrid code network approach composed of hand-coded rules and learning models and requiring a smaller quantity of training data for the development of end-to-end task-oriented DSs. They proposed a means by which a system can continue to learn autonomously after deployment using RL.

Human teaching in connection with a task-oriented DS is performed only by expert users in the dialogue task field. In contrast, the teaching process in non-task-oriented DSs can be performed by ordinary users. Consequently, the imitation learning method, in which the DS can learn from human teaching, suits non-task-oriented DSs effectively, because there is more than one correct response to a user's utterance. Hancock *et al.* [33] proposed a new means by which to extract training examples from a chatbot after deployment through its interaction with regular users. The proposed chatbot has the ability to estimate users' satisfaction level based on their responses.

A number of highly effective efforts, such as the Chinese online shopping DS assistant presented by Yan *et al.* [34], have been made to address the problems associated with task-oriented DSs. The Zhao system can assist a customer visiting an E-commerce website in the completion of purchase-related tasks. Three kinds of datasets were used to develop this DS and it has been embedded into a mobile online shopping application for an e-commerce site for evaluation. Tammewar *et al.* [35] proposed a hybrid model that combines both the generative and retrieval approaches in a personal assistant app that can assist users in scheduling reminders through a conversation in Hindi and English. If the proposed system cannot perform the intended task based on collected information, the conversation is handed over to a human operator.

Moreover, many DSs have been developed within DS development frameworks [36], [37]. Qaffas [36] attempted to improve chatbot semantics using Wit.ai and word sequence kernel. In that study, the author designs a new approach that uses Wit.ai to search for entities in a user-entered query and word sequence kernel to compare the entered query's similarity with other queries in the database to return the most suitable response. Wei *et al.* [37] presented AirDialog, a large dataset for flight booking that contains more than 400,000 task-oriented conversations and an evaluation environment.

We conclude that approaches vary depending on the available data for training and the purpose of the presented system. Therefore, designed approaches for chit-chat agents using a pure seq-2seq model as in [26]–[28] achieve state-of-the-art performance in chatbot building [6], but pure seq-2seq models for developing chatbots might not suit task-oriented DSs perfectly. Moreover, many studies have attempted to use their DSs by applying learning methods [27], [38] or benefiting from the abundant conversations between their bot and users to provide their system with the ability to improve continuously and use new training examples over its lifetime with minimal cost [33].

III. METHODOLOGY

To date, various models have been introduced for DS development, varying from rule-based to data-driven models. The data-driven models are efficient in the case of massive training data availability, while the rule-based models are more efficient in the case of predefined sets of rule and keyword availability.

As the quantity of training data gathered for this project was not sufficiently large to depend solely on a data-driven approach, the rule-based approach was also used. Consequently, a merge of rule-based and data-driven approaches was used to construct the flight booking DS.

To book a flight, the user must define five variables before searching for available flights and beginning the booking process. These are departure (from) and arrival (to) locations, route type (one-way or round-trip), the departure and return dates, the number and type of passengers, and the ticket type (seat class). Moreover, the methodology design we used followed the Saudia airlines constraints, because it has been used by most of that international airline's companies.

Consequently, we must define slots for the flight booking DS, as shown in TABLE 1. These slots specify what the system needs to know to process a user's request (i.e., knowledge that the system must extract from the natural language free text the user wrote).

TABLE 1. List of the system slots.

Slot name	Knowledge type	Entity name	Slot status
Departure city	City name	Location	Mandatory
Arrival city	City name	Location	Mandatory
Route type	"One way" or "Return"	Route type	Mandatory
Departure date	Date	DateTime	Mandatory
Departure time	Time	DateTime	Optional
Return date	Date	DateTime	Mandatory if the "Routing type" value is return
Return time	Time	DateTime	Optional
Adults	Number	Number	Mandatory
Children	Number	Number	Mandatory
Infant	Number	Number	Mandatory
Ticket class	"Guest," "business," and "first"	Ticket class	Mandatory

According to TABLE 1, the flight booking DS must be able to understand and extract six types of knowledge: 1 City, 2 "One way" or "Return," 3 Date, 4 Time, 5 Number, 6 "Guest," "Business," or "First." Location, Route type, Date, Time, Number, and Ticket class entities were developed to capture these knowledge types. To enhance user understanding, the system must be able to recognize whether the user utterance contains a flight booking intention to proceed in the flight booking process. Thus, the flight booking DS must be able to detect one intention (intent), flight booking, as well as the five mentioned entities to fulfill the flight booking task slots.

Furthermore, to enhance the D's ability and facilitate the use's experience an additional greeting intent along with flight booking probability, ticket quantity, and travele's type entities will be detected. Data-driven and rule-based entities are shown in Fig. 1, where the data-driven entities are blue and the rule-based entities are orange. Moreover, the primary and supporting data-driven and rule-based entities are grouped together in a red box.

The greeting intent will provide the system with the ability to identify greeting sentences, and the flight booking probability is a supporting rule-based entity for the flight booking intent, because the flight booking intent is a data-driven entity, and it is not sufficiently mature or trained enough to depend solely on it. Number, a data-driven entity, is backed up by the ticket quantity and the traveler category (adult, child, and infant), rule-based entities. Ticket quantity can detect Arabic singular, dual, and plural forms of tickets, seats, and such and the traveler's category can detect singular and plural forms of adult, children, and infant in Arabic.

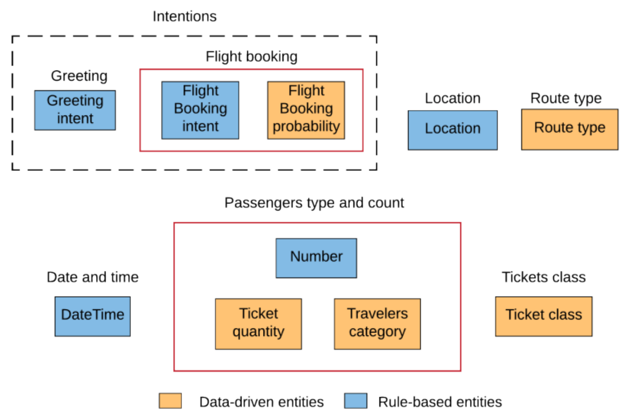

FIGURE 1. DS entities.

Therefore, all of the mentioned knowledge is to be detected through the Wit.ai framework, using the developed rule-based and data-driven entities.

The DS will be developed according to the pipelined architecture shown in Fig. 2. The system architecture consists of three principal components, a natural language Understanding (NLU) component, a Dialogue Manager (DM), and a Natural Language Generation (NLG) component. The NLU component was constructed using Wit.ai [39] and the DM was constructed using the Telegram bot [40], while the NLG component was hard coded. Moreover, the overall system functionality will be enhanced by self-feeding the system with the user' entered messages as new training examples.

A. DATA COLLECTION

Because the lack of data has impacted the data gathering process, data were collected from different sources to serve the syste's different purposes.

First, to simulate real-life airline booking scenarios, we had to specify and restrict the syste's booking ability to particular cities. This limitation can be imposed by selecting from a set of served locations. Consequently, 472 locations were collected manually from the Saudia airline website in the format "city_name airport_code" [41]. The countries and airport names along with the airports' types (international or domestic) were extracted manually from Wikipedia, and the data were pre-processed by removing duplicated airport codes and train station records.

Second, to increase the syste's ability to handle different booking requests by identifying different possible booking scenarios, we used the Wizard of Oz technique with 28 volunteers, asking them to book a ticket usin such a system. In this manner, we gathered approximately 200 preliminary knowledge booking scenarios from the volunteers. Through examining this acquired initial knowledge, we were able to identify the best means of formulating the slot filling sequence using the seven different processes outlined in Fig. 3. Thus, the design of the conversational flow will be affected by the process flow.

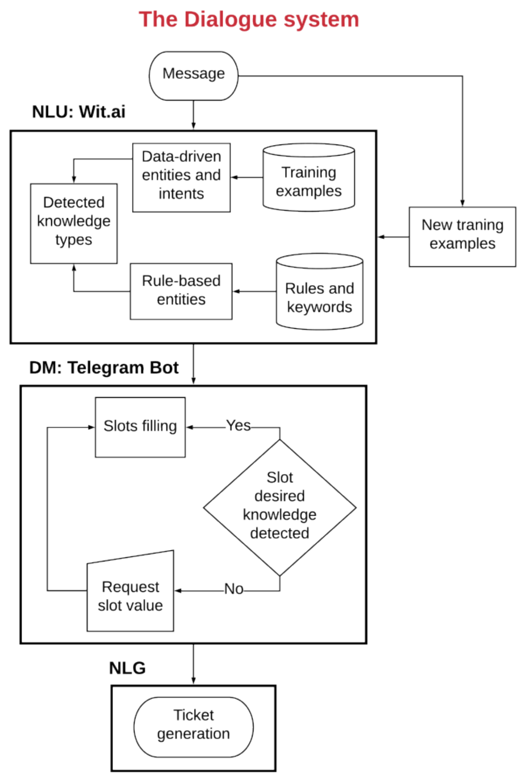

FIGURE 2. Flight booking DS architecture.

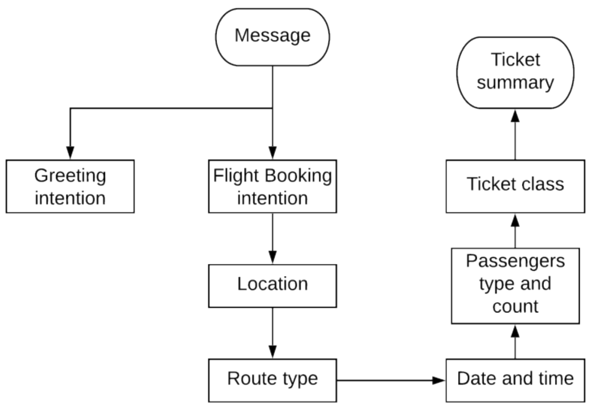

FIGURE 3. DS processes flow.

Finally, because the collected initial knowledge was not sufficient to enhance the system's ability to detect the desired values for the created rule-based entities and the trained data-driven entities, we crowdsourced 1,651 training examples. From the total of 1,651 training examples, 1,518 were positive examples and 133 were negative examples (i.e., nothing can be detected from these examples). The negative examples are typically used in data-driven entities to increase the entity's ability in detecting the desired results precisely. TABLE 2 shows the number of training examples for each entity. The training examples can be used for creating and training single or multiple entities.

TABLE 2. Number of training examples used for creating and training each entity.

Entity name	type	Training examples
Greeting intent	Data-driven	272
Flight booking intent	Data-driven	972
Flight booking probability	Rule-based	659
Location	Data-driven	1235
Date and time	Data-driven	685
Routing type	Rule-based	239
Travelers type	Rule-based	187
Ticket quantity	Rule-based	512
Numbers	Data-driven	333
Ticket class	Rule-based	128

The following sample, "I want a one-way ticket from Riyadh to Jeddah on Wednesday - يوم الاربعاء أريد تذكرة ذهاب فقط من الرياض إلى جده," was used in the creation of the routing type rule-based entity and in the training of location and date data-driven entities. The following example "أريد حجز تذكرة طيران–I want to book a flight ticket" was used only for flight booking intent data-driven entity training.

The data collected were pre-processed by removing emojis and Tatweel characters and replacing Arabic numerals (i.e., ١٢ ٣) with English numerals (i.e., 1 2 3).

B. EVALUATION

The developed system will be evaluated as a whole without considering what is happening inside its components. As a result, only the inputs and outputs of the system are accounted for, and human assessors will evaluate the system's performance.

IV. SYSTEM DEVELOPMENT AND EVALUATION

In this section, the system design and development processes are demonstrated, along with the system evaluation process and techniques.

A. SYSTEM DESIGN AND DEVELOPMENT

Two separate frameworks were used in the system design process. First, the Wit.ai framework was used as the natural language interface for the NLU component. Second, the Telegram Messenger framework was used as the DS interface for the system's DM. Python programming language was used for the system's NLG component.

The Wit.ai framework was chosen rather than other options such as Dialogflow [42] and Rasa [43]. What distinguished Wit.ai, was its ability to support Arabic in contrast to Dialogflow. Moreover, it had a set of predefined trained entities that support Arabic in contrast to Rasa. This was very handy in our case where there is a shortage of training examples. Furthermore, Wit.ai also provides Arabic speech detection, which allows the developed DS to be enhanced to support voice messages in the future.

The collected 1,651 training examples were used in Wit.ai to create data-driven and rule-based entities and to enhance and customize the Wit.ai predefined entities. First two entities in TABLE 3 were created to detect greeting and flight booking intention, with the remaining entities customized as Wit.ai predefined entities. The Wit.ai predefined entities were customized by creating roles to enhance the entities' NLU abilities.

TABLE 3. Developed and customized data-driven entities.

Entity	Roles
Greeting intent	-
Flight booking intent	-
wit/location	From
	To
wit/datetime	FromDateTime
	ToDateTime
wit/number	Adult
	child
	Infant

Similarly, the collected training examples were also used to create the remaining rule-based entities' rules, as shown in TABLE 4, and the keywords' synonyms were used to increase entity discoverability. The traveler's type and ticket quantity entities are supporting entities for wit/number and its rules, while the booking probability entity is a supporting entity for the booking intent. The supporting entities are used to minimize the DS's inability in detecting flight booking sentences resulting from the shortage of training data.

TABLE 4. Developed rule-based entities and keywords.

Entity	Keywords			Clarification
RoutingType	ذهاب فقط - One way	ذهاب وعودة - Return		-
TicketClass	الضيافة – Guest	الاعمال – Business	الاولى - First	-
TravelersType	مفرد الكبار - Singular adults	مثنى الكبار - Dual adults	جمع الكبار - Plural adults	-
	مفرد الأطفال - Singular children	مثنى الأطفال - Dual children	جمع الأطفال - Plural children	-
	مفرد الرضع - Singular infant	مثنى الرضع - Dual infant	جمع الرضع - Plural infant	-
	بين قاعدتين (مثنى الكبار وجمع الكبار) - Dual or plural adults			Such as "بالغين" and "راشدين"
TicketQuantity	مفرد - Singular	مثنى – Dual	جمع - Plural	-
	بين قاعدتين (جمع ومثنى) - Dual or plural tickets quantity			Such as "مسافرين"
BookingProbability	مطار - Airport	أحجز – Book	رحلة - Trip	
	تذكرة – Ticket	أروح - Go	السفر - Travel	

For a variety of reasons, Telegram was chosen from among the different available messaging frameworks and applications that support bot development, such as Facebook messenger, WhatsApp, and Telegram Messenger, to develop the flight booking DS. Telegram has strong messaging encryption, the target audience was more familiar with Telegram than with Facebook messenger, and Telegram provides better facilities for bot developers, such as creating custom buttons in the conversation and accepting payments from Telegram users, than WhatsApp [40].

In the development of the flight booking DS, the dialogue-was designed to be processed according to the mentioned sequence in Fig. 3. At the beginning, for more effective DS and flowchart demonstration, the following example will be used:

“اريـد حـجز رحلة ذهـابـمنالرياضللإمارات لطفلين و٣
بالغين يوم الخميسالساعه ٩مساءً”

English translation: I want to book a one-way trip from Riyadh to the Emirates for two children and 3 adults on Thursday 9 evening.

As shown in Fig. 4, the system will process the entered text by distinguishing whether it is a Telegram command. The system can serve three commands, “start,” “help,” and “restart.” The “start” command will be generated automatically at the beginning of the DS conversation. The “help” command will display a user manual describing how the user can interact with the system and what types of values the system is expecting. Finally, the “restart” command will reset the conversation slots to enable the user to start a new booking.

In a case in which the entered text was not a Telegram command, the text will be pre-processed, resulting in the following text:

“الرياض للإمارات لطفلين و3بالغين يومالخميس
الساعه 9مساءاريد حجزرحلة ذهاب من”

Next, the system will send the cleaned text to Wit.ai API and check if the greeting or flight booking intentions exist in the response message. Then, if the greeting intent is identified, the system will return a proper greeting response.

Moreover, if the flight booking intention was detected for either flight booking intent or flight booking probability entities, the system will process the text progressively to produce the requested ticket.

The greeting intent responses can be predefined or general, depending on whether the system has a specific response to the detected text. Moreover, depending on whether the entered text is short, such as “اريد ان احجزتذكرة- I want to book a ticket,” or long, such as “من الرياض إلىجدة لمسافر واحد كبيرغدا الساعة٦ مساءً على الدرجة الأولىاريد حجز تذكرة ذهاب– I want to book a one-way ticket from Riyadh to Jeddah for one large passenger tomorrow at 6 pm in first class,” the system will seek to fill the required slots if their values were detected or request them from the user otherwise.

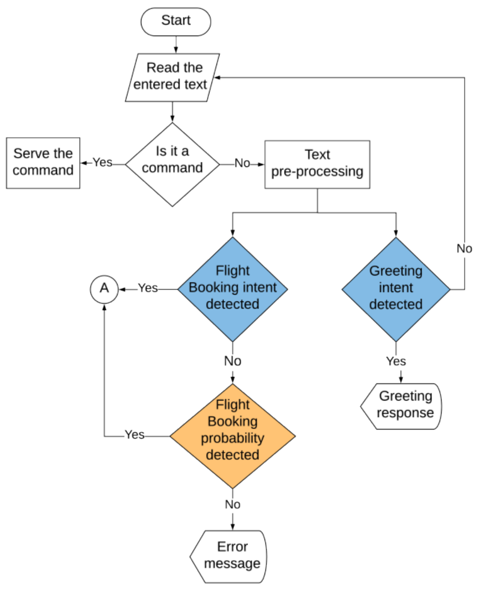

FIGURE 4. **Greeting and booking intents flowchart.**

For the illustrative example, the Wit.ai response returned will be in json format with the values in Table 5.

TABLE 5. **Wit.ai returned values.**

Entity	**Value**
FlightBooking_probability	أحجز (Book)
	رحلة (Flight)
RoutingType	ذهاب فقط (One-way)
From	الرياض (Riyadh)
To	للإمارات (Emirates)
TravelersType	مثنى الأطفال (Dual children)
	بين قاعدتين (Dual or plural adults)
Adult	3
FromDateTime	05T21:00:00.000+03:00
FlightBooking_intent	Detected from the entire sentence

Thus, the system will determine that there is a flight booking intent in the entered text, because both flight booking intent and probability entities were returned. The Wit.ai detects two flight booking probability keywords in the entered text, ''أحجز-book,'' and ''رحلة-flight.''

Once the booking intention is detected, the system will immediately switch to the location process and search for the departure (From) and arrival (To) locations in the entered text, based on the flowchart outlined in Fig. 5. The location entity and the ''From'' and ''To'' roles are used for location detection. The ''From'' role is used for detecting the departure location, while the ''To'' role is used for detecting the arrival location. The ''location'' entity is used for detecting unidentified locations. In the location process, the system will try to match the detected location with the served locations dataset. Moreover, in designing this process we cover the different possible location writing formats the user can use. The following situations are handled by the system:

- Extra letter such as ''لجدة'' at the beginning.
- Missing letter such as ''لطائف'' at the beginning.
- Missing and extra letter such as ''للطائف'' at the beginning.
- Extra word such as ''مدينة جدة'' at the beginning.

Therefore, in the illustrative example, the ''To'' location 'للإمارات'' is considered a served location but it has an additional and missing letter; hence, the extra letter ''ل'' will be removed and the missing ''ا'' will be added ''الإمارات.'' Then, the location process will be continued.

The detected served location must refer to a single record in the served locations dataset, or the user must specify the desired record. Therefore, in our example, the user must choose the desired location (city) served under the detected location. Then, the user's choice will be stored as the ''To'' slot value. This process will be conducted again to detect the ''From'' location. Moreover, according to Saudia airline, the ''From'' and ''To'' locations must not be identical. Thus, the system will also compare the user entries against that rule, before going onto the routing type process.

After detecting the ''Departure city'' and ''Arrival city'' slots' values, the system proceeds to the routing type process and searches for the ''Routing type'' slot value as well as for the remainder of the slots or requests them from the user. When the last process ends, the system can issue a ticket for the user based on the entered text.

B. SYSTEM EVALUATION

The developed flight booking DS was designed to help flight booking seekers to book an airline ticket using the developed DS rather than regular booking channels. To assess the effectiveness of the proposed approach, its ease of use, and its ability to self-feed, as well as to detect areas that require further improvement, the DS was evaluated in two stages.

There were 21 participants (13 females and 8 males) between the ages of 20 and 50. All participants were native Arabic speakers familiar with smart phones and instant messaging applications such as WhatsApp and Telegram. Moreover, all of the participants have good reading and writing skills, and 90.48% of them have previous experience in flight ticket booking.

The evaluation sessions used both pre- and post-questionnaires, with each participant being asked to answer both questionnaires. Multiple choice and scale questions were used to measure each participant's answers. Each participant was requested to book five tickets through the developed DS:

Task 1: Book a ticket of your choice.

Task 2: Book a ticket for yourself and another member of your family to perform Umrah next Monday.

Task 3: Book a ticket for your friend and his family to a Gulf Cooperation Council country, with a return date of a week after the departure date.

Task 4: Book a ticket for a man, his wife, and two children, aged four years and one year, to Turkey for four days

Task 5: Book a ticket of your choice.

We presented these tasks to the participants as scenarios and asked them to interact with the developed DS with the

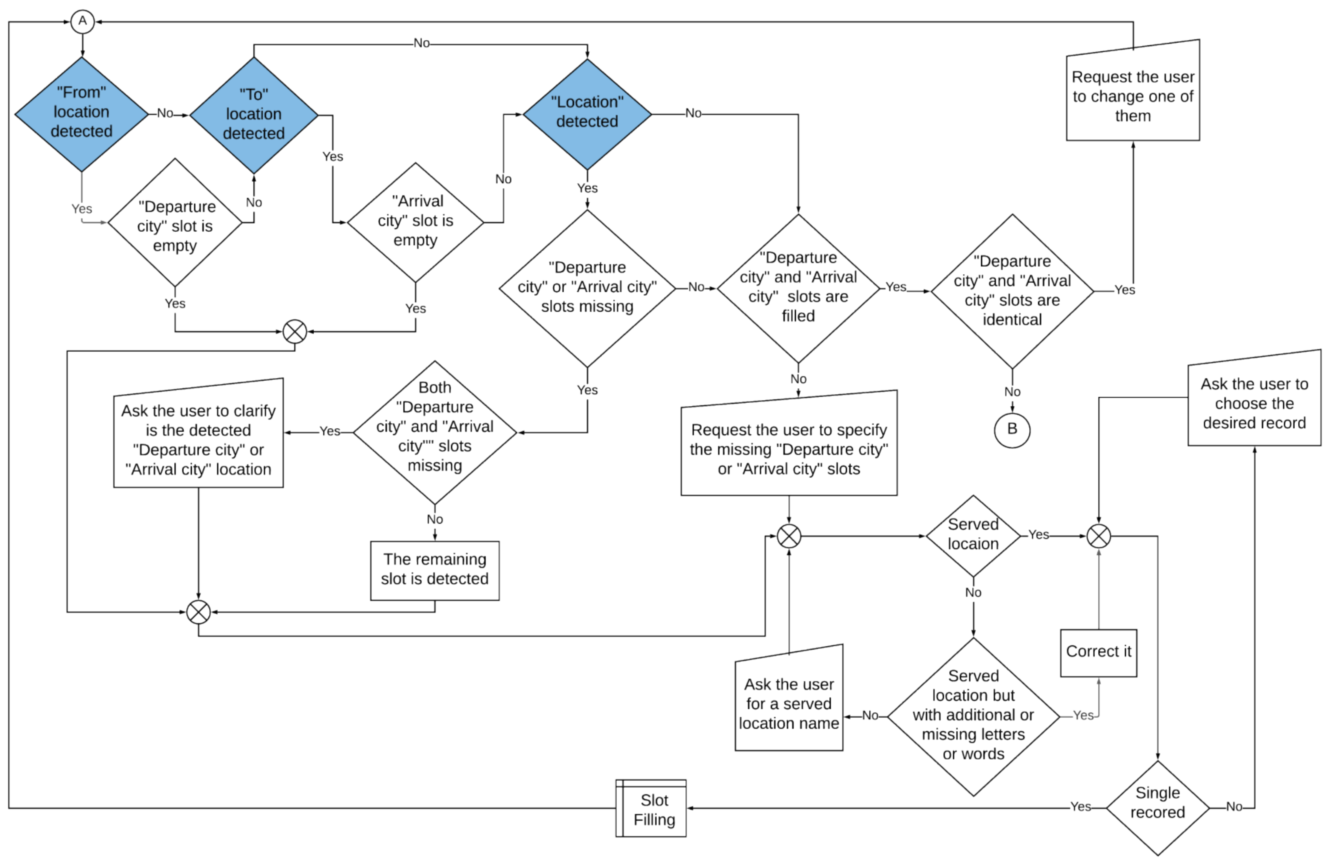
A
"From" location detected
No
"To" location detected
No
"Location" detected
No
Request the user to change one of them
Yes
"Departure city" slot is empty
No
Yes
"Arrival city" slot is empty
No
Yes
Yes
Yes
"Departure city" or "Arrival city" slots missing
No
"Departure city" and "Arrival city" slots are filled
Yes
"Departure city" and "Arrival city" slots are identical
Yes
Yes
No
No
B
Ask the user to clarify is the detected "Departure city" or "Arrival city" location
Yes
Both "Departure city" and "Arrival city"" slots missing
No
The remaining slot is detected
Request the user to specify the missing "Departure city" or "Arrival city" slots
Ask the user to choose the desired record
Served locaion
Yes
No
Correct it
Ask the user for a served location name
No
Served location but with additional or missing letters or words
Yes
Slot Filling
Yes
Single recored
No

FIGURE 5. Location process flowchart.

goal of successfully booking a ticket in each scenario. The first task was repeated at the end in an attempt to measure whether the participant becomes more familiar with the DS through use and can perform the last task with fewer errors than in the previous tasks.

A pilot test was conducted on one participant a week prior to the primary evaluation to determine the testing criteria and to examine the flight booking DS's performance. Then, participants were asked to perform the test individually. The DS was designed to handle multiple users at the same time, but it was difficult for the observer to observe their interactions simultaneously.

Both quantitative and qualitative measurements were used to determine the developed system's effectiveness, ease of use, ability to self-feed, and areas that would benefit from further improvement. The total number of errors in each task and the number of completed tasks were calculated as quantitative variables. The manners in which participants performed each task and whether they were satisfied with the resulting tickets were used as qualitative measurements, along with their comments. Furthermore, owing to the system's ability to self-feed, the system evaluation was performed in two stages. In the first stage, participants were evaluated and the system was enhanced based on their comments and the evaluator's opinion. Then, the developed models were improved to be tested at the next stage.

A brief comparison between the two evaluation stages' participants is provided in TABLE 6. All of the first-stage participants had previous experience in flight booking and approximately 90% of them preferred to book their tickets using text-based channels. In the second stage, 80% of the participants had previous experience in flight booking and 80% of them preferred to book their tickets using text-based channels. When we asked if they were booking their tickets through the airlines' official website, 54.55% of the first stage participants and 80% of the second stage participants answered, "Yes." Moreover, 27.27% of the first-stage participants have previous experience using a DS, and 33.33% of those were satisfied with it, whereas 70% of the second-stage participants have previous experience using a DS and 85.71% of those were satisfied with it.

In the pre-questionnaire, we asked the first-stage participants who had previous DS experience to guess the identity of the party they had communicated with using the DS; 66.67% of them answered that the responses they received were mixed between bot and human responses. In contrast, 85.71% of the second-stage participants who had previous experience were convinced that the party they had communicated with using the DS was a bot, because as the table shows, the second-stage participants have a better understanding of DSs. This suggests that the first-stage participants (and many other people) are confused regarding the DS's response generator's identity,

TABLE 6. **Comparison between first and second stages participants.**

Stage	First	Second
Participants	7 females and 4 males	6 females and 4 males
Have previous booking experience	100%	80%
Prefer booking tickets using text-based booking channel	90.91%	80%
Booking tickets from the official airline's website	54.55%	80%
Have previous DS experience	27.27%	70%
Satisfied about their previous DS experience	33.33%	85.71%

because they were fully confused between DSs and text-based customer service services owing either to their lower number of experiences or to their mistaking the DS's understanding of them for talking to a human agent.

In addition, we asked the participants if the identity of the interlocutor in the text-based booking channel would make a difference in the resulting ticket booking, and 63.64% of the first-stage participants answered with, "Yes," because it would understand them better if it was a human agent, while 50% of the second-stage participants answered with, "Yes," because it would be more accurate and understand them better if it was a human agent. This means that the machines and bots' ability to understand human utterances is still questioned and suspect for many people.

During the evaluation, all of the participants were able to book the requested tickets with a moderate number of committed errors (not functional errors). These required observer intervention and assistance, as shown in Fig. 6 and explained in Table 6. The errors committed by participants were labeled as follows:

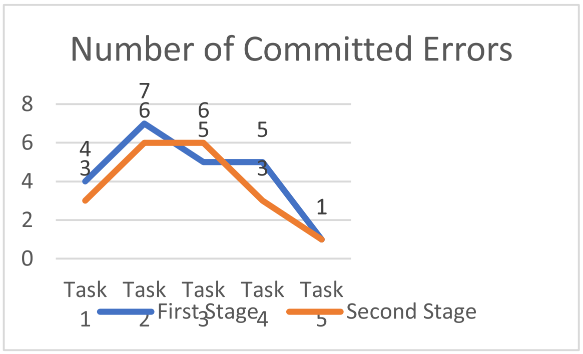

FIGURE 6. **Total number of errors committed by the participants in each task.**

CFR–The participant Could not Formulate a booking Request.

MDL–The Model was unable to Detect entered Locations correctly.

SUL–The participant Selected an Unserved Location.

MDT–The Model was unable to Detect the entered Traveler's type correctly.

UIT–The participant Used an Incorrect Ticket quantity.

MDD–The Model was unable to Detect the entered Date format correctly.

EIDF–The participant Entered an Incorrect Date Format.

The majority of the first task's errors occurred because the participants did not read the user manual, resulting in their not understanding how to formulate a booking request.

The total number of errors in the second task exceeded that of all of the other tasks, because most of them were writing Makkah as the destination location. Since there is no airport in Makkah, the bot was replying that this location has no airport or is not served by the airline, and issuing a request to select a different location. However, users continued writing the entire booking sentence again while the system was expecting only the alternative location. We attempted to overcome this issue in the second evaluation stage by replacing the bot response with a more obvious response requesting the user to enter the correct location name only.

In addition, the majority of the first and third task errors resulted from the model's inability to detect entered locations or dates, an expected behavior of the model resulting from scarcity of the training data. Most of the participant's date formulating errors occurred because they were unaware of the possible date formats that the bot could understand. To mitigate this error in the second stage, we suggested some of the accepted date formats in the error message.

In the fourth task, most of the errors resulted from the participants' lack of knowledge because children of age one year are considered infants, they continue booking a ticket for the fourth task as if it was a child. In addition, the participants continued writing "تذكره – ticket" without providing other information regarding the passenger count, and they became annoyed if the bot considered the requested ticket as one. Moreover, the last task errors resulted from model date formulating errors.

C. EVALUATION DISCUSSION

The overall experience of flight ticket booking using the developed flight booking DS, as reported by the observer, was positive. We noticed that 52% of the participants had difficulty in understanding Telegram commands, thinking that they had to restart the conversation in order to begin booking. To avoid this misunderstanding in the future, the commands' explanation message will be replaced with buttons and a greeting message will be sent after the commands' description message, perhaps providing intimation to the user to start the booking process without having to press the restart button.

The DS in the second stage was enhanced further based on participant and observer comments on the first stage, and the participants were also more experienced in using a DS than the first-stage participants had been. However,

TABLE 7. Committed errors in each task.

Task	Stage	CFR	MDL	SUL	MDT	UIT	MDD	EIDF
1	First	2	2					
	Second	2	1					
2	First	3		3	1			
	Second	1		2	1	2		
3	First	2		1				2
	Second	1	1	1		1		2
4	First	4				1		
	Second	1	1			1		
5	First						1	
	Second							1

the fact that they continued to make mistakes similarly to the previous-stage participants highlights the necessity of focusing more on finding and fixing the causes of the problem.

According to the observer, many people made obvious mistakes while using the DS, such as entering English location names or short Hijri date formats, even though the user manual stated clearly that they would not be accepted. Since only five participants read the DS user manual, we plan in the future to provide hints and tips for the users. These hints will appear when new users use the DS for the first time, one at a time, as the user reaches the relevant process section of the DS.

We found that 47.6% of the participants misunderstood the unserved location error message and re-entered the entire message again. Changing the message reduced the number of participant errors, but they still continued making the same error. This part of the location process, shown in Fig. 7, will be changed not only to accept the corrected location name but also to be able to detect the corrected location name even if the full sentence is repeated.

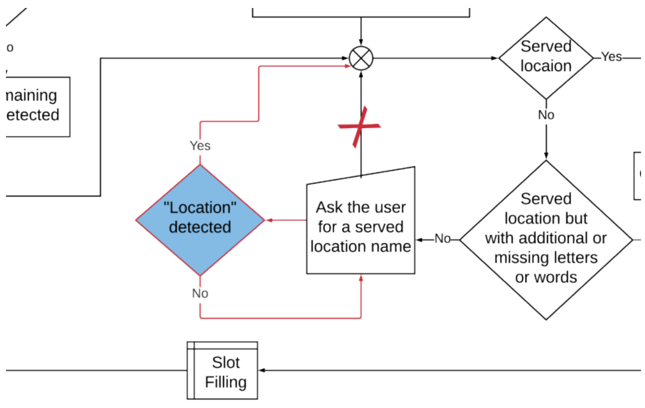

FIGURE 7. Suggested changes in the design of location process.

By dividing the evaluation process into two stages, we were able to confirm the developed system's ability to self-feed, because it became smarter over time. The system at the beginning could not detect departure and arrival locations in such a sentence as "ابي احجز تذكره الرياض جده - I need to book a ticket Riyadh Jeddah," a date format such as "4-4-2020," or the response to a passenger's type number such as "بدونلا يوجد– without, none." Hence, after enhancement, the DS can now detect such sentences.

Moreover, we noticed that participants continued saying "أحجز لي تذكرة–Book me a ticket" when they wanted to book tickets for more than one passenger, while the DS system considered that they wanted one ticket and by default it would be for an adult. Thus, in the future, the ticket quantity must not be considered until the system confirms the detected ticket quantity from the user. In addition, we can improve the DS's ability to detect the total passengers' numbers in such a sentence as "ابي احجز 5 تذاكر–I need to book 5 tickets" and ask the user for the passengers' type of those five passengers, instead of ignoring such information and focusing only on the provided adults, children, and infant passenger number.

During the evaluation, we noticed that the participants repeatedly formulated difficult sentences to test the bot's understanding and how much it could understand. We were expecting such behavior. As Qaffas reported in [36], many people attempt to beat the bot or confuse it with difficult questions.

Finally, in designing the evaluation task. we attempted to test participants' knowledge transfer ability by repeating the first task at the end. By comparing the participants behavior in the first and last tasks, we noticed that the number of committed errors was three times lower in the last task. This means that a few attempts enabled the participants to use the system easily, enhancing the user experience.

V. CONCLUSION

In this article, a new Arabic flight booking DS was developed dedicated to serving airline ticket booking. What distinguishes our DS from others, is the proposed approach, a combination of data-driven and rule-based approaches within a pipeline system architecture. This enables the users to book flight tickets in Arabic through texting.

The evaluation results showed that the developed system was an effective means of ticket booking and was easy to use as the participants gained experience with time and it became smarter after being fed with participant booking examples. The lack of training data had a major effect on DS performance, which we attempted to prevent by providing the DS with self-feeding ability.

In the developed DS, we attempted to simulate real flight booking systems, because the airline companies and flight booking agents did not give us permission to connect our system to their application programming interfaces (APIs). For the future, we are looking forward to connecting our DS with an existing booking system and enabling users to book tickets using voice messages.

REFERENCES

[1] H. Trung, "Multimodal dialogue management-state of the art," Dept. Hum. Media Interact., Univ. Twente, Enschede, The Netherlands, Tech. Rep. 0601, 2006, vol. 2.

[2] K. Nagao and A. Takeuchi, "Speech dialogue with facial displays: Multimodal human-computer conversation," in *Proc. 32nd Annu. meeting Assoc. Comput. Linguistics*, 1994, pp. 102–109.

[3] A. Dingli and D. Scerri, "Building a hybrid: Chatterbot–dialog system," in *Proc. Int. Conf. Text, Speech Dialogue*, 2013, pp. 145–152.

[4] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2009.

[5] M. Hijjawi and Y. Elsheikh, "Arabic language challenges in text based conversational agents compared to the english language," *Int. J. Comput. Sci. Inf. Technol.*, vol. 7, no. 3, pp. 1–13, Jun. 2015.

[6] M. Mnasri, "Recent advances in conversational NLP: Towards the standardization of Chatbot building," 2019, *arXiv:1903.09025*. [Online]. Available: http://arxiv.org/abs/1903.09025

[7] *TOBi—The Vodacom ChatBot*. Accessed: Apr. 7, 2020. [Online]. Available: http://www.vodacom.co.za/vodacom/services/tobi

[8] *Siri—Apple*. Accessed: Apr. 7, 2020. [Online]. Available: https://www.apple.com/siri/

[9] S. AlHumoud, A. A. Wazrah, and W. Aldamegh, "Arabic Chatbots: A survey," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 8, p. 7, 2018.

[10] R. S. Wallace, "The anatomy of A.L.I.C.E.," in *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*, R. Epstein, G. Roberts, and G. Beber, Eds. Dordrecht, The Netherlands: Springer, 2009, pp. 181–210.

[11] D. A. Ali and N. Habash, "Botta: An Arabic dialect Chatbot," in *Proc. 26th Int. Conf. Comput. Linguistics, Syst. Demonstrations (COLING)*, 2016, pp. 208–212.

[12] B. Abu Shawar, "A Chatbot as a natural Web interface to Arabic Web QA," *Int. J. Emerg. Technol. Learn. (iJET)*, vol. 6, no. 1, pp. 37–43, Mar. 2011.

[13] A. Shawar and E. S. Atwell, "An Arabic Chatbot giving answers from the Qur'an," in *Proc. 11th Conf. Sur le Traitement Automatique des Langues Naturelles (TALN)*, vol. 2, 2004, pp. 197–202.

[14] Pandorabots. (May 6, 2016). *ROSIE: Customizable Base Content*. Medium. Accessed: Apr. 8, 2020. [Online]. Available: https://medium.com/pandorabots-blog/rosie-customizable-base-content-efb925fa7d56

[15] J. Nielsen, *Usability Engineering*. Amsterdam, The Netherlands: Elsevier, 1994.

[16] S. S. Aljameel, J. D. O'Shea, K. A. Crockett, A. Latham, and M. Kaleem, "Development of an arabic conversational intelligent tutoring system for education of children with ASD," in *Proc. IEEE Int. Conf. Comput. Intell. Virtual Environ. Meas. Syst. Appl. (CIVEMSA)*, Jun. 2017, pp. 24–29.

[17] O. G. Alobaidi, K. A. Crockett, J. D. O'Shea, and T. M. Jarad, "Abdullah: An intelligent arabic conversational tutoring system for modern islamic education," in *Proc. World Congr. Eng.*, vol. 7, 2013, pp. 1–7.

[18] O. Alobaidi, "Arabic conversational agent for modern Islamic education," Ph.D. dissertation, Dept. Comput., Math. Digit. Technol., Manchester Metropolitan Univ., Manchester, U.K., 2015.

[19] M. Hijjawi, Z. Bandar, K. Crockett, and D. Mclean, "ArabChat: An arabic conversational agent," in *Proc. 6th Int. Conf. Comput. Sci. Inf. Technol. (CSIT)*, Mar. 2014, pp. 227–237.

[20] M. Hijjawi, Z. Bandar, and K. Crockett, "A novel hybrid rule mechanism for the Arabic conversational agent ArabChat," *Global J. Technol.*, no. 8, pp. 185–194, 2015.

[21] M. Hijjawi, Z. Bandar, and K. Crockett, "User's utterance classification using machine learning for arabic conversational agents," in *Proc. 5th Int. Conf. Comput. Sci. Inf. Technol.*, Mar. 2013, pp. 223–232.

[22] M. Hijjawi, Z. Bandar, and K. Crockett, "The enhanced ArabChat: An Arabic conversational agent," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 2, pp. 1–16, 2016.

[23] A. Moubaiddin, O. Shalbak, B. Hammo, and N. Obeid, "Arabic dialogue system for hotel reservation based on natural language processing techniques," *Computación y Sistemas*, vol. 19, no. 1, pp. 119–134, Mar. 2015.

[24] C. A. Black, "A step-by-step introduction to the Government and Binding theory of syntax," *Notes Linguistics*, vol. 81, no. 4, p. 76, 1998.

[25] D. Al-Ghadhban and N. Al-Twairesh, "Nabiha: An arabic dialect Chatbot," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 3, pp. 1–8, 2020, doi: 10.14569/IJACSA.2020.0110357.

[26] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Proc. AAAI Conf. Artif. Intell.*, 2016, vol. 30, no. 1, pp. 3776–3783.

[27] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep reinforcement learning for dialogue generation," 2016, *arXiv:1606.01541*. [Online]. Available: http://arxiv.org/abs/1606.01541

[28] O. Vinyals and Q. Le, "A neural conversational model," 2015, *arXiv:1506.05869*. [Online]. Available: https://arxiv.org/abs/1506.05869

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*. [Online]. Available: http://arxiv.org/abs/1706.03762

[30] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. G. Simonsen, and J.-Y. Nie, "A hierarchical recurrent encoder-decoder for generative context-aware Query suggestion," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2015, pp. 553–562.

[31] A. Bordes, Y.-L. Boureau, and J. Weston, "Learning end-to-end goal-oriented dialog," 2016, *arXiv:1605.07683*. [Online]. Available: http://arxiv.org/abs/1605.07683

[32] J. D. Williams, K. Asadi, and G. Zweig, "Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning," 2017, *arXiv:1702.03274*. [Online]. Available: http://arxiv.org/abs/1702.03274

[33] B. Hancock, A. Bordes, P.-E. Mazare, and J. Weston, "Learning from dialogue after deployment: Feed yourself, chatbot!" 2019, *arXiv:1901.05415*. [Online]. Available: https://arxiv.org/abs/1901.05415

[34] Z. Yan, N. Duan, P. Chen, M. Zhou, J. Zhou, and Z. Li, "Building task-oriented dialogue systems for online shopping," in *Proc. AAAI Conf. Artif. Intell.*, 2017, vol. 31, no. 1, pp. 4618–4625.

[35] A. Tammewar, M. Pamecha, C. Jain, A. Nagvenkar, and K. Modi, "Production ready Chatbots: Generate if not retrieve," 2018, *arXiv:1711.09684*. [Online]. Available: https://arxiv.org/abs/1711.09684

[36] A. A. Qaffas, "Improvement of Chatbots semantics using wit. ai and word sequence kernel: Education Chatbot as a case study," *Int. J. Mod. Educ. Comput. Sci.*, vol. 11, no. 3, p. 16, 2019.

[37] W. Wei, Q. Le, A. Dai, and J. Li, "AirDialogue: An environment for goal-oriented dialogue research," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 3844–3854.

[38] B. Liu, G. Tur, D. Hakkani-Tur, P. Shah, and L. Heck, "Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems," 2018, *arXiv:1804.06512*. [Online]. Available: http://arxiv.org/abs/1804.06512

[39] *Wit.ai*. Accessed: Aug. 10, 2020. [Online]. Available: https://wit.ai/

[40] *Bots: An Introduction for Developers*. Accessed: Feb. 20, 2020. [Online]. Available: https://core.telegram.org/bots

[41] *Saudi Airlines| Travel Reservation| Hotels| Holiday Packages*. Accessed: Feb. 18, 2020. [Online]. Available: https://www.saudia.com/

[42] *Dialogflow*. Accessed: Feb. 19, 2020. [Online]. Available: https://dialogflow.com/

[43] *Rasa: Open Source Conversational AI*. Rasa. Accessed: Feb. 19, 2020. [Online]. Available: https://rasa.com/

AL-HANOUF AL-AJMI received the M.Sc. degree in information technology from King Saud University in 2020. She has served as a software engineer and quality manager in several private companies. Her research interests include natural language processing and data science.

NORA AL-TWAIRESH received the Ph.D. degree in computer science from King Saud University (KSU). She is currently an Assistant Professor with the Department of Information Technology, College of Computer and Information Sciences, KSU, where she is also a member of the iWAN Research Group, and the STC Artificial Intelligence Research Chair. She has authored or coauthored several research articles. Her research interests include natural language processing, data science, and social media mining. She has served as a program committee member in many national and international conferences and a reviewer for several journals.