Journal of Computational Science 55 (2021) 101443

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Towards versatile conversations with data-driven dialog management and its integration in commercial platforms

Pablo Cañas [a], David Griol [b,*], Zoraida Callejas [b]

[a] École Polytechnique Fédérale de Lausanne (EPFL), Route Cantonale, 1015 Lausanne, Switzerland

[b] University of Granada, CITIC-UGR, Dept. Software Engineering, Periodista Daniel Saucedo Aranda sn, 18071 Granada, Spain

ARTICLE INFO

Keywords:
Conversational systems
Dialog management
Statistical approaches
Deep learning
DialogFlow

ABSTRACT

Conversational interfaces have recently become a ubiquitous element in both the personal sphere by easing access to services, and industrial environments by the automation of services, improved customer support and its corresponding cost savings. However, designing the dialog model used by these interfaces to decide system responses is still a hard-to-accomplish task for complex conversational interactions. This paper describes a data-driven dialog management technique, which provides flexibility to develop, deploy and maintain this module. Various configurations for classification algorithms are assessed with two dialog corpora of different application domains, size, dimensionalities and set of possible system responses. The results of the evaluation show satisfactory accuracy and coherence rates in both tasks. As a proof of concept, our proposal has also been integrated with DialogFlow, a platform provided by Google to design conversational user interfaces. Our proposal has been assessed with a real use case, proving that it can be deployed in conjunction with commercial platforms, obtaining satisfactory results for the objective and subjective assessments completed.

1. Introduction

Dialog systems are computer programs that support conversational interactions with their users through speech, text, or multimodal interaction [1–3]. These systems have recently become mainstream and a key research subject with the generalized use of mobile personal assistants (e.g., Apple's Siri, Google Assistant, Microsoft's Cortana), advances in automatic speech recognition and natural language understanding with the application of deep learning techniques, the greater computing processing power to process these algorithms, and the availability of greater amounts of data to train them [4]. Current uses of conversational systems include the interaction with smart speakers (e.g., Amazon Echo and Google Nest), social robots (e.g., Pepper and Furhat), and conversational systems for e-government tasks increasingly used due to the conditions of lockdown and teleworking originated by the COVID-19 pandemic (e.g., the chatbot system developed by the WHO). Such systems are making this range of services more efficient without the need for human resources, hence generating a potential billion-dollar industry around them [3].

Task-oriented dialog systems (also known as slot-filling systems) engage an interaction with the users by means of asking them a series of questions to complete a specific task instead of engaging them in general conversational interaction (in comparison to chatbots). In these tasks, the main objective of the conversational system is to fulfill a data structure with a set of slots required to complete the transactions. Users can interact with flexibility in order to formulate their queries to the system and provide additional information to the one strictly required by the system prompts.

Although expert systems are still used to develop commercial dialog systems [5,6], statistical data-driven approaches are currently the primary trend to develop academic and industrial conversational systems [1]. These approaches can tackle deviations from the users' expected inputs, are easier to adapt to other domains, and evolve learning from the observed conversations.

Spoken conversational interfaces are traditionally made up of four different components: an automatic speech recognizer (ASR), which records the sequence of words uttered by the speaker; a natural language understanding module (NLU), which obtains the semantics from the recognized sentence by performing morphosyntactic analysis; a dialog manager (DM), which decides the next response of the system, interpreting the semantic representation of the input in the context of the dialog; and a text-to-speech synthesizer (TTS), which transforms the

* Corresponding author.
E-mail addresses: pablo.canascastellanos@alumni.epfl.ch (P. Cañas), dgriol@ugr.es (D. Griol), zoraida@ugr.es (Z. Callejas).

https://doi.org/10.1016/j.jocs.2021.101443
Received 18 April 2021; Received in revised form 31 August 2021; Accepted 6 September 2021
Available online 20 September 2021

response in natural language into synthesized speech. Statistical techniques are providing very positive results in most of these areas, such as speech recognition and spoken language understanding [7,8]. However, its potential for dialog management has started to be studied more recently [2,9,10].

Designing good dialog models is a key task to develop conversational interfaces given that this model controls the main guidelines of the conversation flow related to the design of effective and natural system prompts; detect, prevent and recover from errors; offer help and act cooperatively; effectively recognize users' intentions promote engagement and retention; consider context information; adapt the interaction and make it more personal and pleasant; promote engagement, etc.

Early models for dialog strategies were implemented using expert systems, predefined rules and dialog trees [11]. This methodology consists of manually determining the system's response to each of the user inputs. Such approach, which is still broadly used nowadays in most commercial platforms, can be appropriate for very simple use cases; for instance, systems answering a reduced set of isolated frequently asked questions. However, more complex dialog systems usually require several user-system interactions for a successful interaction, thus making the use of this methodology unfeasible for maintainability and scalability [1].

As a solution to this problem, new methodologies for statistical dialog modeling have been proposed during the last years [12]. Recent literature includes proposals based on partially observable Markov decision processes [13] and reinforcement deep learning [14], which generate user-system interaction simulations to learn the appropriate response for every input. Supervised-learning-based solutions have also been proposed, including the use of neural networks [15], stochastic finite-state transducers [16], Bayesian networks [17], and other deep learning techniques [18].

In this paper, we describe a proposal for statistical dialog management based on a classification process that decides the next system action by considering a data structure that encodes the dialog history. A comparative assessment of different classification algorithms has been completed using two dialog corpora of different application domains, size, languages and complexity. The evaluation results show the effects related to the use of different classifiers with regard the representation of the input features and the configuration of the network's hyperparameters.

There currently exist several frameworks that ease the task of building industrial conversational agents, being Google's DialogFlow[1] one of the most popular ones. Most of these toolkits allow specifying tree-based implementations for the dialog manager, in which the system will respond to the specified user utterances [4]. However, some toolkits, like DialogFlow, also allow developers to integrate on the cloud their own statistical model of the dialog manager for the agent implementation. This brings a huge potential to develop and maintain such module for commercial and industrial set-ups.

To achieve this objective, in this paper we also show how our proposal to develop statistical-based dialog managers can be easily integrated with toolkits like DialogFlow. As a proof of concept, we have implemented a practical conversational system for one of the proposed tasks, in which we use the functionalities provided by DialogFlow for natural language understanding and integrate a statistical dialog manager developed using our proposal with the corresponding corpus.

The remainder of the paper is as follows. Section 2 describes related work on statistical data-driven dialog management as well as the main available frameworks for the development of dialog systems. Section 3 presents our proposal to develop statistical dialog managers for conversational systems, which has been implemented and evaluated with two different tasks. Section 4 describes the experimental set-up and corpora used and presents a discussion of the results. We have also integrated our proposal with a commercial toolkit, which is described in Section 5 to provide a proof-of-concept of the practical deployment in one of the application domains studied. Finally, Section 6 presents the conclusions and future research lines.

2. State of the art

The main task of the dialog manager of a conversational system is to decide the next system action considering the current user's input and the state of the dialog [19,1]. This implies detecting and correcting possible errors and misrecognitions made by the ASR and NLU modules, deciding when to ask for a confirmation (using the confidence scores provided by these modules), consider additional information provided by the user not strictly required by the system, decide when to consult the data repositories of the system, etc. Considering this list of tasks and the knowledge sources that must be considered to deal with them (e.g., user utterances, data repositories, confidence scores, context information, etc.), the DM can be considered the central component of a spoken dialog system [20,21].

As described in the previous section, the use of expert systems to develop dialog systems are costly to design and maintain, cannot be easily extended or adapted to other application domains, and are not robust to unexpected inputs [22,23]. These models are also static unless manually updated (i.e., cannot be automatically adapted using conversational data) and cannot be guaranteed to be optimal [24]. In addition, manually designing all the rules for the dialog manager is very difficult (sometimes an impossible task) for practical domains, given the uncertainty of the inputs provided to the DM by the ASR and NLU modules.

2.1. Statistical data-driven dialog management

Statistical data-driven dialog models have been proposed to address these critical problems. These models are used to decide and optimize the subsequent system response according to a probabilistic process that considers the uncertainty of the outputs generated by the ASR and NLU modules (i.e., the system's belief about the current state of the dialog considering multiple hypotheses).

Different methodologies can be differentiated according to the data and algorithms used to learn the statistical dialog model. Corpus-based or example-based methods use data from previous dialogs in the same domain (provided by human-human conversations, acquired employing the Wizard of Oz (WoZ) technique, using conversational systems previously developed for similar application domains or through user simulation methods) to learn the parameters of the statistical dialog model [25,15]. This methodology has been benefited from the availability of labeled large datasets (e.g., MultiWoZ [26]), the definition of challenges in which datasets are available for research groups and companies to define statistical proposals for a set of evaluation tasks (e.g., DSTC), and the use of few-shot and zero-shot learning techniques when there is not sufficient data to train the models.

Reinforcement learning methods usually employ simulated users (given the number of dialogs that are required) to iteratively optimize the dialog strategy followed by the system (also known as dialog policy) by means of maximizing the expected rewards associated with selecting actions in Markov decision processes (MDPs) [27]. These methods have been mainly applied to develop dialog managers for slot-filling tasks [28,29].

MDPs do not allow to consider the uncertainty associated to the different processes that must be completed by the conversational system (ASR, NLU, DM, and response generation). Partially Observable Markov Decision Processes (POMDP) allow considering this uncertainty using a probability distribution associated to the set of states in which the system could be [30,31]. However, the size of the space of states is usually very large for practical domains, making exact belief state updating intractable. Different solutions have been proposed to address this problem: create summary spaces or partitioning the state space into

[1] https://dialogflow.com/ (Last access: August 2021).

partition beliefs to scale the problem [32,13], using a framework for Bayesian updates of dialog states [17], exploit similarities among the different spaces [33], etc. However, there are still several problems that must be addressed to apply this methodology for developing practical commercial systems: optimization requires a large number of dialogs to train the dialog policy, it is not easy to define the aspects to take into account to define the reward function, practical deployment is challenging, etc.

In these techniques, the dialog manager is usually divided into two main components. The Dialog Context Model (or Dialog State Tracking) determines the state of the dialog at each moment. To do this, this component stores information related to the history of the dialog, information to be gathered in the dialog, information about the domain (e.g., specific restrictions and regulations), discourse obligations, information related to the users (e.g., personal information, preferences, goals, etc.). The Dialog Control component (or Dialog Policy) decides the next system action considering the user's utterance and the state provided by the dialog context model. Dialog State Tracking has been explored extensively in the different editions of the Dialog State Tracking Challenge (DSTC), where multiple belief tracking approaches are compared on a shared task [34–37].

Other statistical approaches for dialog management include example-based dialog management [38], dialog modeling using hidden Markov models [39], stochastic finite state transducers [40,41], and Bayesian networks [42,43].

Conversational systems have adopted deep learning techniques in recent years. Deep neural networks (DNNs) have replaced hidden Markov models (HMMs) in ASR from around 2010, with a dramatic increase in accuracy [7]. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are also widely used in NLU for utterance classification [44], slot filling [45], classify intents and extract entities [46], frame semantic parsing [47], etc. Natural language generation (NLG) systems have also been recently developed using end-to-end neural approaches [8], long short-term memory (LSTM) recurrent neural networks [48], gated recurrent units (GRU) and transformers [49], or sequence to sequence (Seq2Seq) systems [8]. TTS systems have been developed using DNNs, showing promising results [50]. These approaches are also currently used to develop end-to-end neural dialog systems, in which the objective is to use DNNs within a Seq2seq architecture to map the user's input utterance to the system's prompt also in natural language without intermediate processes and representations [51,52].

With regard to dialog management, deep learning techniques have been mainly applied in recent years for dialog state tracking in combination with reinforcement learning [53–56]. The input information considered by the neural networks usually includes the last system prompt, the last user utterance, and the values of the slots collected in the previous turns [57,58]. General dialog models can be also specialized to more specific domains as proposed in [59,60]. The use of RNNs has been proposed to incorporate feedback from the user in the reward function and reduce the time required to train the dialog policy [61,55,62]. Multi-attention dialog state tracking networks have been also recently proposed to encode the dialog history to capture slots semantic relationships [63].

Hybrid methods have also been proposed to leverage the best of rule-based, statistical data-driven and neural dialog approaches, for instance, to combine reinforcement learning methods with knowledge from experts [64], use probabilistic rules for the statistical dialog model [65,66], or avoid generating repetitive or nonsense responses [67,68].

There is also a current research interest to develop end-to-end architectures for conversational systems, which try to directly map the input user's utterance to an output system response without requiring the traditional modular architecture to develop these systems (see Section 3). This approach is usually based on RNNs, the Sequence-to-Sequence (Seq2Seq) architecture, the use of word embeddings, or LSTM models [69,70]. These models were initially applied to

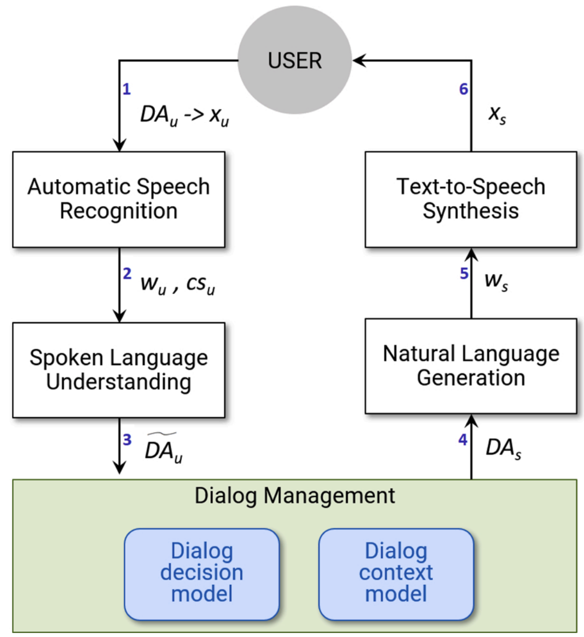


Fig. 1. Pipeline architecture for a spoken dialog system.

open-domain dialog systems. However, the application of the end-to-end approach to task-oriented systems is still very preliminary and there are many issues to be addressed (e.g., context modeling, semantic consistency, response diversity, etc.) [1].

In this paper, we explore the utility of using statistical classification algorithms for the development of a complete statistical dialog manager. Firstly, we propose an encoding to represent the state of the dialog, and we apply it to two dialog corpora of different application domains, size, dimensionalities, and set of possible system responses. Secondly, we run experiments using both traditional machine learning and deep learning algorithms to understand the potential benefits of the latter for this type of task. We also analyze the potential of recurrent-like architectures by using the system's output for the previous interaction to predict the answer of the following turn. All the architectures in our experiments were trained on a single GPU using PyTorch[2] and Scikit-learn[3] frameworks.

2.2. Platforms and frameworks to develop dialog systems

Several tools have been developed during recent years to develop and deploy dialog systems [71]. Advanced toolkits and frameworks (such as Google DialogFlow, Google Assistant, Amazon Alexa Skills Kit, Microsoft Bot Framework, or RASA) use machine learning algorithms to complete the NLU task classifying the user's utterances as intents and extracting entities. Most of these tools also include predefined categories of intents and entities (e.g., dates, cities, numbers, etc.) to facilitate developers to specify just the additional ones specific to the application domain.

DM and NLG tasks are usually completed in these tools by defining handcrafted rules, defining context conditions as input and output for each of the intents (e.g., DialogFlow). It is also possible to specify slots associated to queries (e.g., IBM Watson Assistant and the Amazon Alexa Skills Kit).

DialogFlow and IBM Watson Assistant use a parameter table to specify and extract the slots required by the system's prompts. Amazon Alexa and DialogFlow allow creating dialog models specifying required

[2] https://pytorch.org/ (Last access: August 2021).
[3] https://scikit-learn.org/ (Last access: August 2021).

and optional slots with assigned systems prompts. Slot-filling dialogs can be controlled by means of properties that determine if the objectives of the dialog have been already completed or additional steps are required. Microsoft Bot Builder allows defining slot-filling dialog management models based on forms or sequences of steps followed to collect information from users.

Only RASA and Alexa Conversations allow the use of an interactive learning mode to train a statistical dialog model in which developers must provide feedback to indicate whether the responses selected by the system were correct or wrong. This way, the dialog policy is optimized after this training. Alexa Conversations also allows using a dialog simulator that developers can configure by providing annotated dialog samples, the list of system prompts, and expected actions.

In this paper, we integrate our proposal into the DialogFlow platform. To achieve this, our model is stored in a cloud hosting service, and the commercial platform interacts with it via cloud functions to update the dialog state, handle the logic to resolve the user's request, and make a prediction for the following system response.

3. Our proposal for statistical dialog management

Fig. 1 shows a typical pipeline for a spoken dialog system. Multimodal conversational systems consider additional input/output modalities (text, gestures, emotional states, etc.). The following processes are sequentially completed in the flow to simulate the same processes that human beings follow to carry out a conversation:

1. The user provides a response to a system prompt (e.g., a query, command, provide slots values, etc.), which can be denoted as a dialog act DA_u rendered as an acoustic signal X_u.
2. The acoustic signal is the input of the ASR, that generates a list of the N-best recognition hypotheses or a word graph ($w_{u-0} \ldots w_{u-n}$) and the confidence scores associated to each one of them ($CS_{u-0} \ldots CS_{u-n}$).
3. These make up the input of the NLU module, which interprets the user's dialog act and estimates the dialog act DA_u (represented by $\widetilde{\mathrm{DA}_u}$) with a set of semantic concepts (usually represented by intents and entities provided by the user for the slots related to the application domain).
4. The DM takes the estimated dialog act as input in order to complete the following tasks. Firstly, the system's Dialog Context Model is updated to reflect the current state of the dialog by incorporating the information in the frames of the estimated dialog act. Secondly, the Dialog Decision Model decides the next system action given the information in the Dialog Context Model, which may involve producing a system dialog act as DAs.
5. The NLG component converts system DAs into a word string w_s.
6. The TTS component renders w_s as an acoustic signal X_s which may prompt the user for further input leading to another cycle through the process.

In our proposal for statistical dialog management, a dialog is represented with a sequence of states s_i that consist of pairs (system turn A_i – user turn U_i). The main objective of the DM at each time i is to select the best system action (represented by means of a dialog act DA_{s-i}) given the preceding dialog states:

$$\widehat{\mathrm{DA}}_{s-i} = \underset{\mathrm{DA}_{s-i} \in \mathcal{DA}_s}{\mathrm{argmax}}\, P(\mathrm{DA}_{s-i} | S_1, \ldots, S_{i-1}) \tag{1}$$

where set $\mathcal{DA}_s$ contains all the possible system answers.

As the number of possible states sequences is very large, we define a data structure (that we called Dialog Register, DR) that denotes the representation of dialog states for the dialog state tracking component. The DR_i stores the intents and entities values provided by the user throughout the previous history of the dialog (i.e., a summary of the information in the sequence $S_1, \ldots, S_{i-1}$ considering that the different sequences of state spaces that lead to the same DR are equivalent). Using this data structure, the selection of the next system action by the DM is carried out using the following equation:

$$\widehat{\mathrm{DA}}_{s-i} = \underset{\mathrm{DA}_{s-i} \in \mathcal{DA}_s}{\mathrm{argmax}}\, P(A_i | \mathrm{DR}_i, \mathrm{DA}_{s-i-1}) \tag{2}$$

where the updated DR at time i and the action selected by the DM in the previous turn (DA_{s-i-1}) are considered to decide the best system action for the current system turn (DA_{s-i}).

The classification function to implement the previous equation can be defined in several ways. In previous work, we have evaluated classical techniques employed in machine learning and natural language processing (e.g., multinomial naive Bayes classifiers, n-gram based classifiers, classifier based on grammatical inference techniques and classifier based on neural networks [72,73,15]. In this work, we evaluate the benefits of using neural networks to define the classification function. The input layer of the network holds the input pair (DR_i, A_{i-1}) corresponding to the dialog register and the state. The values of the output layer can be seen as an approximation of the a posteriori probability of belonging to the class associated to the dialog act DA_s.

In our proposal, the information in the DR is coded considering the presence or absence of intents and entities along with its confidence score rather than representing the actual value. The value 0 for a specific slot in the DR denotes that this slot is empty (i.e., there is no reference to the user having provided a value for such intent or entity). The value 1 denotes that the user has provided a value for the corresponding slot and the ASR and NLU modules have indicated that the value is reliable according to the confidence measures that these modules provide to the DM. Finally, the value 2 denotes that there is a value for the corresponding slot, but the confidence scores show that it is not reliable.

This codification allows completing the classification reducing the number of different inputs for the classifier, given that the only information that the DM requires in slot-filling tasks to select the next system action is the presence or absence of values in this data structure. In addition, the DR can be modified to include additional values for the codification according to additional requirements of each task, information related to the regulations of the task or related to the users, task-independent information (e.g., *affirmation*, *negation*, and *not-understood* dialog acts), use different codifications for each slot, etc.

The codification of the input (DR_i, DA_{s-i-1}) for the classifier is as follows:

- The last system answer (DA_{s-i-1}) is modeled using a variable, which has as many bits as possible actions defined as output for the DM (i.e., classes).

  $$\overrightarrow{\mathrm{DA}}_{s-i} = (x_{1_1}, x_{1_2}, x_{1_3}, \ldots, x_{1_{51}}) \in \{0, 1\}^N$$

  where N is the number of system actions defined for the DM.
- For the experiments described in the paper, each slot in the Dialog Register (DR_{i-1}) can take the values {0, 1, 2}. Therefore, every characteristic has been modeled using a variable with three bits.

  $$\overrightarrow{\mathrm{DR}}_{i-1} = (x_{i_1}, x_{i_2}, x_{i_3}) \in \{0, 1\}^3 i = 1, \ldots, M$$

  where M is the number of slots in the DR.

4. Experiments

The following section describes the design of the experiments followed in this research. It includes the description of the corpora used, the model architectures employed, and the results of the different experiments.

Table 1
DIHANA corpus' main features.

Main features	Values
Number of users	225
Number of dialogs per user	4
Number of user turns	6280
Average number of user turns per dialog	7
Average number of words per user turn	7.7
Vocabulary size	823
Duration of the recording (h)	10.86

Table 2
DIHANA corpus' informable and requestable slots.

Informable slot
Origin city
Destination city
Departure date
Arrival date
Departure hour
Arrival hour
Ticket class
Train type
Order number
Services list
Requestable slot
Timetable
Price
Train type
Order number
Services list

Table 3
DSTC2 corpus' main features.

Main features	Values
Average number of utterances per dialog	14
Vocabulary size	1229
Training dialogs	1618
Validation dialogs	500
Test dialogs	117
Number of entity types (slots)	8
Number of distinct entities	452

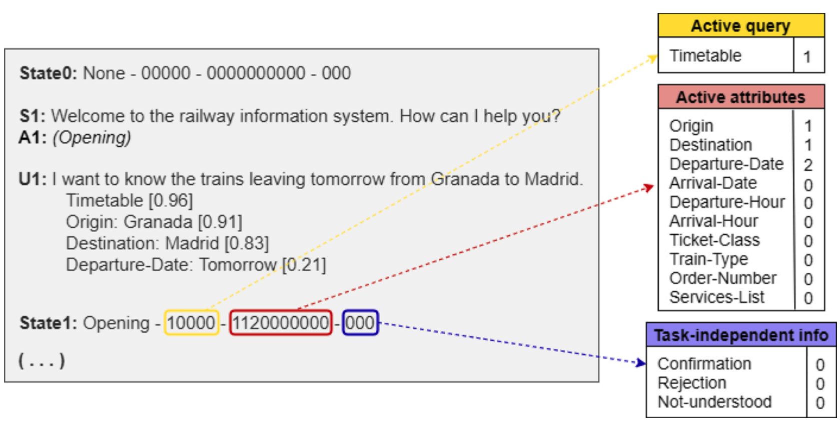


Fig. 2. Example of encoding for the train scheduling domain corpus.

4.1. *Corpora*

Two dialog datasets have been used to evaluate our proposal.

4.1.1. *DIHANA*

The DIHANA corpus [25] is a dialog dataset in the train scheduling domain, where users plan a rail trip around Spain. Users can ask the conversational interface for recommendations based on different slots, and the system process the petition and retrieve the required information. The corpus consists of a set of 900 dialogs acquired using the WoZ technique. DIHANA's main characteristics are shown in Table 1.

The DIHANA corpus contains a total of 10 *informable* slots. An *informable* slot describes any entity that the user can provide a value for, to constraint the search during the dialog. There are also five *requestable* slots, which define any entity that the user can ask for information about. DIHANA's informable and requestable slots are shown in Table 2.

To create the classification configuration, we followed the codification described in our proposal. We have identified 19 different input features:

- The previous action taken by the system.
- Five task-dependent attributes, corresponding to the *requestable* slots the user can ask for.
- Ten task-dependent attributes, which denote each of the *informable* slots that have been mentioned in the dialog.
- Three task-independent attributes, which will provide important information to build a more complete system. These are: *acceptance*, if the user has confirmed a piece of information; *rejection*, if the user has denied some information; and *not-understood*, if the system has not identified the user's input.

System DAs were labeled attending to 28 different possible system responses. These include asking the user for a slot, confirming the value of a slot, retrieving information regarding a slot, and opening and closing remarks. The preprocessed corpus consists of 4,006 samples. Fig. 2 shows an example of the encoding process followed.

4.1.2. *Dialog State Tracking Challenge 2*

The Dialog State Tracking Challenge 2 (DSTC2) dataset was presented by Henderson et al. [34] in the format of a competition for SIGDIAL 2014[4] . It consists of 2235 dialogs collected using Amazon Mechanical Turk in a restaurant information domain, in which users of the conversational system search for a restaurant based on their preferences. Its main characteristics are shown in Table 3.

Similar to the DIHANA corpus, DSTC2 contains a set of informable and requestable slots. However, unlike DIHANA, the range of slots to define a search is more limited, while the variety of request options becomes larger. There are four informable slots and eight requestable slots, shown in Table 4.

[4] https://www.sigdial.org/files/workshops/sigdial2014/ (Last access: August 2021).

Table 4
DSTC2 corpus' informable and requestable slots.

Informable slot
Area
Food
Name
Price range
Requestable slot
Area
Food
Name
Price range
Address
Phone
Post code
Signature

DSTC2 was preprocessed analogously to DIHANA. Therefore, attributes were codified with the values 0, 1, 2 according to the status of each attribute in the dialog: no information, complete information, low confidence information. Nevertheless, due to the richness of the dialogs, we analyzed 23 input attributes:

- The previous action taken by the system.
- Eight task-dependent attributes, corresponding to the requestable slots the user can ask for.
- Four task-dependent attributes, which will denote each of the informable slots that have been mentioned in the dialog.
- Ten task-independent attributes, which will provide important information regarding the user's input. They become active for the following user's actions: greeting the system, trying to end the dialog, an acknowledgment, an affirmation, a negation, thanking the system, requesting alternative suggestions, requesting more information in general, asking the system to repeat what it just said, and asking the system to start from the beginning.

The amount of classes is also larger for this task. A total of 38 different system responses were identified, ranging from confirming values, giving information about a particular slot or requesting the user to provide a value for a slot, asking the user to repeat its utterance or confirming the domain. The corpus includes a total of 11,537 training examples, three times the size of DIHANA.

4.2. Experiment design

In this subsection, we describe the machine learning models evaluated as classification functions and the evaluation methodology that we have followed.

4.2.1. Machine learning models

We have used a wide variety of traditional machine learning models together with deeper architectures, to perform an assessment on the utility of the latter for the proposed tasks. The algorithms used are the following:

1. Logistic regression.
2. K-nearest neighbors (KNN).
3. Decision tree.
4. Gradient boosting.
5. Five multilayer perceptron (MLP) architectures, with a different number of hidden layers (one to five). Each hidden layer has a linear activation function, while the output layer has a sigmoid activation function.

Further deep learning architectures based on convolutions were considered, but they were discarded due to their inappropriateness to the dataset format used for this task. While CNNs are suitable to create high-level representations of data where a spatial inductive bias applies, the 1-D vector encoding representation we crafted provides low potential to this type of architectures.

4.2.2. Evaluation methodology

We have run experiments for the nine described machine learning models on the DIHANA and DSTC2 corpora. Specifically, two batches of experiments for each dataset, including and excluding the previous response attribute. This variable introduces a recurrent structure to our network, where the output of a specific entry influences the prediction for the following. With this, we seek to analyze the influence of including the context of previous interactions in the quality of the system answer.

We split both datasets for training and testing. While the DSTC2 corpus is already divided by its creators, we performed a random 80/20 split for the DIHANA corpus. We later performed a fivefold cross-validation phase with cross-entropy loss to determine the best values for the optimizer and model hyperparameters. In particular, we set:

- Five neighbors for the KNN algorithm.
- Entropy as the function to measure the quality of the decision tree split.
- ADAM optimizer, no L2 regularization penalty, and 256 neurons per hidden layer for each MLP model.
- Learning rates of 0.0005 for the MLPs with 1 and 2 hidden layers, 0.0001 for the MLPs with 3 and 4 hidden layers, and 0.00005 for the one with 5 hidden layers.
- Number of training epochs of 200 for the MLPs with 1 and 2 hidden layers, and 300 for the rest.

The metrics used to evaluate the experiments were the following:

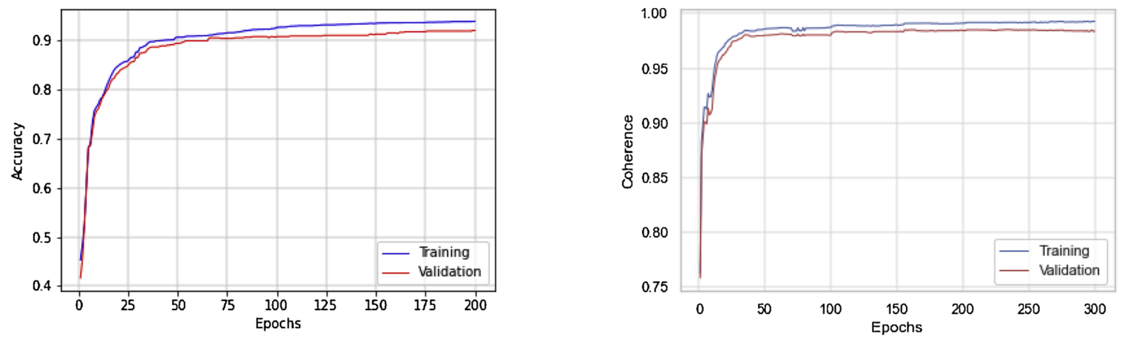
- Accuracy: percentage of turns in which the DM 1-best hypothesis is the same as in the reference answer in the corpus. This measures raw 1-best accuracy.
- Coherence: percentage of turns in which the system response is found for the same input in the whole training set. We have defined this metric because we have identified that the system will have a stochastic behavior when different responses could be given without affecting the quality of the dialog. This introduces some noise that

Table 5
DIHANA corpus' result (best results are presented in bold).

Model	Accuracy	Acc. w/o context	Coherence	Coh. w/o context	*p*-value against best
Logistic regression	88.53%	87.66%	93.27%	92.64%	1.5×10^{-9}
KNN	86.78%	88.27%	91.77%	93.27%	2.06×10^{-9}
Decision tree	91.65%	**90.65%**	**97.01%**	95.64%	1.2×10^{-5}
Gradient boosting	91.27%	90.40%	96.13%	95.26%	3.6×10^{-9}
MLP 1 hidden	**92.02%**	**90.65%**	96.88%	**95.76%**	–
MLP 2 hidden	91.65%	90.52%	96.76%	**95.76%**	1.3×10^{-4}
MLP 3 hidden	91.27%	90.27%	96.26%	95.51%	6.9×10^{-7}
MLP 4 hidden	90.65%	90.52%	95.76%	**95.76%**	2.04×10^{-7}
MLP 5 hidden	90.77%	89.40%	95.76%	94.64%	1.05×10^{-4}

Table 6
DSTC2 corpus' results (best results are presented in bold).

Model	Accuracy	Acc. w/o context	Coherence	Coh. w/o context	p-value against best
Logistic regression	71.59%	**70.95%**	97.46%	97.25%	1×10^{-16}
KNN	70.44%	64.99%	97.69%	91.34%	1×10^{-16}
Decision tree	**73.57%**	70.72%	**98.97%**	97.12%	–
Gradient boosting	73.50%	70.05%	98.23%	96.50%	1×10^{-16}
MLP 1 hidden	72.73%	70.87%	98.40%	**97.28%**	1×10^{-16}
MLP 2 hidden	71.05%	70.69%	98.61%	97.25%	1×10^{-16}
MLP 3 hidden	72.80%	70.90%	98.38%	**97.28%**	3.7×10^{-8}
MLP 4 hidden	70.75%	70.82%	98.28%	97.20%	8.4×10^{-11}
MLP 5 hidden	70.85%	70.87%	98.23%	97.20%	9×10^{-15}



(a) Accuracy evolution plot - MLP 1 hidden layer for DIHANA

(b) Coherence evolution plot - MLP 3 hidden layers for DSTC2

Fig. 3. Examples of accuracy and coherence evolution for the best performing models in each cohort.

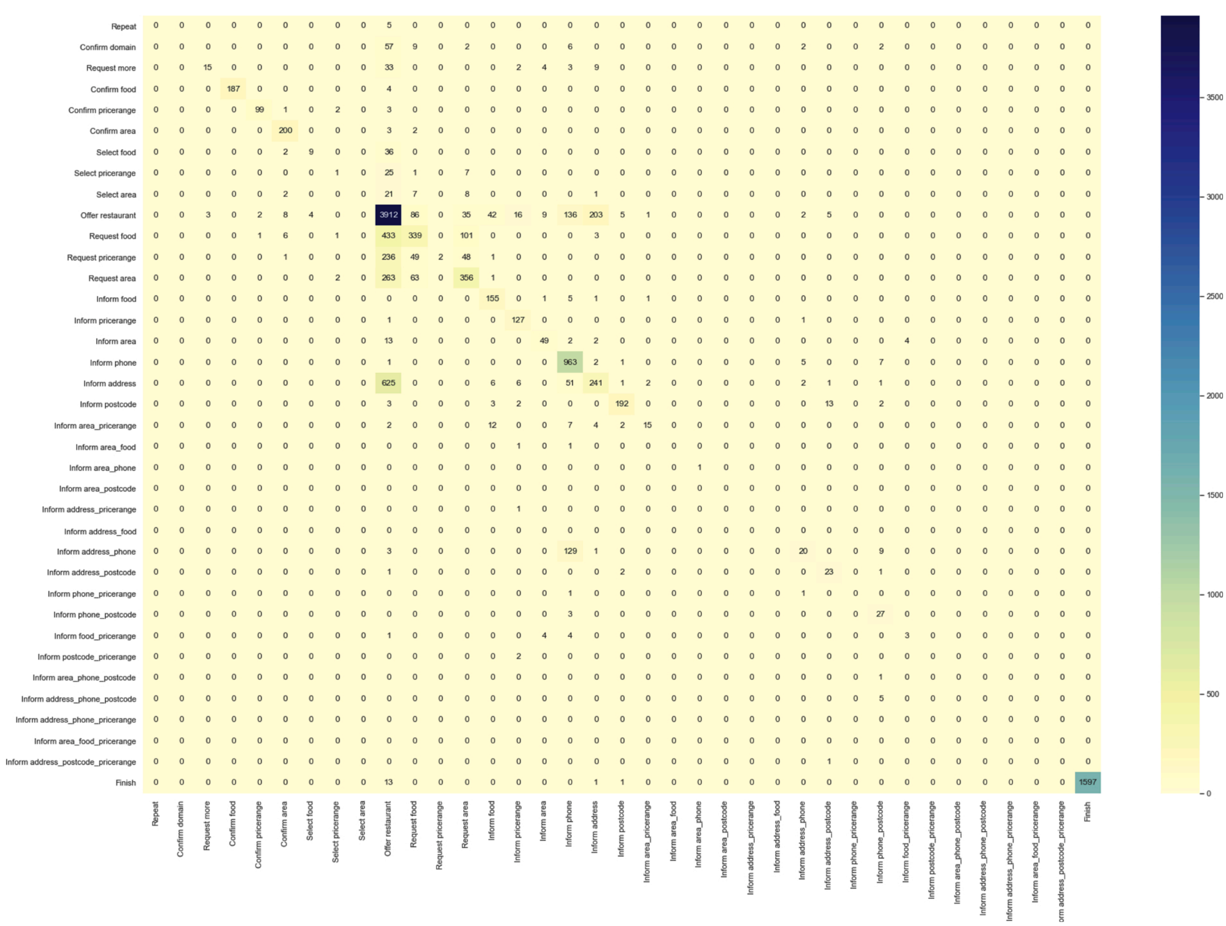


Fig. 4. Confusion matrix – MLP 3 hidden layers for DSTC2.

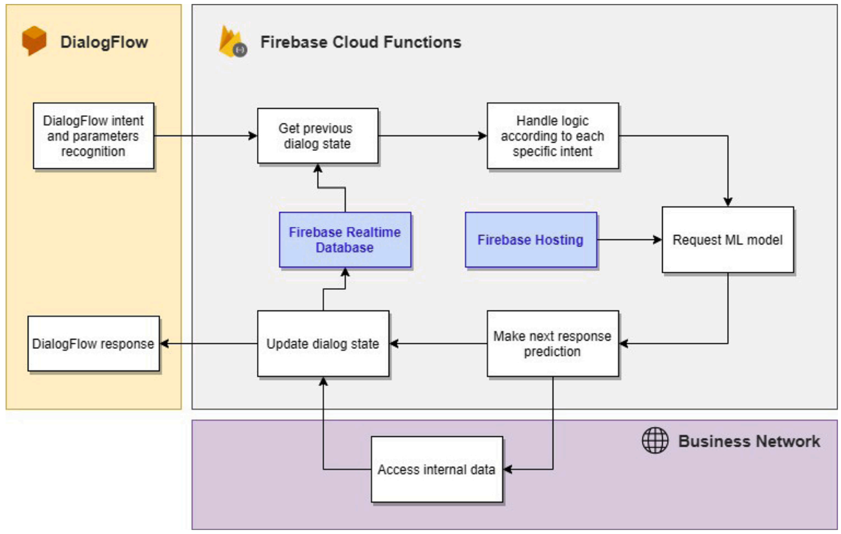


Fig. 5. Proposal for the statistical dialog manager implementation.

the models are not able to distinguish, so we defined a measure of dialog coherence to assess the quality of our proposal.

We have carried out a paired t-test to compare the different model results and verify that the evaluation metrics are statistically significant.

4.3. Results and discussion

Tables 5 and 6 describe the results obtained for the experiments in the DIHANA and DSCT2 corpora, respectively. Overall, we can observe very slight differences in performance between traditional machine learning models and deeper architectures, but nothing leading to a clear advantage towards the latter for this type of tasks.

We can also see how every model performs worse when trained without the previous response feature. This proves that a recurrent-like structure where the context of the previous interaction is saved provides a better quality predictor to guess what the next response should be.

Regarding the DIHANA corpus, we can observe that the best performing model is the MLP with one hidden layer. We report a 1-best guess rate of 92.02% and a coherence rate of 96.88%, which are statistically supported by the paired *t*-test. This model's micro-averaged precision, recall, and F1-score are 91.40%, 92.00%, and 91.70%, respectively. Fig. 3 (left) presents the accuracy evolution for this model, showing fast and converging learning with no overfitting. At the same time, the results show good calibration even for the large class unbalancing found in the dataset.

With regard to the DSTC2 corpus, we can observe that, while the coherence rate is very high, the accuracy largely decreases for this cohort. This is due to the high stochasticity of the system from which the dataset was created, since it would retrieve different answers for the same inputs, introducing a significant noise and making it hard to make an accurate 1-best guess. This is also confirmed in the confusion matrix plotted (Fig. 4), where we can observe a set of classes that are hard to distinguish.

However, this system will still be valid in a production setting, since the probability of giving an answer that would be coherent in a real scenario is very high. Fig. 3 (right) shows the coherence evolution for the MLP model with three hidden layers, which is one of the best performers for this metric.

The best accuracy performance is obtained by the decision tree model, with an accuracy of 73.57%, statistically significant as shown by the paired *t*-test results. Besides, this model's micro-averaged precision, recall, and F1-score are 73.57%, 73.72%, and 73.64%, respectively.

Finally, we can observe that none of the deeper MLP architectures evaluated has a better performance versus the one hidden layer model in any of the metrics proposed.

5. Integration with commercial solutions

Developing a dialog manager in most commercial conversational platforms involves defining a set of possible responses for each user intent. However, this set of responses is static and hence limits the flexibility of the dialog system.

For example, let us imagine a scenario based on the DIHANA corpus, where a user requests information to buy a train ticket. The user could start the interaction querying for different pieces of information: origin and destination cities, departure and arrival dates, price range, duration, services, train type, etc. A possible option is to define a single intent for these requests, as all of them are related to the same user's intention to book a train ticket. However, the number of combinations of parameters to consider becomes exponentially large for practical domains. Moreover, suppose the user does not provide all the information pieces. In that case, the agent needs to ask for the remaining data, and the range of context possibilities to take into account increases. Another option is to define unique intents for each piece of information, but again taking into account all the different combinations makes dialog management unfeasible.

To solve this, we have integrated our statistical dialog manager model with a commercial solution, and substitute the rule-based implementation. For our proposal, we have used Google's DialogFlow, since it is one of the most prominent and widely adopted platforms.

5.1. DialogFlow

DialogFlow facilitates the development of conversational interfaces by automatically implementing the natural language understanding module with training phrases provided for each intent (end-users intention for a conversation turn) and defining the dialog manager by using context conditions for each intent and the responses to return to the end-users for each of them.

DialogFlow currently supports 32 languages and dialects.[5] Conversational interfaces developed using this toolkit can be integrated into wearable devices, cars, intelligent speakers, web plugins, and other mobile applications.

[5] https://cloud.google.com/dialogflow/docs/reference/language (Last access: August 2021).

Regarding the NLU module development, DialogFlow has three basic concepts:

1. Intents: An intent is a specific action that users can invoke by using sentences that match their NLU model. Developers must provide a set of training phrases for each intent. As a result, depending on the user input, the agent maps each user response to a specific intent in order to provide a system response. Therefore, each intent represents a dialog turn within the conversation.
2. Entities: An entity represents a term or object relevant to the intents and provides a specific context for them. The entities are usually keywords used for identifying and extracting valuable data from user inputs. DialogFlow provides a wide variety of predefined system entities, such as dates, times, cities, colors, or units of measure, but developers can also define their own domain-dependent entities. An entity consists of an entity type (e.g., fruit) and entity values (e.g., banana, strawberry, orange).
3. Contexts: They represent the current state of the interaction and allow agents to carry information from one intent to another. They can be combined to control the conversational path in order to define conditions required to access an intent (input contexts) or defined after accessing them (output contexts).

Using the fulfillment functionality provided by DialogFlow, it is possible to connect natural language understanding and processing for each intent to any business logic, such as querying databases, accessing third-party APIs, or using machine-learning-based models to predict an adequate response given the dialog context. We use this functionality to integrate our proposal for more scalable statistical dialog models for the DM.

5.2. Proposed approach to statistical dialog management

Fig. 5 shows the architecture for the proposed statistical DM approach. As it can be observed, it integrates Firebase applications[6] to provide cloud functions, real-time databases, and hosting. Nonetheless, other internal or third-party services can be used to facilitate these services.

DialogFlow's NLP module is used to select the user intents and the entities in their utterances. Instead of defining a tree-based model, intents are not used to retrieve a predefined response, but to extract the context information to feed the statistical DM model appropriately with the dialog history. The context is sent to the cloud function, which will first obtain the dialog state from the previous interaction with such user.

The statistical dialog model (see Section 3) selects the next system response according to the dialog state defined by the context (for instance, confirm a particular piece of information, request additional information pieces, or inform about the results of a query). Depending on the type of response, the framework could require accessing a third party or internal database for completing the request (e.g., to inform about the ticket price for a specific train). The dialog state is updated with the data gathered and crafted during the interaction, in order to be ready for the next user input.

The framework's architecture provides modularity, scalability, speed, domain-independence, ability to handle complex and long interactions, and easiness for assembling with the rest of the modules required by complex conversational systems.

5.3. Use case: train scheduling domain

DIHANA's train scheduling domain was used to implement our proposed dialog management framework. The best performing model, a multilayer perceptron with one hidden layer, was trained and stored as a JSON object in a Firebase Hosting instance.

DialogFlow's NLP module was created by defining the set of intents, parameters and entities required for the use case. A total of 13 intents were defined, each related to a specific request or piece of information that the user could ask. Table 7 shows an example of some of the training phrases that were defined for two of those intents, the one providing information about the departure schedule, and the one asking for the duration of the trip.

The set of entities corresponds to the ten attributes shown in Table 8. Seven of these entities were already predefined in DialogFlow, while the rest were custom-defined by us.

A specific handler for each of the different DialogFlow intents was implemented for the cloud function. Following the previously described architecture, the first step is to access the Firebase Realtime Database instance to obtain the previous system response, as well as all the information that was already stored for the interaction. After this, depending on each specific intent, new information is added to the state (e.g., for the *Say-Departure-Date* shown in Table 7, departure schedule data).

The dialog state is then encoded and sent to the statistical dialog model, which uses this information as input to predict the next system response. Depending on the type of response (e.g., to provide the schedule for a train route), a new request to a third party or internal business layer can be required to inform about the trains fulfilling the conditions required by the user.

After this, the updated state is inserted in the Firebase Realtime Database, together with the system response, so that this information is

Table 7
Intent examples for the train scheduling domain.

Intent name	**Training phrases (Spanish/English transl.)**		**Parameters**
Say-Departure-Date	Para mañana. Me gustaría salir el 2 de abril. Para mañana a las 3. Salgo el 4 de marzo a las 8 de la tarde. Me gustaría coger el tren a las 5.15 de hoy. Me gustaría salir el 2 de abril a las 16:00.	For tomorrow. I would like to depart April the 2nd. For tomorrow at 3. I depart March the 4th at 8 pm. I would like to take the train today at 5.15. I would like to depart April the 2nd at 16:00	departureDate departureHour
Ask-Route-Duration	Sí, ?‘cuál es la duración del trayecto? ?‘Cuál es el tiempo de recorrido? Sí, me gustaría saber el tiempo que se tarda. ?‘Cuánto se tarda? ?‘Cuánto tarda el tren en llegar?	Yes, what is the route duration? What is the route duration? Yes, I would like to know how long does it take How long it takes? How long does the train take to arrive?	

Table 8
Parameters and entities defined for the train scheduling domain.

Parameter name	Entity type	Entity values
origin destination	city (system)	Madrid, Barcelona, Vigo, …
departureDate arrivalDate	date (system)	2020-05-04, tomorrow, …
departureHour arrivalHour	time (system)	09:30, 4 pm, noon, …
ticketClass	ticketClass (crafted)	tourist, preference
trainType	trainType (crafted)	AVE, Alvia, Cercanías, …
services	services (crafted)	cafeteria, wifi, newspaper, …

[6] https://firebase.google.com/ (Last access: August 2021).

Table 9
Results of the objective evaluation.

Metric	Evaluation value
Dialog success rate	80%
Turn coherence rate	78%
Average #turns	7
Average #requests	2.89
#Turns shortest dialog	10
#Turns longest dialog	5
% Different dialogs	55%

Table 10
Results of the subjective evaluation.

Question	Score
How well did you understand the system messages?	5.00 ± 0.00
How well did the system understand you?	3.80 ± 0.83
Was it easy for you to get the requested information?	4.00 ± 1.12
Was the interaction with the system quick enough?	4.60 ± 0.71
If there were system errors, was it easy for you to correct them?	3.30 ± 1.41
In general, are you satisfied with the performance of the system?	4.40 ± 0.73
Would you use this system to schedule your future train rides?	4.20 ± 0.68

available for the next interaction. Finally, the new system response is sent to DialogFlow.

5.4. Deployment and evaluation

One of the main reasons that make chatbot platforms such as DialogFlow ideal for industrial applications is their straightforward integration with a wide collection of popular environments. DialogFlow's integrations include Facebook Messenger, Twitter, Slack, Skype, Telegram, Google Assistant, and Amazon Alexa, among others, as well as embeddings for web applications.

A preliminary evaluation process was followed to validate our proposal and the quality of the conversational agent. As a result, a total of 20 recruited users with diverse ages (ranging from 16 to 61), gender (40% female, 60% male), educational and technological backgrounds were selected and interviewed to evaluate the train scheduling chatbot. Each of them interacted with the system in 3 different dialogs, making a total of 60 interactions, and answered a satisfaction questionnaire.

First, an objective evaluation of the system was completed, analyzing seven metrics extracted from the interactions between the users and the system. Table 9 shows the metrics and results of the evaluation. The results show that the success rate is acceptable and the turn coherence is high. The percentage of unique dialogs also points that the system is quite versatile.

Users were asked to provide their subjective opinion on the system's performance with seven questions, scoring from 1 (lowest) to 5 (highest). Results, presented in Table 10, show a positive perception of the application. While the weakest point is the error recovery capability, users believe that the interaction with the system is clear and fast. The overall satisfaction is also high, with a large percentage of returning customers. This information validates the viability of the proposed solution for industrial purposes.

Fig. 6 shows an example of a successful dialog extracted from one of the tests. Although the user speaks with colloquial wording, providing unnecessary extra information and other tags such as *'more or less'* or *'everything has become clear to me'*, the system is able to retrieve very accurate responses, and successfully complete the interaction.

6. Conclusions and future work

In this paper we have presented an approach for developing slot-

U1: Muy buenas tardes. (Good afternoon.)
S1: Bienvenido, ¿en qué puedo ayudarte? (Welcome, how can I help you?)

U2: ¿Podría decirme los horarios de trenes de Madrid a Zaragoza, por favor? (Could you please tell me the train schedule from Madrid to Zaragoza?)
S2: ¿En qué fecha desearías viajar? (In which date would you like to travel?)

U3: Para mañana, si es posible. (For tomorrow, if possible.)
S3: Buscando los trenes disponibles, espere un segundo... Hay un tren disponible de Madrid a Zaragoza el día 26/5/2019 a las 16:00. ¿Puedo ayudarte en algo más? (Searching for available trains, please wait... There is an available train from Madrid to Zaragoza on the 26/5/2019 at 16:00. Anything else?)

U4: ¿Cuánto me costaría ese billete? (How much would that ticket cost me?)
S4: El trayecto seleccionado tiene un coste de 31 euros en clase turista. ¿Necesitas algo más? (The selected route has a cost of 31 euros in tourist class. Do you need anything else?)

U5: ¿Cuánto tiempo dura el viaje, más o menos? (How long does the trip take, more or less?)
S5: El tiempo de duración del tren seleccionado es de 90 minutos. ¿Te puedo ayudar en algo más? (The selected train duration is 90 minutes. Can I help you in anything else?)

U6: ¿Y qué tipo de trenes hay disponibles en ese billete? (And what type of trains are available with that ticket?)
S6: El tipo de tren del vehículo seleccionado es AVE. ¿Tienes alguna otra consulta? (The train type of the selected vehicle is AVE. Do you have any other enquiry?)

U7: No, muchas gracias, me ha quedado todo muy claro. Muy amable. (No, thank you, everything has become clear to me. Very kind of you.)
S7: Espero haber sido de ayuda. Que tenga un buen día. (I hope I have been helpful. Have a nice day.)

Fig. 6. Example of a successful dialog extracted from the evaluation. The dialog is presented in the original language, Spanish, while a translation into English is shown enclosed in parentheses.

filling dialog managers based on machine learning techniques, and provided a lightweight representation of the conversation context that allows considering the dialog history and easily extend and adapt this information for different application tasks. We have evaluated the proposal with two corpora in different application domains, attaining satisfactory accuracy rates, being the MLP the machine learning model that shows the best performance.

We have integrated our proposal within the DialogFlow platform, easily assembling the set of components in the interface, and deploying a ready-to-use application that can be integrated into different environments and devices. We have validated our proposal developing and evaluating a dialog system based on a real use case, a train scheduling domain. This implementation has covered the end-to-end process of developing a personalized conversational interface learning a statistical dialog manager for the task and integrating it with DialogFlow using our framework. The evaluation results show the viability and potential value of our proposal to develop commercial conversational systems.

Advanced algorithms for text processing have recently emerged, it would be interesting to study their suitability for dialog management tasks. Moreover, it would be valuable to explore further the scalability of our proposal for larger and more complex datasets. Future work also includes extending the evaluation to compare different classification alternatives with real users and the automation of the processes required for creating the structure of intents and entities in toolkits such as DialogFlow.

Conflict of interest

None declared.

Acknowledgements

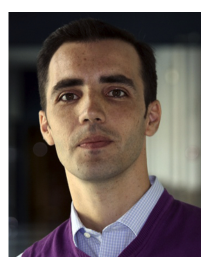
The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 823907 (MENHIR project: https://menhir-project.eu) and the Spanish project PID2020-118112RB-C21. Funding for open access charge: Universidad de Granada / CBUA.

References

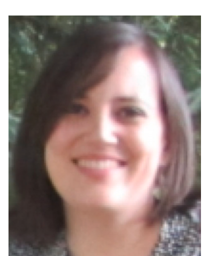
[1] M. McTear, Conversational AI. Dialogue Systems, Conversational Agents, and Chatbots, Morgan and Claypool Publishers, 2020.
[2] M. McTear, Z. Callejas, D. Griol, The Conversational Interface: Talking to Smart Devices, Springer, 2016.
[3] R. Bavaresco, D. Silveira, E. Reis, J. Barbosa, R. Righi, C. Costa, R. Antunes, M. Gomes, C. Gatti, M. Vanzin, S.C. Junior, E. Silva, C. Moreira, Conversational agents in business: a systematic literature review and future research directions, Comput. Sci. Rev. 36 (2020) 100239.
[4] S. Janarthanam, Hands-On Chatbots and Conversational UI Development: Build Chatbots and Voice User Interfaces with Chatfuel, Dialogflow, Microsoft Bot Framework, Twilio, and Alexa Skills, Packt Publishing Ltd, 2017.
[5] C. Pearl, Designing Voice User Interfaces: Principles of Conversational Experiences, O'Reilly Media, 2016.
[6] R. Batish, Voicebot and Chatbot Design: Flexible Conversational Interfaces with Amazon Alexa, Google Home, and Facebook Messenger, Packt Publishing Ltd, 2018.
[7] D. Jurafsky, J. Martin, Speech and Language Processing, Standord Edu, 2020.
[8] O. Dusek, J. Novikova, V. Rieser, Evaluating the state-of-the-art of end-to-end natural language generation: the E2E NLG challenge, Comput. Speech Lang. 59 (2020) 123–156.
[9] A. Celikyilmaz, L. Deng, D. Hakkani-Tür, Deep Learning in Natural Language Processing, Springer Singapore, Ch. Deep Learning in Spoken and Text-Based Dialog Systems, 2018, pp. 49–78.
[10] L. Mateju, D. Griol, Z. Callejas, J. Molina, A. Sanchis, An empirical assessment of deep learning approaches to task-oriented dialog management, Neurocomputing 439 (2021) 327–339.
[11] J. Lopes, M. Eskenazi, I. Trancoso, From rule-based to data-driven lexical entrainment models in spoken dialog systems, Comput. Speech Lang. 31 (1) (2015) 87–112.
[12] J. Gao, M. Galley, L. Li, Neural Approaches to Conversational AI, Now Publishers, 2019.
[13] S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, K. Yu, The Hidden Information State model: a practical framework for POMDP-based spoken dialogue management, Comput. Speech Lang. 24 (2) (2010) 150–174.
[14] H. Cuayáhuitl, S. Keizer, O. Lemon, Strategic dialogue management via deep reinforcement learning, in: Proc. NIPS'15 Workshop on Deep Reinforcement Learning, Montreal, Canada, 2015, pp. 1–10.
[15] D. Griol, Z. Callejas, R. López-Cózar, G. Riccardi, A domain-independent statistical methodology for dialog management in spoken dialog systems, Comput. Speech Lang. 28 (3) (2014) 743–768.
[16] L. Hurtado, J. Planells, E. Segarra, E. Sanchis, D. Griol, A stochastic finite-state transducer approach to spoken dialog management, in: Proc. 11th Annual Conference of the International Speech Communication Association (InterSpeech'10), Makuhari, Chiba, Japan, 2010, pp. 3002–3005.
[17] B. Thomson, S. Young, Bayesian update of dialogue state: a POMDP framework for spoken dialogue systems, Comput. Speech Lang. 24 (4) (2010) 562–588.
[18] J. Ni, T. Young, V. Pandelea, F. Xue, V. Adiga, E. Cambria, Recent Advances in deep Learning Based Dialogue Systems: A Systematic Survey, 2021, pp. 1–76. arXiv: 2105.04387.
[19] Y. Wilks, R. Catizone, S. Worgan, M. Turunen, Some background on dialogue management and conversational speech for dialogue systems, Comput. Speech Lang. 25 (2) (2011) 128–139.
[20] C. Lee, S. Jung, K. Kim, D. Lee, G. Geunbae-Lee, Recent approaches to dialog management for spoken dialog systems, J. Comput. Sci. Eng. 4 (1) (2010) 1–22.
[21] D. Traum, The Routledge Handbook of Language and Dialogue, Taylor & Francis, Ch. Computational Approaches to Dialogue, 2017, pp. 143–161.
[22] V. Rieser, O. Lemon, Reinforcement Learning for Adaptive Dialogue Systems: A Data-Driven Methodology for Dialogue Management and Natural Language Generation, Springer, 2011.
[23] D. Traum, Data-Driven Methods for Adaptive Spoken Dialogue Systems: Computational Learning for Conversational Interfaces, Springer, Ch. Statistical Approaches to Adaptive Natural Language Generation, 2017, pp. 103–130.
[24] S. Young, Probabilistic methods in spoken dialogue systems, Philos. Trans. R. Soc. Lond. Ser. A: Math. Phys. Eng. Sci. 358 (1769) (2000) 1389–1402.
[25] D. Griol, L.F. Hurtado, E. Segarra, E. Sanchis, A statistical approach to spoken dialog systems design and evaluation, Speech Commun. 50 (8) (2008) 666–682.
[26] M. Eric, R. Goel, S. Paul, A. Sethi, S. Agarwal, S. Gao, A. Kumar, A. Goyal, P. Ku, D. Hakkani-Tur, MultiWOZ 2.1: a consolidated multidomain dialogue dataset with state corrections and state tracking baselines, in: Proc. 12th Language Resources and Evaluation Conference, Marseille, France, 2010, pp. 422–428.
[27] E. Levin, R. Pieraccini, W. Eckert, A stochastic model of human-machine interaction for learning dialog strategies, IEEE Trans. Speech Audio Process. 8 (1) (2000) 11–23.
[28] S. Singh, D. Litman, M. Kearns, M. Walker, Optimizing dialogue management with reinforcement learning: experiments with the NJFun system, J. Artif. Intell. Res. 16 (2002) 105–133.
[29] A. Black, S. Burger, A. Conkie, H. Hastie, S. Keizer, O. Lemon, N. Merigaud, G. Parent, G. Schubiner, B. Thomson, J. Williams, K. Yu, S. Young, M. Eskenazi, Spoken dialog challenge 2010: Comparison of live and control test results, in: Proc. Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL'11), Portland, Oregon, 2011, pp. 2–7.
[30] J. Williams, S. Young, Partially observable Markov decision processes for spoken dialog systems, Comput. Speech Lang. 21 (2) (2007) 393–422.
[31] S. Young, M. Gasic, B. Thomson, J. Williams, POMDP-based statistical spoken dialog systems: a review, Proc. IEEE 101 (5) (2013) 1160–1179.
[32] J. Williams, S. Young, Scaling up POMDPs for dialog management: the summary POMDP method, in: Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU'05), San Juan, Puerto Rico, 2005, pp. 177–182.
[33] M. Gasic, S. Young, Gaussian processes for POMDP-based dialogue manager optimization, IEEE/ACM Trans. Audio Speech Lang. Process. 22 (1) (2013) 28–40.
[34] M. Henderson, B. Thomson, J. Williams, The second dialog state tracking challenge, in: Proc. 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), Philadelphia, PA, USA, 2014, pp. 263–272.
[35] M. Henderson, B. Thomson, J. Williams, The third dialog state tracking challenge, in: Proc. IEEE Spoken Language Technology Workshop (SLT), South Lake Tahoe, Nevada, USA, 2014, pp. 324–329.
[36] J. Williams, A belief tracking challenge task for spoken dialog systems, in: Proc. NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data (SDCTD'12), Montreal, Canada, 2012, pp. 23–24.
[37] J. Williams, A. Raux, D. Ramachandran, A. Black, The dialog state tracking challenge, in: Proc. 14th Annual SIGdial Meeting on Discourse and Dialogue, Metz, France, 2013, pp. 404–413.
[38] C. Lee, S. Jung, K. Kim, D. Lee, G. Geunbae-Lee, Example-based dialog modeling for practical multi-domain dialog system, Speech Commun. 51 (5) (2009) 466–484.
[39] H. Cuayáhuitl, S. Renals, O. Lemon, H. Shimodaira, Human-computer dialogue simulation using Hidden Markov models, in: Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU'05), San Juan, Puerto Rico, 2005, pp. 290–295.
[40] M. Calvo, L.F. Hurtado, F. García, E. Sanchis, E. Segarra, Multilingual spoken language understanding using graphs and multiple translations, Comput. Speech Lang. 38 (2016) 86–103.
[41] C. Hori, K. Ohtake, T. Misu, H. Kashioka, S. Nakamura, Recent advances in WFST-based dialog system, in: Proc. 11th Annual Conference of the International Speech Communication Association (Interspeech'10), Makuhari, Japan, 2010, pp. 268–271.
[42] T. Paek, E. Horvitz, Conversation as action under uncertainty, in: Proc. 16th Conference on Uncertainty in Artificial Intelligence (UAI'00), Stanford, CA, USA, 2000, pp. 455–464.
[43] H. Meng, C. Wai, R. Pieraccini, The use of belief networks for mixed-initiative dialog modeling, IEEE Trans. Speech Audio Process. 11 (6) (2003) 757–773.

[44] R. Collobert, J. Weston, A unified architecture for natural language processing: deep neural networks with multitask learning, in: Proc. 25th international conference on Machine learning (ICML'08), Helsinki, Finland, 2008, pp. 160–167.
[45] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, G. Zweig, Using recurrent neural networks for slot filling in spoken language understanding, IEEE/ACM Trans. Audio Speech Lang. Process. 23 (3) (2014) 530–539.
[46] D. Hakkani-Tur, G. Tur, A. Celikyilmaz, Y.-N. Chen, J. Gao, L. Deng, Y.-Y. Wang, Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM, in: Proc. 17th Annual Conference of the International Speech Communication Association (Interspeech'16), San Francisco, CA, USA, 2016, pp. 715–719.
[47] E. Vanzo, A. Bastianelli, O. Lemon, Hierarchical multi-task natural language understanding for cross-domain conversational AI: HERMIT NLU, in: Proc. 20th Annual SIGdial Meeting on Discourse and Dialogue (SIGdial'19), Stockholm, Sweden, 2019, pp. 254–263.
[48] T.-H. Wen, M. Gasic, N. Mrksic, L.M. Rojas-Barahona, P.-H. Su, D. Vandyke, S. Young, Multi-domain neural network language generation for spoken dialogue systems, in: Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'16), San Diego, California, USA, 2016, pp. 120–129.
[49] T. Castro-Ferreira, C. van der Lee, E. van Miltenburg, E. Krahmer, Neural data-to-text generation: a comparison between pipeline and end-to-end architectures, in: Proc. Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19), Hong Kong, China, 2019, pp. 552–562.
[50] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: a generative model for raw audio, in: Proc. 9th ISCA Speech Synthesis Workshop (SSW'16), Sunnyvale, USA, 2016, pp. 125–126.
[51] K. Cho, B. van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: encoder-decoder approaches, in: Proc. 8th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST'14), Doha, Qatar, 2014, pp. 103–111.
[52] A. Kannan, K. Kurach, S. Ravi, T. Kaufman, B. Miklos, G. Corrado, A. Tomkins, L. Lukacs, M. Ganea, P. Young, V. Ramavajjala, Smart reply: automated response suggestion for email, in: Proc. ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'16), San Francisco, California, USA, 2016, pp. 955–964.
[53] P. Su, N. Mrksic, I. Casanueva, I. Vulic, Deep learning for conversational AI, in: Proc. Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT'18), New Orleans, Louisiana, USA, 2018, pp. 27–32.
[54] H. Shi, T. Ushio, M. Endo, K. Yamagami, N. Horii, Convolutional neural networks for multi-topic dialog state tracking, in: Proc. 7th International Workshop on Spoken Dialogue Systems. Dialogues with Social Robots – Enablements, Analyses, and Evaluation (IWSDS'16), Saariselkä, Finland, 2016, pp. 451–463.
[55] P. Shah, D. HakkaniTur, L. Heck, Interactive reinforcement learning for task-oriented dialogue management, in: Proc. NIPS'16 Deep Learning for Action and Interaction Workshop, Barcelona, Spain, 2016, pp. 1–11.
[56] R. Loftin, B. Peng, J. MacGlashan, M. Littman, M. Taylor, J. Huang, D. Roberts, Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning, Auton. Agents Multi-Agent Syst. 30 (1) (2016) 30–59.
[57] M. Henderson, B. Thomson, S.J. Young, Deep neural network approach for the dialog state tracking challenge, in: Proc. 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGdial'13), Metz, France, 2013, pp. 467–471.
[58] M. Henderson, B. Thomson, S.J. Young, Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation, in: Proc. IEEE Spoken Language Technology Workshop (SLT'14), South Lake Tahoe, NV, USA, 2014, pp. 360–365.
[59] N. Mrksic, D.Ó. Séaghdha, T. Wen, B. Thomson, S.J. Young, Neural belief tracker: data-driven dialogue state tracking, in: Proc. 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, Canada, 2016, pp. 1777–1788.
[60] N. Mrksic, D.Ó. Séaghdha, B. Thomson, M. Gasic, P. Su, D. Vandyke, T. Wen, S. J. Young, Multi-domain dialog state tracking using recurrent neural networks, in: Proc. 53rd Annual Meeting of the Association for Computational Linguistics and 7th Int. Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP'15), Beijing, China, 2015, pp. 794–799.
[61] P.-H. Su, D. Vandyke, M. Gasic, N. Mrksic, T.-H. Wen, S. Young, Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management, in: Proc. 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGdial'17), Saarbrucken, Germany, 2017, pp. 147–157.
[62] P.-H. Su, D. Vandyke, M. Gasic, N. Mrksic, T.-H. Wen, S. Young, Reward shaping with recurrent neural networks for speeding up online policy learning in spoken dialogue systems, in: Proc. 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGdial'15), Prague, Czech Republic, 2015, pp. 417–421.
[63] A. Kumar, P. Ku, A. Goyal, A. Metallinou, D. Tur, Ma-DST: multi-attention based scalable dialog state tracking, in: Proc. 34th AAAI Conference on Artificial Intelligence, New York, NY, USA, 2020, pp. 8107–8811.
[64] J. Williams, The best of both worlds: unifying conventional dialog systems and POMDPs, in: Proc. 9th Annual Conference of the International Speech Communication (Interspeech'08), Brisbane, Australia, 2008, pp. 1173–1176.
[65] D. Griol, Z. Callejas, Discovering dialog rules by means of an evolutionary approach, in: Proc. 20th Annual Conference of the International Speech Communication Association (Interspeech 2019), Graz, Austria, 2019, pp. 1473–1477.
[66] P. Lison, C. Kennington, OpenDial: a toolkit for developing spoken dialogue systems with probabilistic rules, in: Proc. ACL'16 System Demonstrations, Berlin, Germany, 2016, pp. 67–72.
[67] E. Razumovskaia, M. Eskenazi, Incorporating rules into end-to-end dialog systems, in: Proc. 3rd NeurIPS Workshop on Conversational AI, Vancouver, Canada, 2019, pp. 1–11.
[68] D. Adiwardana, M.-T. Luong, D.R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, Q.V. Le, Towards A Human-Like Open-Domain Chatbot, 2020, pp. 1–36. arXiv:2001.09977.
[69] H. Qun, L. Wenjing, C. Zhangli, B&anet: combining bidirectional lstm and self-attention for end-to-end learning of task-oriented dialogue system, Speech Commun. 125 (2020) 15–23.
[70] B. Zhang, X. Xu, X. Li, Y. Ye, X. Chen, Z. Wang, A memory network based end-to-end personalized task-oriented dialogue generation, Knowl.-Based Syst. 207 (2020) 106398.
[71] 50+ Platforms for Chatbot Design, Development and Analytics-Less is More. Medium. https://medium.com/voice-tech-podcast/50-platforms-for-chatbotdesign-development-and-analytics-less-is-more-a8677ab2b7d5, (Accessed 18 April 2021).
[72] D. Griol, L. Hurtado, E. Segarra, E. Sanchis, Managing unseen situations in a stochastic dialog model, in: Proc. AAAI Workshop Statistical and Empirical Approaches for Spoken Dialogue Systems, Boston, USA, 2010, pp. 25–30.
[73] L. Hurtado, D. Griol, E. Segarra, E. Sanchis, A stochastic approach for dialog management based on neural networks, in: Proc. 9th Int. Conference on Spoken Language Processing (Interspeech'06), Pittsburgh, USA, 2006, pp. 49–52.

Pablo Cañas has a M.Sc. in Data Science at École Polytechnique Fédérale de Lausanne, BSc in Computer Science at Universidad Carlos III de Madrid. Pablo has experience in industry as a software engineer at Amazon and as a machine learning engineer at a fintech startup, and in research institutions, such as the University of California San Diego and CERN. He is specialized in deep learning, NLP and computer vision.



Dr. David Griol obtained his Ph.D. degree in Computer Science from the Technical University of Valencia (Spain) in 2007. He has also a B.S. in Telecommunication Science from this University. He is currently Associate Professor in the Department of Software Engineering at the University of Granada (Spain). He was professor at the Computer Science Department at the Universidad Carlos III de Madrid from 2008 to 2019. He has participated in several European and Spanish projects related to natural language processing and speech technologies. His research activities are mostly related to the development of statistical methodologies for the design of conversational interfaces. His research interests include dialog management and optimization, corpus-based methodologies, user modeling and simulation, adaptation and evaluation of spoken dialog systems and machine learning approaches. He has been a visiting researcher in University of Ulster (Belfast, UK), Technical University of Liberec (Liberec, Czech Republic), University of Trento (Trento, Italy), University of Ulm (Ulm, Germany), Technical University of Berlin (Berlin, Germany) and Telecom ParisTech (Paris, France).



Dra. Zoraida Callejas is Associate Professor in the Department of Software Engineering at the Technical School of Computer Science and Telecommunications of the University of Granada (Spain). She completed a PhD in Computer Science at University of Granada in 2008 and has been a visiting researcher in University of Ulster (Belfast, UK), Technical University of Liberec (Liberec, Czech Republic), University of Trento (Trento, Italy), University of Ulm (Ulm, Germany), Technical University of Berlin (Berlin, Germany) and Telecom ParisTech (Paris, France). Her research activities have been mostly related to speech technologies and in particular to the investigation of affective dialogue systems. She has participated in numerous research projects, and is a member of several research associations focused on speech processing and human-computer interaction.