A PROPOSAL FOR DEVELOPING AND DEPLOYING STATISTICAL DIALOG MANAGEMENT IN COMMERCIAL CONVERSATIONAL PLATFORMS

PREPRINT

Pablo Cañas, David Griol, Zoraida Callejas
Software Engineering Department, University of Granada, Spain
{dgriol, zoraida}@ugr.es

This is a pre-print version of the chapter: Cañas, P., Griol, D., Callejas, Z. (2022). A Proposal for Developing and Deploying Statistical Dialog Management in Commercial Conversational Platforms. In: , et al. Hybrid Artificial Intelligent Systems. HAIS 2022. Lecture Notes in Computer Science(), vol 13469. Springer, Cham. https://doi.org/10.1007/978-3-031-15471-3_35 (https://link.springer.com/chapter/10.1007/978-3-031-15471-3_35)

ABSTRACT

Conversational interfaces have recently become ubiquitous in the personal sphere by improving an individual's quality of life and industrial environments by automating services and their corresponding cost savings. However, designing the dialog model used by these interfaces to decide the following response is a hard-to-accomplish task for complex conversational interactions. This paper proposes a statistical-based dialog manager architecture, which provides flexibility to develop and maintain this module. Our proposal has been integrated using DialogFlow, a natural language understanding platform provided by Google to design conversational user interfaces. The proposed hybrid architecture has been assessed with a real use case for a train scheduling domain, proving that the user experience is highly valued and can be integrated into commercial setups.

***K*eywords** Conversational systems · Dialog management · Statistical approaches · DialogFlow

1 Introduction

Conversational Interfaces (CUI) are systems that simulate interactive conversations with humans [1, 2, 3]. These systems are meant to display human-like characteristics and support the use of spontaneous natural language to interact with different purposes, such as performing transactions, responding to questions, or chatting.

These interfaces have become a key research subject for many companies, as they have understood the potential revenue of introducing these devices into society's mainstream. Virtual Personal Assistants (VPAs), such as Google Now, Apple's Siri, Amazon's Alexa, or Microsoft's Cortana, are the most representative examples. These VPAs are used for a wide variety of tasks, from setting the alarm to updating the calendar, passing through getting directions, finding the nearest restaurants or stores, or even planning a recipe or reporting the news. CUIs are also used to make their services more engaging and increase customer satisfaction in various applications, such as making appointments and reservations, answering questions, e-government procedures, and support services.

On the other hand, machine learning and deep learning methodologies have become part of many of the current intelligent systems [4, 5, 6]. These methodologies have traditionally been used in the field of conversational interfaces to develop automatic speech recognizers [7, 8] and, more recently, to implement natural language understanding modules, natural language generators, and dialog management processes [4, 9, 10].

The dialog model of the dialog manager defines the conversational system behavior in response to user utterances and environmental states. Thus, the system performance depends significantly on the quality of this model. Early models for dialog strategies were implemented using expert systems, predefined rules, and dialog trees [3]. This methodology consists of manually determining the system's response to each user input. This approach, which is still widely used in most commercial platforms, can be appropriate for very simple use cases. For instance, on systems answering a reduced set of isolated frequently asked questions. However, highly complex dialog systems usually require several user-system turns for a successful interaction, thus making the use of this methodology unfeasible to maintain and scale [2].

Commercial platforms to develop conversational interfaces have integrated new statistical methodologies proposed in the academic settings for natural language understanding (intents and entity recognition). However, they have not yet adopted this perspective for dialog management, which would allow straightforward extension and adjustment of these interfaces to various application domains [2, 3].

To address this problem, this paper proposes a practical framework to develop statistical-based dialog managers that can be easily integrated with toolkits like Google Dialogflow[1]. As proof of concept, we have implemented a practical conversational system for a train scheduling domain. We use the functionalities provided by DialogFlow for natural language understanding, and a statistical dialog manager developed using our proposal with a dialog corpus acquired for the task.

The remainder of the paper is as follows. Section 2 describes related work on statistical data-driven dialog management and the leading available commercial platforms for developing conversational interfaces. Sections 3 and 4 describe an implementation of a practical conversational agent following our proposal and the results of its evaluation. Finally, Section 5 presents the main conclusions and future research lines.

[1] https://cloud.google.com/dialogflow/

2 State of the art

2.1 SDS modules

Conversational interfaces and dialog systems have traditionally been developed through the conjunction of several modules that emulate the main processes in human-human speech communication:

- Automatic Speech Recognition (ASR): obtains the sequence of words uttered by a speaker. It is a very complex task that has been a matter of research (see [11][12]) to overcome the issues of linguistics, transmission channel, or interaction context.
- Spoken Language Understanding (SLU): obtains the semantics from the recognized sentence. It involves morphological, lexical, and syntactical analysis [13].
- Dialog Management (DM): decides the following action of the system, interpreting the incoming semantic representation of the user input in the context of the dialog [14].
- Natural Language Generation (NLG): usually considered part of the dialog manager, obtains sentences in natural language from the internal representation of information handled by the dialog system [15].
- Text-To-Speech Synthesis (TTS): transforms the generated sentences into synthesized speech [16].

2.2 Dialog Management

The dialog management process is the main object of study in this paper. The objective of the dialog manager is to consider the information provided by the user during a conversation and the results of accessing the data repositories to decide the subsequent system response. Different statistical techniques have been proposed to automatically learn the dialog model from dialog corpora [2, 3]. A widespread methodology to implement statistical dialog management systems is applying reinforcement learning techniques in conjunction with simulated user models to iteratively learn the dialog model by means of the interactions with the simulated user [17]. First, the user model is trained with a real dialog corpus to learn how to respond to a given dialog state. During the second phase, the simulated user interacts with the dialog manager so that it optimizes the dialog strategy based on the feedback given. This allows automatic training with simulated dialogs, and it even enables the exploration of dialog strategies that were not present in the original dialog corpus.

Early models for dialog strategy learning were implemented using Markov Decision Processes (MDPs). MDPs allow forward planning and hence are used as a statistical framework for dialog policy optimization by several authors, such as [18]. However, MDPs assume that the entire space of states is observable, and this assumption does not hold when uncertainty is introduced in the system. As a result, different authors proposed the use of Partially Observable MDPs (POMDPs), which account for uncertainty [19]. The idea is to retain a full and rich state representation but only maintain probability estimates over the most likely states. Results demonstrate their increasing robustness for the design of spoken dialog systems.

In 2013, Microsoft created the Dialog State Tracking Challenge (DSTC) [20], a series of dialog state tracking tasks annually designed for the research community to develop and evaluate new models and methodologies within the dialog management field. Other statistical approaches for dialog management include example-based dialog management, dialog modeling using Hidden Markov Models, stochastic finite-state transducers, and Bayesian networks.

Our proposal for statistical dialog management is described in [9]. The dialog model decides the next system response by employing a classification process that considers the set of pairs (system turn A_i - user turn U_i) preceding the current one to decide the next system response. A data structure (that we called Dialog Register, DR) is used to store the values of the entities provided by the user during the dialog. The classification function can be defined in different ways. In [9], we propose the use of artificial neural networks to improve the results of classifying the information in the DR and the current state of the dialog to select as output the following system response.

2.3 Tools for SDS implementation

The rise of conversational systems in recent years has boosted the interest of large technology companies, such as Google, Amazon, Microsoft, and IBM, in offering commercial platforms that facilitate the development and deployment of these systems. These platforms provide a set of pre-built components and share key characteristics related to their use and implementation. One of the most important is the use of intents and entities to define the language understanding task and, therefore, the input format for the DM.

DialogFlow is a human-computer interaction technology developed by Google to create natural language conversations. It sets the foundation for some Google products, such as Google Assistant or Google Home. Both the speech-to-text and text-to-speech modules are implemented using their technology. The NLU component can be implemented by defining the different possible inputs corresponding to each user's intent, and their pre-trained algorithms will easily identify the intent. The dialog manager can be implemented with predefined answers to each intent, but it also provides a method to add the developer's trained system. DialogFlow also supports more than 20 languages, can be integrated with wearable devices, cars, smart devices, messaging platforms, web applications, or any other mobile application, and has a large community supporting the product.

Amazon Lex is a service for building conversational interfaces into any application using voice and text. It is the system powering Alexa, Amazon's VPA. As with DialogFlow, Amazon Lex has already integrated the ASR and TTS modules, utilizing Amazon's deep learning technologies, and the NLU component uses the same intent identification idea. This allows developers to focus on designing the dialog manager response. Besides, it supports easy deployment of the resulting chatbot into mobile applications and other messaging services such as Facebook Messenger or Slack. A very nice feature is that responses for the CUI can be defined directly from the Amazon Web Services (AWS) console, allowing very simple integration of the dialog management system into the SDS.

Microsoft LUIS, an acronym for Language Understanding Intelligent Service, is a machine learning-based service to build natural language into apps, bots, and IoT devices. It is used by big multinational companies such as UPS or the actual Microsoft team for their internal and external interactions. As well as its competitors, it allows developers to define user intents in any of the top 15 spoken languages, and it has a predefined machine learning algorithm that predicts the user response very accurately. It also identifies entities and other context variables. This service can be integrated with Microsoft Bot Framework and the Azure Bot Service to develop, deploy, and evaluate intelligent chatbots based on an extensible SDK, tools, templates, and AI services related to speech, such as natural language understanding or question-answering. The developed chatbots can be connected to channels such as Facebook, Messenger, Kik, Slack, Microsoft Teams, Telegram, text/SMS, or Twilio.

As a proof of concept, we have selected Google DialogFlow to show a practical integration of our proposal for statistical dialog management into commercial platforms to develop conversational interfaces. The process to complete this integration does not differ between the described platforms, given the similar descriptions used for the natural language understanding task and the availability of cloud technologies to integrate deep learning frameworks and other services, such as Google Tensorflow and Firebase.

3 Implementation of a practical conversational agent

This section details the implementation of a real conversational system by developing a statistical dialog manager, and its integration with the rest of the functionalities and modules provided by DialogFlow, Firebase Functions, and Firebase Realtime Database. The practical application task that we have selected is to provide information about the Spanish railway system to aid in rail travel planning [10].

3.1 DialogFlow Basic Elements

As described in Section 2, a spoken dialog system requires the integration of five main components: the automatic speech recognizer, the language understanding module, the dialog manager, the natural language generator (which can also be integrated as additional functionality of the dialog manager), and the text-to-speech synthesizer. DialogFlow, as part of Google's development toolkit, already provides the ASR and TTS modules. As a result, the only module left to build is the natural language understanding module. For this purpose, DialogFlow has three basic elements that developers can use to build their systems: intents, entities, and contexts.

Intents are the main element in building conversations. Each intent defines examples of user utterances that can trigger the intent, what to extract from the utterance, and how to respond. As a result, the agent will map user inputs to a specific intent to provide a response. This would represent one dialog turn within the conversation. Intents consist of four main components:

1. Intent name. Used to identify the matched intent.
2. Training phrases. Examples of what users can say to match a particular intent. From the ones the developer provides, DialogFlow automatically expands the phrases to match similar user utterances.
3. Actions and parameters. Define the relevant information extracted from user utterances. Examples of this kind of information include dates, times, names, or places.

4. Response. The system output displayed to the user. In our case, responses will not be defined. Instead, user inputs will be sent to a webhook to respond to the user using our statistical model for dialog management.

For the train scheduling task, the conversations gathered in the training corpus were taken into account to build the intent set. Figure 1 shows different examples for the considered intents, along with their translation to English.

Intent Name	Training Phrases	Parameters
Change-Departure-Date	¿Y a las 4 de la tarde ? *(And for 4 pm ?)* ¿Y para el 4 de abril de 2019 ? *(And for the 4th of April 2019 ?)* ¿Puedes decirme para el 3 de mayo a las 5:00 ? *(Could you tell me for May the 3rd at 5:00 ?)*	departureDate departureHour
Confirmation	Claro *(Of course)* Es correcto *(That is correct)* Correcto *(Correct)* Sí *(Yes)*	-
Say-Destination	Mi destino es Barcelona *(My destination is Barcelona)* Quiero ir a Barcelona en un AVE *(I want to go to Barcelona by AVE)* A Barcelona *(To Barcelona)* Ir a Barcelona *(Go to Barcelona)* A Barcelona el día 7 de abril *(To Barcelona the 7th of April)* Viajo a Barcelona para mañana *(I am travelling to Barcelona tomorrow)* El destino es Barcelona *(The destination is Barcelona)* Voy a Barcelona el día 8 de mayo en AVE *(I am going to Barcelona the 8th of May by AVE)* Viajo a Barcelona *(I am travelling to Barcelona)*	destination departureDate trainType
Say-Departure-Date	Para mañana *(For tomorrow)* Me gustaría salir el 2 de abril *(I would like to depart April the 2nd)* Para mañana a las 3 *(For tomorrow at 3)* Salgo el 4 de marzo a las 8 de la tarde *(I depart March the 4th at 8 pm)* Me gustaría coger el tren a las 5 y cuarto de hoy *(I would like to take the train today at quarter past 5)* Me gustaría salir el 2 de abril a las 16:00 *(I would like to depart April the 2nd at 16:00)* Me gustaría coger el tren el 3 de abril *(I would like to take the train April the 3rd)* Salgo el 4 de marzo *(I depart March the 4th)*	departureDate departureHour
Say-Arrival-Hour	Sí, quiero que me digas los que lleguen a las 4:00 *(Yes, I would like to know the ones that arrive at 4:00)* Quiero que llegue a las 4:00 *(I would like the train to arrive at 4:00* Me gustaría llegar a las 3:00 *(I would like to arrive at 3:00)* La hora de llegada debe ser las 3:00 *(The arrival hour must be 3:00)*	arrivalHour

Figure 1: Intents defined for the train scheduling SDS, along with their corresponding translation to English

Parameters indicate the type of information that should map in each training sentence. For example, although we have trained the intent *Say-Destination* with the city Barcelona, this indicates that it is an object of type *destination*, and could be substituted by any other representative of the same type (Madrid, Seville, Bilbao...). As a result, any time the intent *Say-Destination* is matched, we will have a value for type *destination* that will give us relevant information about the user's query.

It is important to remark that not all parameters necessarily have to appear in each intent. That is because the training phrases were defined based on the corpus set, and parameters were not mentioned in every type of utterance. To give an example, the *departureDate* parameter might appear in the *Say-Destination* intent depending on the information provided by the user.

3.2 Entities

Entities are objects used as a mechanism to identify and extract valuable data from natural language. That information can be used and treated as an input into other logic, such as looking up information, carrying out a task, or returning a personalized response.

DialogFlow incorporates a wide variety of predefined system entities, which allow the extraction of information without any additional configuration. Some of these include dates, times, cities, colors, units of measure, or names. However, developers can define custom entities depending on the domain for which the CUI is being built. To do so, two main components are needed:

1. **Entity type**. Defines the type of information to extract from user input.
2. **Entity entry**. These are the elements that belong to the same entity type.

Table 1 shows all the parameters used for this application domain together with their corresponding entity type (system or custom). These parameters correspond to each domain-dependent slot specified to create the deep learning model. As a result, every time one of the parameters is mentioned in the intents, that slot will be denoted as mentioned in order to predict the system response.

Parameter Name	Entity Type
origin	city (system)
destination	city (system)
departureDate	date (system)
arrivalDate	date (system)
departureHour	time (system)
arrivalHour	time (system)
ticketClass	ticketClass (custom)
trainType	trainType (custom)
services	services (custom)

Table 1: Parameters defined for the train scheduling SDS

3.3 Contexts

Contexts represent the current state of a user's request and allow agents to carry information from one intent to another. They can be combined to control the conversational path that the user will take during the dialog. However, the dialog state was not kept using the DialogFlow context element for the train scheduling SDS. In this case, we wanted to keep a considerable number of context variables to make the system work, and using contexts became an arduous task to accomplish. For that reason, using Firebase Realtime Database arose as a feasible alternative. As a result, every time a user input is made, the system will look for the last stored state, and it will update it with the new information gathered, hence always having an updated dialog state. Subsequently, a new database table was created, containing a field for every slot needed for the model prediction as well as the next state and the real values of the users' query parameters.

3.4 DialogFlow Fulfillment

As described, intents have a component to provide a system response once an intent is matched. However, this system is very limited because responses do not include backend logic and can be associated with individual intents. For this reason, we created a service using Firebase Cloud Functions that handle the information extracted by DialogFlow's natural language processor to generate dynamic responses based on the deep learning model created. Figure 2 shows a summary of how all the pieces are assembled, constituting a complete dialog manager for the conversational system.

The architecture goes as follows: first, the model is loaded; later, the input tensor is created; finally, the output is predicted. For this project, the model was uploaded to a server using Firebase Hosting services. Therefore, anytime a request is made to the cloud function, a GET petition is made to the URL where the model is hosted, and the model is loaded for subsequent use. Inside the cloud function, there is a specific handler for each of the different DialogFlow intents previously defined. Such handlers receive as an argument a conversation object that stores all the variables gathered by DialogFlow during the language understanding phase (for example, parameters).

After this, the statistical model predicts the next action based on the current dialog state. Based on this, the system then updates the dialog state, and finally retrieves the appropriate answer to the user.

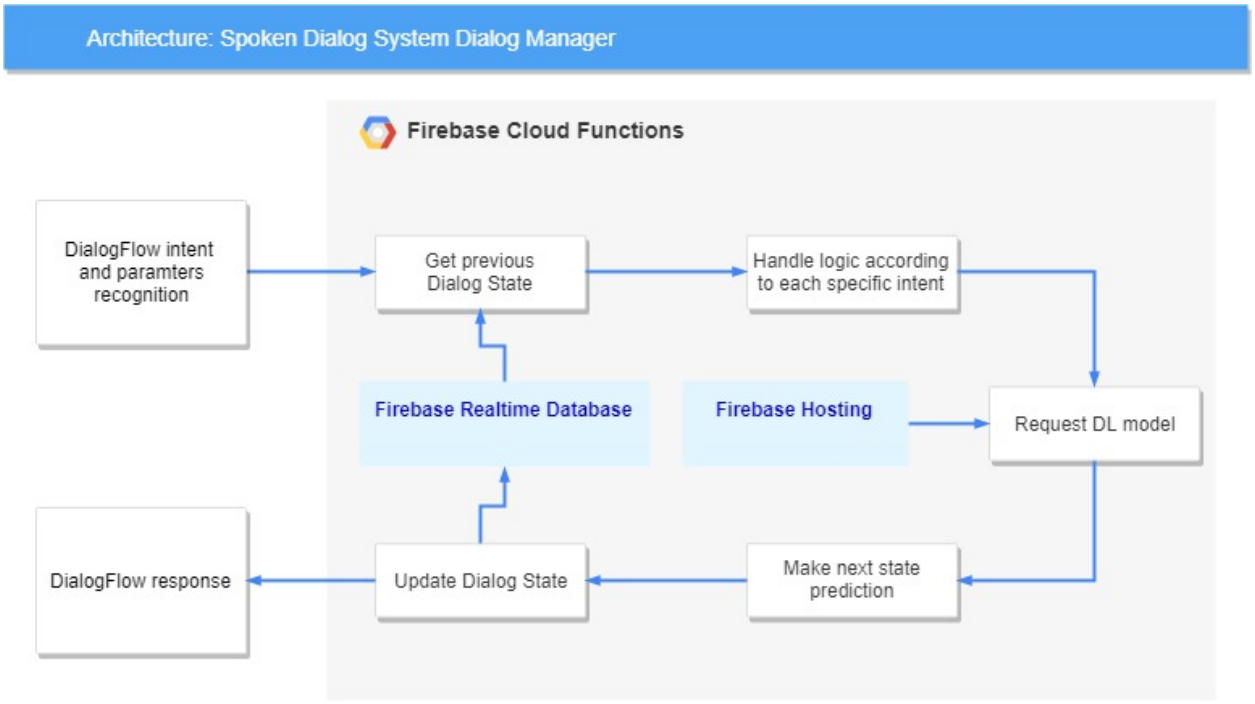

Figure 2: Dialog Manager Architecture for the proposed SDS implementation

That is the algorithm followed to retrieve an answer to user utterances: an intent is identified, the handler inside the cloud function is triggered, the model is requested to a hosting service, the database object is updated, the state is sent to the model to make a prediction, and an output message is sent back depending on the prediction made.

There are currently two versions of the Dialogflow platform: Dialogflow Essentials (ES) and Dialogflow Customer Experience (CX). Both versions use the previously described terms of entities, intents, parameters, fulfillment and webhooks. The CX version provides the concept of flows to model more complex conversations by defining conversational paths and grouping them into topics. The concept of pages is also introduced to define the state of the conversation. Each flow is usually composed of several pages and these are in charge of maintaining the dialog with the user. For each page a form can be defined, which contains a list of parameters that are obtained from the users on each page. It is also possible to migrate a project from ES to CX.

The process previously described allows to integrate the statistical dialog model in a conversational system designed with the Dialogflow Essentials version. With regard the Dialogflow Customer Experience version, it is possible to migrate the model created using Dialogflow ES or to define each of the dialog states of the statistical dialog model by means of pages, in which the list of parameters of each page corresponds to the set of entities provided by the user until the current instant of the dialog.

The proposed solution constitutes a hybrid approach for a spoken dialog system implementation. We combine a statistical dialog model with the rule-based response system developed by commercial chatbot development platforms. The statistical model can also follow a hybrid implementation, since it can be made of an ensemble of statistical methods in order to increase the robustness of the system.

4 Evaluation of the Conversational Agent

This section reports the evaluation process followed to validate the conversational agent in an authentic setting. As a result, an evaluation methodology was planned to perform the assessment of the system. After this, real users tested the CUI, out of which the evaluation was made. The assessment process was divided into an objective and a subjective evaluation.

4.1 Experiment setup

A total of 20 users were interviewed to evaluate the train scheduling chatbot. The selection of the study group was heterogeneous, comprising men and women from an age range between 21 and 60 years old, with the basic knowledge of technology to make them comfortable using a smartphone and who take a medium distance train at least once per month. Furthermore, users were told different details regarding the conversational system:

- Some users were not given any more information and were asked to try the conversational system to solve their requests. With this, we tried to look for possible outliers and classes we had not taken into account during the modeling process.
- Other users were only given specific functionalities of the application. For example, some were explained that the system could solve requests regarding scheduling, while others were asked to request information about prices and services. With this approach, we tested how the conversational system responded to very particular scenarios.
- Finally, another set of users were told about all the system specifications. This population set allowed to test the conversational system as a whole entity.

4.2 Objective evaluation

For the objective evaluation, we analyzed 9 different metrics extracted from the interactions between the user and the conversational system:

1. Dialog success rate: percentage of dialogs that were finished successfully, with users having their requests resolved.
2. Turn success rate: percentage of turns in which the system responded with a coherent answer to the user's input.
3. Dialog length: average number of turns per dialog. Considering that a turn corresponds to a two-side interaction, a user-system communication will count as one turn.
4. Request length: average number of requests made in one dialog.
5. Number of turns of the shortest dialog.
6. Number of turns of the longest dialog.
7. Percentage of different dialogs.
8. Repetitions of the most observed dialog.
9. Number of turns of the most observed dialog.
10. Number of requests of the most observed dialog.

Table 2 shows the results obtained for the objective evaluation. The results show an 80% success rate and 78% coherent answers rate. Most of the incoherences were found in dialogs ending unsuccessfully. We also found dialogs scoring a perfect 100% turn success rate.

Metric	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
Evaluation Value	80%	78%	7	2.89	10	5	55 %	6	5	2

Table 2: Objective evaluation results

4.3 Subjective evaluation

Besides the objective metrics, users were asked to assess the system's performance subjectively. A total of seven questions were made: 1. How well did the system understand you?; 2. How well did you understand the system messages?; 3. Was it easy for you to get the requested information?; 4. Was the interaction with the system quick enough?; 5. If there were system errors, was it easy for you to correct them?; 6. In general, are you satisfied with the system's performance?; 7. Would you use this system to schedule your future train rides?

Each question was scored from 1 (lowest) to 5 (highest) according to the response given by the user: 1. Never/Not at all; 2. Rarely/Poorly; 3. Sometimes/In some measure; 4. Usually/Well; 5. Always/Very well.

Table 3 shows the subjective evaluation results. It can be observed that results are generally positive, with most answers scoring between 4 and 5. The weakest points of the application have appeared to be related to the system understanding

in specific contexts. Users would perceive that the system was coherent usually, but once a mistake was made, it was not easy to recover the conversation thread. There is a total agreement for question 2, where users thought system messages were clear. Also, most of the respondents believed that the interaction was very fast, being a recurrent weak point in spoken dialog systems. In general, users are very satisfied with the performance of the system. Most of them believed that it was easy to get the information they were looking for, and many interviewees affirmed that they would use the system to schedule their future train rides.

Question	Avg. Response	Std. deviation	Med. Response	Mode Response
Q1	3.80	0.83	4	3
Q2	5	0	5	5
Q3	4	1.12	4	5
Q4	4.60	0.71	5	5
Q5	3.30	1.41	4	4
Q6	4.40	0.73	5	5
Q7	4.20	0.68	4	4

Table 3: Subjective evaluation results

5 Conclusions and future work

In this paper, we have described a proposal for the integration of statistical methodologies for dialog management. The proposal seamlessly interacts with the rest of the components of an SDS, constituting a hybrid approach in the development of these systems. On the one hand, it simplifies the time and effort required for the development of this module of the conversational system. On the other hand, it allows the development of the rest of its modules using the commercial platforms offered by large technology companies to build chatbots. As a proof of concept, we have integrated a statistical dialog management model in a practical conversational system developed with Google's DialogFlow and Firebase technologies. The results of this system's objective and subjective evaluation show the correct functioning of the system and the overall acceptance by the users. Future work includes proposing an improved statistical management model with additional information that requires error management models and developing techniques for the efficient deployment of this type of system according to the number of users interacting simultaneously with it.

6 Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 823907 (MENHIR project: https://menhir-project.eu) and the projects supported by the Spanish Ministry of Science and Innovation through the GOMINOLA (PID2020-118112RB-C21 and PID2020-118112RB-C22, funded by MCIN/AEI/10.13039/501100011033), and CAVIAR (TEC2017-84593-C2-1-R, funded by MCIN/AEI/10.13039/501100011033/FEDER).

References

[1] Steve Young. *Hey Cyba. The Inner Workings of a Virtual Personal Assistant*. Cambridge University Press, 2021.

[2] Michael McTear. *Conversational AI. Dialogue systems, Conversational Agents, and Chatbots*. Morgan and Claypool Publishers, 2020.

[3] Michael McTear, Zoraida Callejas, and David Griol. *The Conversational Interface: Talking to Smart Devices*. Springer, 2016.

[4] Lukáš Matějů, David Griol, Zoraida Callejas, José Manuel Molina, and Araceli Sanchis. An empirical assessment of deep learning approaches to task-oriented dialog management. *Neurocomputing*, 439:327–339, 2021.

[5] Nishant Kumar and Martin Raubal. Applications of deep learning in congestion detection, prediction and alleviation: A survey. *Transportation Research*, 133:103432, 2021.

[6] Weihua Li, Ruyi Huang, Jipu Li, Yixiao Liao, Zhuyun Chen, Guolin He, Ruqiang Yan, and Konstantinos Gryllias. A perspective survey on deep transfer learning for fault diagnosis in industrial scenarios: Theories, applications and challenges. *Mechanical Systems and Signal Processing*, 167:108487, 2022.

[7] Jiabin Xue, Tieran Zheng, and Jiqing Han. Exploring attention mechanisms based on summary information for end-to-end automatic speech recognition. *Neurocomputing*, 465:514–524, 2021.

[8] Emna Rejaibi, Ali Komaty, Fabrice Meriaudeau, Said Agrebi, and Alice Othmani. Mfcc-based recurrent neural network for automatic clinical depression recognition and assessment from speech. *Biomedical Signal Processing and Control*, 71:103107, 2022.

[9] P. Canas, D. Griol, and Z. Callejas. Towards versatile conversations with data-driven dialog management and its integration in commercial platforms. *J. Comput. Sci.*, 55:101443, 2021.

[10] D. Griol, L.F. Hurtado, E. Segarra, and E. Sanchis. A domain-independent statistical methodology for dialog management in spoken dialog systems. *Comput. Speech Lang*, 28(3):743–768, 2014.

[11] Alexandros Tsilfidis, Iosif Mporas, John Mourjopoulos, and Nikos Fakotakis. Automatic speech recognition performance in different room acoustic environments with and without dereverberation preprocessing. *Comput. Speech Lang*, 27:380—-395, 2013.

[12] Douglas O'Shaughnessy. Invited paper: Automatic speech recognition: History, methods and challenges. *Pattern Recognition*, 41:2965–2979, 2008.

[13] Wei-Lin Wu, Ru-Zhan Lu, Jian-Yong Duan, Hui Liu, Feng Gao, and Yu-Quan Chen. Spoken language understanding using weakly supervised learning. *Comput. Speech Lang*, 24:358–382, 2010.

[14] David R. Traum and Staffan Larsson. *The Information State Approach to Dialogue Management*, pages 325–353. Springer Netherlands, 2003.

[15] Oliver Lemon. Learning what to say and how to say it: Joint optimisation of spoken dialogue management and natural language generation. *Comput. Speech Lang*, 25:210–221, 2011.

[16] T. Dutoit. *An Introduction to Text-to-Speech Synthesis*. Kluwer Academic Publishers, 1996.

[17] E. Levin, R. Pieraccini, and W. Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Trans. Speech Audio Proc*, 8(1):11–23, 2000.

[18] E. Levin, R. Pieraccini, and W. Eckert. Using Markov decision process for learning dialogue strategies. In *Proc. ICASSP'98*, volume 1, pages 201–204, Seattle, WA, USA, 1998.

[19] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. *Comput. Speech Lang*, 24:150–174, 2010.

[20] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The Dialog State Tracking Challenge. In *Proc. of SIGDIAL'13*, pages 404–413, 2013.