Dialogue management in conversational systems: a review of approaches, challenges, and opportunities

Hayet Brabra, Marcos Baez, Boualem Benatallah, Walid Gaaloul, Sara Bouguelia, Shayan Zamanirad

Dialogue Management in Conversational Systems: A Review of Approaches, Challenges, and Opportunities

Hayet Brabra, Marcos Báez, Boualem Benatallah, Walid Gaaloul, Sara Bouguelia, Shayan Zamanirad

***Abstract*—Attracted by their easy-to-use interfaces and captivating benefits, conversational systems have been widely embraced by many individuals and organizations as side-by-side digital co-workers. They enable the understanding of user needs, expressed in natural language, and on fulfilling such needs by invoking the appropriate backend services (e.g., APIs). Controlling the conversation flow, known as *Dialogue Management*, is one of the essential tasks in conversational systems and the key to its success and adoption as well. Nevertheless, designing scalable and robust dialogue management techniques to effectively support intelligent conversations remains a deeply challenging problem. This article studies dialogue management from an in-depth design perspective. We discuss the state of the art approaches, identify their recent advances and challenges, and provide an outlook on future research directions. Thus, we contribute to guiding researchers and practitioners in selecting the appropriate dialogue management approach aligned with their objectives, among the variety of approaches proposed so far.**

***Index Terms*—Conversational System, Dialogue Management, Dialogue State Tracking, Dialogue Policy.**

I. Introduction

With the noteworthy growth of digital data and services, users need technologies that offer quick access to such data and facilitate the interaction with these services. Conversational systems which now become more and more proliferating in industry, academia, and society, are excellent examples of these technologies. Indeed, they rely on human-friendly interfaces, using natural language (e.g., text or voice), to access complex cognitive backend, which tries to understand user needs and serves them by invoking the proper services. Notable examples are Siri, Google Assistant, Amazon Alexa, Baidu, and Cortana which are with us all the time.

Conversational systems are generally categorized into two classes [1]: (1) task-oriented systems and (2) non-task-oriented systems. Non-task oriented dialog systems focus on open domain conversations with no specific task to accomplish like chit-chat messages. In contrast, task-oriented systems allow users to accomplish tasks (e.g., maintain schedules, organise projects, booking flight) using information provided by users during conversations. This article focuses on task-oriented conversational systems. No matter what the category, one of the critical aspects in the development of conversational systems is the design of a dialog management component that is able to ensure robust, intelligent, and engaging conversations [2], [3]. This is because of the paramount and diverse roles that it plays, which especially include (i) tracking of information that is entered implicitly or explicitly by users, (ii) understanding conversation context and resolving ambiguity, (iii) controlling the conversation flow between users and system, (iv) communicating with external services/databases and finally (v) identifying system actions to fulfill the ultimate user goal. Despite many years of research, both the scientific and industrial world still struggle to understand which dialog management (DM) approaches are suitable for the type of conversational system they seek to offer [4]. Moreover, designing effective dialog management remains a deeply challenging problem [3], [5].

In this article, we seek to accelerate the fundamental understanding of DM from an in-depth design perspective. Previous surveys mostly focused [1], [6], [7], [8], [9] on (i) providing background related to understanding conversational systems (e.g., overview of the entire system architecture) and (ii) exploring and analyzing their design models/techniques from a general perspective (e.g., explaining the application of Markov chain model) without considering particularly the dialogue management component. Other aspects that are studied deeply in recent years are related to the dialogue corpora [10], evaluation [11], and user interface features [12] of conversational systems. The closest to our work is the short survey by Harms et al. [4], where they survey dialog management approaches, but taking a different analysis point of view to focus on capabilities (e.g., learning, error handling) and practical aspects (e.g., tool availability, dependencies) mostly in commercial solutions. In contrast, to Harms's and the aforementioned works, the present survey takes a comprehensive look at past and emerging design approaches to DM, following an analytical framework that considers relevant design aspects of DM.

We analyze and characterize existing approaches for DM, based on the analysis of a wide range of literature and a selection of bots development platforms. We discuss their advances and limitations and we define opportunities for future research directions. We noted that each DM approach has its strengths and weaknesses and are tailored, in most cases, to be used in a particular context – all of which are relevant design considerations for choosing a dialog management approach. Hence, in characterizing current approaches in terms of meaningful dialog management design attributes, our work would contribute to guiding researchers and practitioners in selecting

H. Brabra, M. Báez and S. Bouguelia are with LIRIS – University of Claude Bernard Lyon 1, Villeurbanne, France .
E-mail: {hayet.brabra, marcos.baez, sara.bouguelia}@univ-lyon1.fr
B. Benatallah and S. Zamanirad are with the University of New South Wales, Sydney, Australia.
E-mail: b.benatallah@unsw.edu.au; shayanz@cse.unsw.edu.au
W. Gaaloul is with the Computer Science Department, SAMOVAR, Telecom SudParis, Evry, France.
E-mail: walid.gaaloul@telecom-sudparis.eu

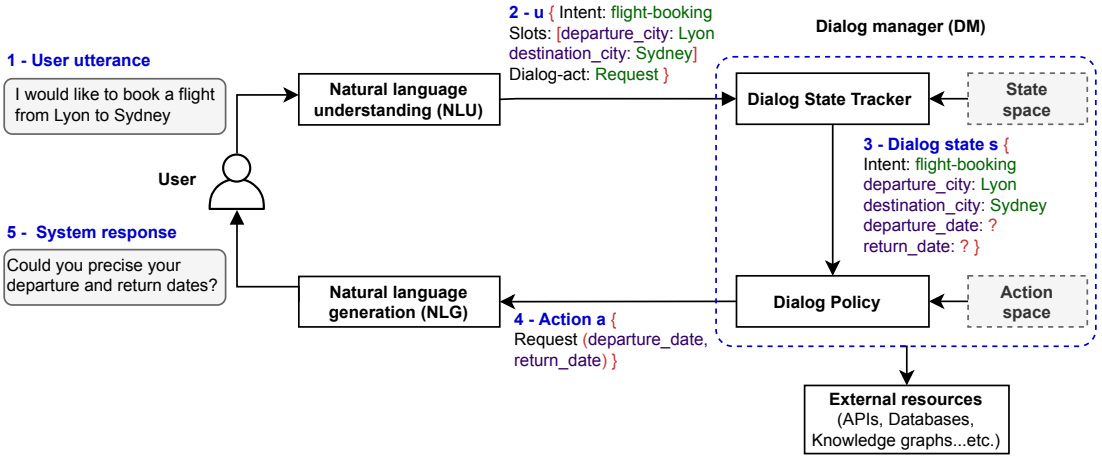


Fig. 1: Dialog manager and its information flow

the appropriate approach aligned with their objectives, among the variety of DM approaches proposed so far.

II. Problem Fundamentals and Dimensions

This section first articulates the fundamentals to understand dialogue management and presents then the set of dimensions that we consider to analyze the different approaches.

A. Dialogue Management

Fig.1 depicts the role of the dialogue management component (or simply dialogue manager) and its interaction flow with the common conversational components according to the pipeline architecture [1], which has been widely adopted by most traditional conversational systems and still used underlying modern commercial and research systems [13]. In its simplest form, an interaction with a conversational agent involves a user input in natural language or user utterance, followed by the system response, in what is called a conversation turn. In each turn, the conversational system first uses the Natural Language Understanding (NLU) component [14] to convert the user utterance (e.g., *"I would like to book a flight from Lyon to Sydney"*) into a structured representation **u** that is typically encoded using three main types of information: Intent, entities and dialogue acts. An intent refers to users' goal/task, which a conversational system should be able to respond to (e.g., *"flight-booking"*). Entities, often called slots, describe the parameter(s) needed to fulfill the intent (e.g.,*"Departure_city: Lyon"*). Dialogue acts are hidden actions in user utterances to indicate whether the user is making a statement, asking a question, among others [14] (e.g., *"Request()"*).

All this information is fed into the so-called Dialogue State Tracking (DST) model which is a function that maps a given structured representation **u** into a suitable state, known widely as a dialogue state **s**, from the state space of the conversational system. The dialogue state **s** keeps track all information that the conversational system requires to make its decision about how to answer the user. For example, in the flight-booking domain, the dialog state might indicate the user's intent, such as flight-booking, its main slots such as a preferred travel class, departure_city, etc. , and which information has been confirmed and which not. To maintain the dialog state, the DST model may leverage information entered explicitly by the user or implicitly as a result of exploiting the conversation context.

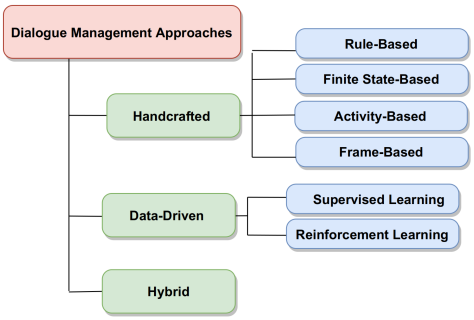


Fig. 2: Dialogue management approaches

Based on the dialogue state, the decision of the next system action is taken by a second model recognized as Dialogue Policy (DP). DP model can be viewed as a function that maps a given dialogue state into an appropriate action (usually represented as a dialogue act) from the action space of the conversational system. For example, in the flight-booking domain, the action space might be constrained to four actions: Request, Confirm, Query, Execute. *Request* asks the user for missing information (e.g., Request (departure_date, return_date). *Confirm* clarifies the intent and/or the slots with the user (e.g., Confirm (departure_city="Lyon")). *Query* gets a list of flights and informs the user (e.g. Inform(flight="TR3253", departure_time="10am")). *Execute* submits a booking request once all information is provided. Finally, based on the outcome of DP (e.g., Request (departure_date, return_date)), the Natural Language Generation (NLG) component [14] produces an appropriate response (e.g., *"Could you precise your departure and return dates?"*) to the user.

Although we have adopted the pipeline architecture as a reference in this paper because of its simple interpretation and nuanced distinction of components, it is worth noting that there are a wide variety of system architectures that have been proposed both in research labs or industry. Under these architectures, as will be explained later, we can find that (i) DST and NLU, (ii) DST and DP, (iii) or even all the four components are merged into one module.

B. Analysis Dimensions

In this article, we study the dialog management component from a design standpoint, analyzing the various approaches that have emerged in the literature. In doing so, we consider the following dimensions that guide our discussion and analysis:

- **Dialogue state:** This dimension identifies how dialog state is represented, what information it includes, whether it is handcrafted by engineers or learned from data.
- **Dialogue state tracking**: This dimension identifies how the dialog state is determined.
- **Dialogue policy:** This dimension identifies how the next action is determined.
- **Context support:** Context is defined as any pertinent information that is leveraged to understand user needs and satisfy them appropriately. This dimension focus on analysing the extent of context support in DM using three contextual features, namely, *Conversation history*, *User profile*, and *External knowledge*. It assesses how well these features are covered and how they are supported.

III. Dialogue management approaches

Based on the analysis of a wide range of literature and a selection of bots development platforms, we uncovered three main approaches that have been adopted to implement the DM models (i.e., DST and DP). These approaches are illustrated in Fig. 2 and summarized in Table I and II according to the dimensions of analysis.

A. Handcrafted approaches

Handcrafted approaches rely on programs or/and models that are fully specified by developers or domain experts to track the dialogue state and define the policy. We distinguish between four kinds of handcrafted approaches, namely Rule-Based, Finite State-Based, Activity-Based, and Frame-Based.

1) *Rule-Based*

The most traditional approach to DM is to adopt handcrafted rules [15], [16], [17], [18], [19], [20]. In this approach, bot developers define the dialogue state as well as the policy by encoding a set of rules. The simplest modeling of these rules is structuring them in the form of pattern/response pairs, which perform NLU, DM, and NLG tasks at once by taking the user utterance and producing the corresponding response. In this respect, various languages have been adopted to specify such rules. A notable example is the Artificial Intelligence Markup Language (AIML) [21]. AIML is built around two core units: categories and topics. Categories are blocks of rules, each one consisting of a (i) pattern defining user input (e.g. "Hi bot"), and (ii) template defining the corresponding response (e.g. "Hi human"). Topics, on the other hand, are collections of categories. ELIZA [22], PARRY [23] and ALICE [21] are the first generation of conversation systems that leverages this language. The main drawback of AIML is that it is too verbose, as it requires a lot of rules to perform even simple tasks [24]. Other rule-based languages include Rivescript[1] and Chatscript[2]. They provide an easy-to-understand syntax and extensive additional features compared to AIML.

[1]https://www.rivescript.com/

[2]https://github.com/ChatScript/ChatScript

In the above approaches, the DST and DP are reduced to a simple pattern matching that is bound directly to user utterance. Alternatively, more sophisticated rules [25], [20] consist of preconditions (PRE) and actions (POST) that make use of NLU outputs (i.e., generally is the 1-best NLU interpretation results [26]) to track the dialogue state and decide on the next action to perform. An important advantage of a rule-based approach is that it is easy to implement and does not require any training data, which is a benefit for quick bootstrapping. It also offers an easy way for developers to incorporate domain knowledge. Despite that, they lack flexibility and require considerable effort from developers to encode rules. As the number of rules grows, finding overlaps and conflicts between rules causes a laborious maintenance cost.

Context support

Conceptually, rule-based approaches have the ability to incorporate information from user profiles (e.g., gender, location, favorite songs, etc.) either to update dialogue state with more personalized information and decide system action that will adequately satisfy their preferences. However, user profile information has been only supported by a few approaches [15], [18]. Furthermore, most of the rule-based approaches act only on the last conversation turn when deriving their decisions, making them only tailored in pair-wise utterance exchanges. Whereas, a handful of approaches [27], [15], [16], [20] has considered dialog history to handle missing information in the dialogue state. Both features (i.e., user profile and dialog history) have been hard-coded into the DST and DP models which naturally come at a high development cost and limited scalability.

2) *Finite State-Based*

In this approach, the DST and DP are exposed as a single module modeled using a finite-state model that provides a set of a predefined sequence of steps representing the dialogue state at any point during the conversation [2], [28], [29]. Each state is restricted to a prescribed number of transitions to other states and defines the set of actions that the conversational system can/should perform in a given situation. For example, it may ask the user to answer a question or execute the task that the user wants. Transitions are triggered as a result of recognizing a pattern that matches a user utterance. Fig. 3 shows an example of a finite state-based DM modeled using a state machine, for a "flight booking" scenario. For example, when the system is in "One-way trip" state, there are two transitions available based on what the user will answer: (i) "Yes" moves to "Get final confirmation" state, (ii) "No" moves to "Ask for return date". Indeed, the finite state-based approach generally involves handcrafted rules to determine the current state of dialogue and move between states. Notable research systems following this approach include AVA [30], IRIS [31], Devy [32] and DIASY [33] and DialogOS [34].

Generally, the finite state-based approach has the same advantages and limitations as a rule-based. However, it comes with other shortcomings such as versatility and robustness in situations where a user does not follow predefined sequences of states [24]. Considering the FSM example (Fig. 3, if a user's initial utterance carries all the required information (e.g. *"I want to book a one-trip flight from Lyon to Sydney for the*

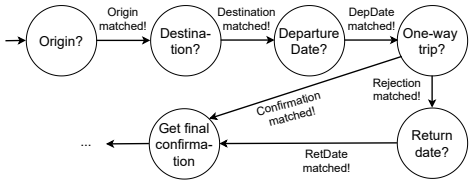


Fig. 3: DM model defined using a finite state machine (FSM)

next Monday") or at least part of it (e.g. *"I want to book a one-trip flight to Sydney?"*), the dialogue manager cannot handle such a situation because it is programmed to ask for the information one by one by following the defined sequence of states (e.g. origin → destination → date→ type of trip).

Context support

In FSM-based category, IRIS [31], Devy [32] and DIASY [33] are the notable approaches that provide context support through modeling of conversation history and/or user profile features. These features are used to infer the missing slot values in the dialog state. DAISY supports the dialog history by storing all steps that it may take and already fulfilled slots to solve a typical data science task, (e.g. model retraing). This allows DAISY to solve a similar task in the future without the data scientist involvement. IRIS supports the conversation history by defining a set of named variables to save the result of each dialog task for future use by other tasks. In doing this, IRIS aimed to support the so-called *anaphora*, one of the most important interaction patterns that characterizes human language, which denotes expressions that depend on previous expressions. For example, after executing a task that has just returned a result (e.g. sum of two numbers), a user can ask IRIS to "multiply that in 2" where "that" refers to the output of the previous task. Instead of hard-coding history variables within the FSM, IRIS dedicates an internal API to dynamically add new named variables as the conversation progresses. Devy relies on a domain-specific model implemented using a pull-based architecture to support both dialog history and user profile features. User profile comprises development-related information (e.g., active project, issues, code reviewers, etc.) extracted from the developer activities and computer during the conversation. The aim is to infer the required slots for performing development actions without developer involvement.

*3) **Activity-Based***

The activity-based approach allows the development of DM model (i.e. the DST and DP is exposed as single module) by leveraging the basic concept of workflows, namely activities, triggers and actions. While the FSM approach is a declarative approach that provides a high-level specification of the dialog manager *behavior* in terms of possible states that may occupy during the conversation, the activity-based model is a procedural approach that provides a concrete implementation of dialog manager by precisely specifying the *workflow* that it may go through during the conversation. Various platforms including Chatfuel[3], FlowXO [4], and ManyChat[5], facilitate the definition of these workflows by providing design canvas along with visual elements (e.g. Carousel, Quick Reply, Text, etc.). Recent research work leveraging on the activity-based approach is [35] which relies on a business process model described using BPMN notation. This model is then transformed into DM rules defined in AIML to build a conversational system that guides the process actors through the process steps.

Context support

Conversation history and user profile are the main considered features by the activity-based approaches adopted by ManyChat, Chatfuel and FlowXO. The aim is either to fill missing slots or decide more personalized actions, e.g., provide a recommendation about a product. With regard to the user profile feature, there are two sources from which related information can be gathered, namely user-bot conversations and the social media profiles (e.g. Facebook) of the user. Furthermore, to define which information needs to be collected from both conversation history and user profile, the common adopted approach is that the bot developer has to manually create a set of attributes using the provided GUIs or APIs.

*4) **Frame-Based***

The frame-based approach relies on a domain ontology that defines a set of frames [29], each specifying the required information that the conversational system is designed to acquire from the user in order to fulfill a given dialog task (e.g., play music). A predefined frame may include a dialog domain (e.g., Music), an intent (e.g., Play Music), and a collection of required slots (e.g., Genre, Musician). Historically, the frame-based approach was first introduced in 1977 in the influential Genial Understander system (GUS) [36]. Other traditional systems include WITAS [37] and COMIC [38]. Under these systems, the dialog state represents the current state of the frame (e.g., which slots have been filled and which haven't), whereby the DST model determines it by exploiting NLU outputs that are usually obtained using handcrafted rules. The DP, on the other hand, is quite simple as it lies in asking questions until the completion of the whole frame and then reporting back the results of the action associated with the frame to the NLG. This latter relies on a template-based generation to produce the final answer to the user [29]. The growing interest in using the frame-based approach, on the other hand, pushed the industry to create VoiceXML, which has become later one of the W3C voice-related specifications. VoiceXML has been widely adopted to implement frame-based dialogs within the speech industry and supported with opensource and commercial platforms [6]. Compared to FSM and activity-based approaches, the frame-based affords more flexibility thanks to its ability to efficiently process over-informative inputs from users while allowing them to fill in the slots in different orders and different combinations. This, on the other hand, requires sophisticated DP algorithms in order to determine the next system action or question based on a set of features, including mainly the user's previous utterance, slots to be filled, and a number of priorities devoted to dialog control. In addition, considerable testing efforts are needed to ensure that the system would not ask an inappropriate question under any conditions unforeseen

[3]https://chatfuel.com/
[4]https://flowxo.com/
[5]https://manychat.com/
[6]https://www.voip-info.org/voicexml/

at design time. Despite that, the frame-based approach is still until now underlying modern conversation systems like Apple's Siri, Amazon's Alexa, and the Google Assistant [29]. In addition, there are many commercial systems that provided similar capabilities to VoiceXML to implement frame-based DM models, including the user-definable skills in Amazon Alexa, the actions in Google Assistant and DialogFlow. The difference compared to the traditional frame-based approaches lies in using machine learning approaches to NLU tasks and adding control models in addition to the frames. The control models can be designed by mean of the FSM or workflow models, which may avoid bot developers to specify complex dialog policies.

Context support

Context in traditional frame-based systems is supported mainly for resolving anaphora pattern (e.g., GUS [36], WITAS [37]) and inferring slot values from previously encountered slots (e.g., GUS). Both conversation history and user profile are the main features that have been considered for context understanding. They supported using reasoning procedures and models handcrafted by bot developers. Modern systems, including Amazon Alexa, the actions in Google Assistant and DialogFlow, on the other hand, offer developers with more sophisticated, easy-to-use and powerful IDEs and libraries to support the management of context and expand it to integrate external knowledge, such as knowledge graphs (e.g., Home Graph in actions for Google Assistant) in addition to conversation history and user profile.

B. Data-driven approaches

In contrast to handcrafted approaches where the DM logic has to be defined by hand, data-driven approaches were proposed to learn the dialog state and policy from data. They involve mainly machine learning (ML) approaches, including supervised learning (SL) and reinforcement learning (RL). The supervised approach learns from a pure corpus of labeled data, whereas reinforcement approach focuses on optimizing the learning by a trial-and-error process governed by a series of reinforcements (i.e., rewards or punishments).

1) Supervised learning

A variety of SL models have recently been applied to the DST, DP, or both as a single module exposed either independently from NLU and NLG components or jointly leading to the emergence of so-called end-to-end conversation systems.

Dialogue state tracking

The earliest SL approaches to DST rely on statistical learning algorithms, including conditional random fields [39], [40] and maximum entropy models [41], [42], [43] which often depend on the NLU outputs to define the dialog state in term of slot-value pairs. These approaches heavily rely on handcrafted features to learn dialog state representations. More recently, a wide spectrum of research has been relying on neural-based approaches, which started to receive more attention, especially with the adoption of deep learning (DL) models that have significantly contributed to DST performance improvement. Most of these approaches consider merging NLU and DST into a single model that acts directly on user utterances to update the dialog state. One benefit of this merging is that it removes the information loss and error propagation at the NLU stage and requires fewer labeled data since there is no need to learn independent parameters for each model. Notable DL models were initially adopted to DST are multi-layer perceptrons (MLP) [44], [45], recurrent neural networks (RNN) [46], [47], and convolutional neural networks (CNN) [45]. They are used to learn feature representations for user/system utterances as well as the associated slots/values. An illustrative example (refers to Fig. 4) is the Neural Belief Tracker proposed by Mrkšić et al. [45] which provides a binary decision for each slot-value pair candidate. The model has been tested with tow DL models (MLP and CNN), both of which build upon pre-trained collections of word embedding vectors and outputs embedding for each input (i.e., the user utterance, the dialogue acts related to the last system output, and a single candidate slot-value pair). The three obtained embeddings then interact among themselves in order to produce the interaction summary vectors which then go through a binary softmax layer to produce the final decision about the candidate slot-value pair.

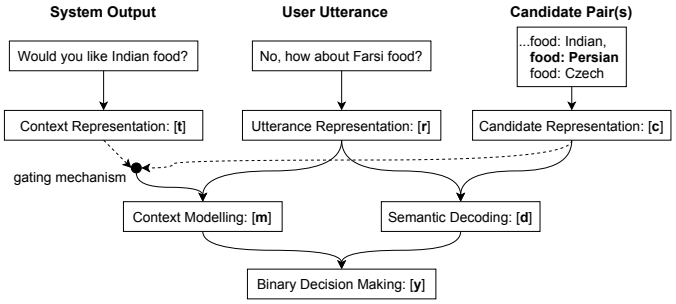


Fig. 4: Architecture of the NBT Model [45]

More effective DL approaches to DST immensely involved memory-enhanced NN architectures. Gate RNNs, such as Long-Short Term Memory (LSTM) [48], [49], [50] and the networks based on Gated Recurrent Unit (GRU) [50], [51] are adopted in order to capture long-term dependencies in dialog history, resulting in an accurate update of dialog state. MemN2N known as a NN with a recurrent attention mechanism over a large external memory is another memory-based approach adopted to the DST task framed as a machine reading problem. Having an attention mechanism allows the model to consider only the most important information from the dialogue history while ignoring others. The main advantage of memory-based approaches is the fact that the dialogue state is affected by the long-term features of previous utterances. Therefore, the resultant tracking results are expected to keep the consistency of the topic with previous conversations while efficiently exploiting context to gather information for the dialog state.

More recent DL models [52], [53] adopted graph neural networks with attention mechanisms to incorporate slot relations (e.g., similarity, co-occurrence) in DST, as a solution to alleviate the data sparsity problem. In [52], the authors introduced graph attention matching networks that encode the slot relation graph and user/system utterances, and output slot representation vectors. These vectors along with previous conversation states are then taken by a recurrent graph attention network (GAT) with GRU units to update the dialog state.

GAT represents a variant of graph neural networks, which in turn are deep neural networks associated with graphs.

The above DST approaches can fall into two categories: Predefined ontology-based [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [52] and Open-vocabulary candidate-generation [50], [54], [53], [51]. The former assumes that a predefined ontology is provided in advance to define all slots and their values in each domain. Thus, the dialogue state is represented as a binary/multinomial distribution over the value set for each slot, pre-specified in the ontology. While these approaches have been shown to be accurate as they reason over a known candidate set of each slot, their applicability in practice is still threatened and depends on the ontology coverage. Since the set of possible values is typically dynamic (e.g., movies or usernames) and unbounded (e.g., date or location), defining such an ontology that covers all entries is challenging [55], [54], [53]. Open-vocabulary approaches, on the other hand, estimate the slot value candidates from conversation history and/or language understanding outputs, without any predefined ontology. This provides a key step toward a DST with zero-shot generalization, whereby adding new intents, slots or even domains do not require the need for collecting new data or even retraining. Such an objective is now dominating the research attention and was the focus of the recent dialogue system technology challenge (DSTC8) [56] .

Dialogue policy

There are two approaches to applying SL for the DP. The first one mainly implements a DP model as a pipelined module, trained independently of DST and NLU modules. In this approach, the DP often takes as input the dialogue state from the DST model or in some cases acts directly on the NLU results to output the next system action. The most widely used SL models to implement a DP are NNs,[57], Bidirectional LSTM (BLSTM), CNN or a combination of the two above (i.e., BLSTM/CNN) [58], [59]. For example, the model in [58], as shown in Fig. 5a, represents a DP as a NN with one hidden layer and an output layer consisting of two softmax partitions and six sigmoid partitions. Regarding the softmax outputs, the first one is for predicting the dialogue act among five dialogue acts (request, offer, confirm, select, bye) while the second for predicting the associated slots (e.g., price-range, area), whose values require information from the user. The sigmoid partitions are for offer slots, whose values are made by the system.

The second approach is to implement a DP as an end-to-end model that reads directly from a user utterance and produces a system action. The sequence-to-sequence model, also known as *Seq2Seq*, is the main model used in this approach. Such a model is based on the encoder-decoder architecture which takes a sequence as input and generates another sequence as output. In a conversational system, the source sequence is a user utterance along with a dialogue history, and the target sequence is a corresponding action (e.g., API call, database query). The seq2seq model is initially implemented using RNN with LSTM cells, where the hidden state of RNN is utilized as the representation of a dialogue state. This model is later augmented with an attention mechanism to improve its ability to handle long-term dependency [60], [61]. Using end-to-end memory networks is another alternative to build end-to-end models for a DP. A notable example is the DP model proposed by Zhang et al [3]. The model, as shown in Fig. 5b, relies on three memories (i.e., slot-value memory, external memory, control memory) with shared read/write functions and augmented with an attention mechanism. It takes as input the embedding of the current user utterance and the previous system response and predicts the next system action in the form of a dialogue act like "*query(cuisine=chinese)*". Compared to the pipelined policy, the key advantage of the end-to-end approach is inferring the representation of the dialog state which avoids the design of its related features. However, this requires a lot of training data that may be expensive to collect. Moreover, in both approaches, since the DP is trained using supervised learning, the effects of selecting an action on the future course of the dialogue are not considered, which may lead to a behavior that is not necessarily optimal. In addition, the DP does not expose the ability to recover from its own mistakes or users' feedback. Such a feature can be added to the DP by training it using RL, as will be explained in section III-B2.

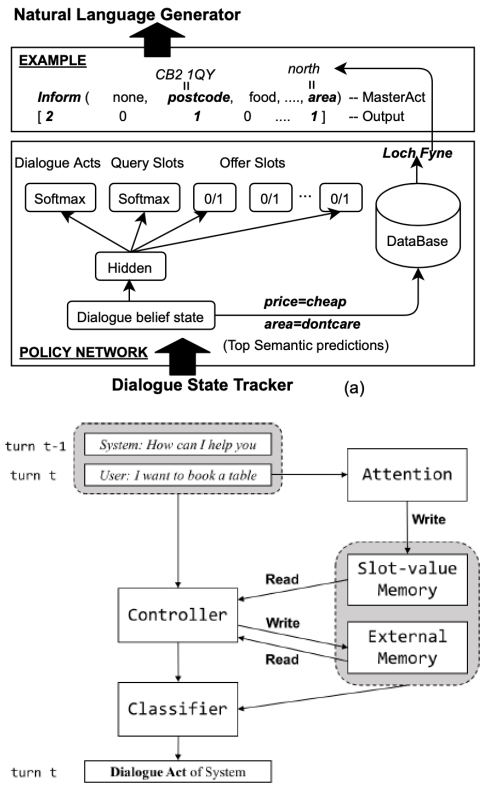


Fig. 5: Dialogue policy: (a) As DST-depend model [58]; (b) As end-to-end model [3]

Context support

Conversation history has been widely considered by SL approaches as a key feature for context understanding. Its support varies from one approach to another. While some SL approaches [40], [41], [42], [44], [45] assume that considering the last conversation turn preceding the current one as conversation history is enough to understand the context, others [39], [43], [46], [47], [55], [3], [49], [62] leverage on the whole

history from the first conversation turn up to the last one. There are three commonly used methods for modeling conversation history. The first consists of concatenating all the utterances in the dialogue [48], [61] which represents one of the reasons that result in increased computation time. Addressing this issue, the second method [39], [40], [43], [46], [47], [55], [3], [49], [62] lies in capturing particular features such previous user/system dialogue acts or by considering only the previous dialog state. For both methods, SL models including Memory networks and embedding techniques have been shown very effective for modeling history features and reasoning over them. The third method has been recently adopted by [63], which leverages the graph structural information in dialogue history to accurately capture its semantics. Herein, the off-the-shelf Spacy tool is used to extract the dependency relations among the dialogue history words and a new recurrent cell architecture is introduced to enable representation learning on graphs.

In addition to the conversation history, some of the recent SL approaches [53] go a step forward by effectively leveraging knowledge on conversation domains. This has been done by considering potential relationships between slots across different conversation domains so that their values can be transferred between similar slots (e.g., the *address* slot value in a restaurant domain could be transferred to the *destination* slot for a taxi domain). Within these approaches, Graph attention networks, knowledge graphs and embedding methods are used to automatically build relationships between slots. On top of these models, inference mechanisms are applied to infer relevant knowledge that is required for DST and DP tasks. Furthermore, some approaches [64], [65] attempted to leverage further domain-specific or general knowledge extracted from either structured or unstructured data. For instance, the work of [65] proposes to encode user utterances using feature vectors with associative knowledge, exploiting information extracted by inference on a knowledge graph built from Wikidata. These feature vectors are then taken by a neural-based DST model to improve its predictions over slot-values pairs that are not mentioned explicitly in user utterances.

2) Reinforcement learning

RL approach treats DM as an optimization problem, whereby its performance is continuously improved over time through experiencing with users. RL-based DM is typically modeled as Markov Decision Process [66], [13], [2], where the dialogue manager acts as an agent traveling through a network of dialogue states interconnected with transition probabilities. In simple term, at each conversation turn t, the dialogue manager is in a given state (e.g., S1), takes action (e.g., request (origin_city, date, time)), transitions to a new state **s'** (e.g., S2) with a probability (e.g., P= 0.80) and receives a reward (e.g., R = 200). Typically, the reward is defined using a function that captures a set of dimensions, including accomplishment of the task, user satisfaction, efficiency of interaction, dialog duration, etc. For instance, in the movie-recommendation dialogue system proposed by Zarandi et al. [67], the system gets a large positive reward (+10) if it correctly fulfills the user request, a large negative reward (-12) otherwise, and a small penalty (i.e., negative reward) for confirmation (-3) and elicitation (-6) questions to prevent the system re-confirming numerous times than needed. In addition, it receives a small positive reward (+1) if the user leaves the conversation after a successful execution and a medium negative reward(-5) otherwise.

Another important approach to model RL-based DM is the Partially Observable Markov decision process (POMDP). Compared to MDP that relies on the assumption that the dialogue state is fully observable (i.e., is always known to the system with certainty), POMDP caters for unobserved dialogue states [68], [69], [70]. This makes it able to deal with uncertainties in user utterances [71], [68]. More particularly, the dialogue state is defined as a distribution over all possible states, including the wrong ones arising from an incorrect interpretation or misunderstanding of user utterances.

Dialogue state tracking

Past approaches to DST in MDP and POMDP models have widely focused on applying Bayesian inference (BI) to determine the dialogue state. More precisely, given an existing dialogue state, the last system action and a set of observations (for instance, the N-best lists of user dialogue acts), the most basic dialogue state trackers (e.g., [72], [73]) enumerate all possible dialogue states and scoring them according either to a probability formulation proposed by Kaelbling et al. [74] or its variants. However, it has been found that these approaches have quickly led to intractable issues resulting from a large set of states [69], [26]. Addressing these concerns, approximations methods including N-best [75], [76] and Factored Bayesian network [77] [78] approaches, have been proposed, exploiting domain-specific properties of dialogue task to reduce the tracking state space. Other DST alternatives include either simple rules [79] as in handcrafted conversational systems explained earlier, or probabilistic rules [27] that are defined as domain-independent rules augmented with basic probability operations. Compared to the previously explained rules (refer to Section III-A1), probabilistic rules take as input all hypotheses suggested in the N-best interpretation list of the user utterance and output a dialogue state in the form of a distribution over possible states, i.e., slot-value pairs. To update such a distribution, these rules exploit dialogue acts related to the last user utterance and preceding system action. Although considering all N-best list interpretations can deal either with uncertainties or recognition errors if happen, these probabilistic formulas designed by hand, require careful tuning and does not benefit or learns from dialogue data [26].

More recently, deep learning models [80], [81], [82] have been applied, which use a compact representation of a set of features learned automatically to represent the dialogue state. For instance, at each conversation turn the dialogue state in [81] is represented with a feature vector, consisting of the following: (i) act and slots corresponding to the user action, act and slots corresponding to the last system action; (iii) a bag of slots corresponding to all previously filled slots; (iv) other features like current turn count, etc.

Dialogue Policy

Whether the DM is modeled by MDP or POMDP, the ultimate objective of RL is to find the optimal policy that maximizes the expected (discounted) cumulative reward. There

are two popular classes of RL algorithms [66]: value-based and policy gradient methods.

Value based methods are based on the idea of modeling the optimal policy using the so-called Q-value function $Q^{\pi}(s, a)$ [7]. More precisely, the Q-value function provides the average discounted long-term reward if a specific action a is taken given a dialogue state s and then a policy π is followed thereafter. Basic approaches [83], [84], [85] known as exact algorithms rely on observing the set of states, actions, and rewards and then compute the value function to estimate the optimal policy. However, it has been shown that exact algorithms quickly lead to an intractable problem. Therefore, sophisticated approaches [79], [80], adopting approximation algorithms, have been proposed to overcome this issue. The underlying idea is to use a set of state features to generalize the estimation of the Q-value at states that have similar features. For example, Xu et al. [79] propose a dialogue policy model that makes use of a neural network function [86] to approximate the Q-value function.

Policy gradient methods try to optimize the policy directly, without having to learn the Q-value function. Here, the policy itself is directly parameterized by $\theta \in R$ which is the corresponding coefficient vector to be learned from data [7]. REINFORCE [87] is a popular example of such a policy gradient method, which has been already used to learn a task-oriented dialogue policy that adapts to new user behaviors unseen during training [88].

In a recent empirical study that compares the most representative RL algorithms using the PyDial framework [66], it has been shown that value-based methods have a higher learning rate in domains with a small action and state spaces, whereas policy-based methods can scale robustly to larger state and action spaces [66]. Furthermore, for having efficient policy learning, the earlier implementation of both methods required manual features for the state and action-space representation and involved either using summary state and action spaces or restricting the set of possible actions which in turn requires expert domain knowledge [89]. Consequently, many research works [90], [91], [89], [81] have applied deep learning models to enhance the performance of both RL methods, without resorting to feature-engineering. This leads to a new approach known as deep reinforcement learning (DRL). For instance, in value-based methods, the core idea of the DRL is to adopt deep neural models such as MLP, CNN, or RNN to approximate the Q-value function [13]. These improvements, however, made RL methods only able to support simple conversations because they basically operate in flat state-action space, hence they have been known as flat RL methods. In fact, according to numerous studies [92], [93], flat RL methods are not able to learn well and be data-efficient in large domains and especially where the conversation tasks are complex.

These challenges motivate the study of the so-called Hierarchical RL [94] which is now being actively explored in order to avoid the curse of state/action space. Fundamentally, HRL provides a principled approach for learning dialogue policies over complex conversation tasks by decomposing complicated conversation tasks (e.g., travel planning) into a sequence of sub-tasks (e.g., book-flight-ticket, book-hotel, etc.), and managing those sub-tasks at the same time. In such a setting, the dialogue policy can be designed across multiple layers following the task hierarchy levels. For instance, in the recent work of Peng et al [92], the dialogue policy has two layers: a top-level layer selects which subtasks to complete, and a low-level layer chooses primitive actions to execute the selected subtask. For the whole dialogue management problem, it is mostly framed using the mathematical framework of options over MDPs, where options generalize primitive actions to higher-level actions.

Although the RL-methods, including DRL and HRL have superior potentials than traditional methods, including SL and rule-based, they suffer from data requirement problem since they require many interaction samples to optimize the policy learning. Interacting with simulated users to obtain these samples has been adopted widely, providing an inexpensive solution compared to the interaction with real humans. Hence, extensive research has been made toward building realistic user simulators that intend to approximate real users' behaviors in conversation. Diverse approaches have been emerged, including agenda-Based (e.g, [95]) and data-driven (e.g.,[96]. Consequently, many user simulators have been publicly made available, including [97], [98] and ones instantiated in open-source dialog system development platforms such as PyDial and CONVLAB as interesting examples.

Despite the immense effort on user simulation, building a human-like simulator still a key challenging task [99]. In addition, simulated users may not cover entirely the behavior of end-users will be using the system. Thus, using a hybrid approach, where the DP is initially trained with simulated users and then fine-tuned with human users, has been shown convenient in many initiatives (e.,g, [100]. Furthermore, more sophisticated strategies exploiting both user types have been emerged. One common strategy is to integrate planning into DP learning, as in Dyna-Q (DDQ) framework [101]. Specifically, DDQ incorporates an environment model into the dialog system, known as the world model which can simulate the real user behaviors and generate simulated experiences. During DP learning, real user experiences are utilized for (1) training the world model and generate simulated experiences, and (2) improving the DP as well. Simulated experiences are then used to further improve the DP via indirect RL (planning). The earned benefit is allowing efficient DP learning by using only a small number of real user interactions. More effective approaches in this context are aimed either to obtain high-quality simulated experiences or find a balance between real user experiences and simulated ones, by incorporating heuristics, discriminators, and active learning into the planning step [102], [99]. Furthermore, with the aim of improving the DP learning speed, many RL approaches have initialized the DP to be reasonable before switching to online interaction with real or simulated users. This can be done by using either SL [5], [80] on available corpus or dialogues generated [103]as a result of running an agent-backend with a rule-based policy.

[7]Mathematical foundations related to RL are out of scope of this survey, and the interested reader can refer to [13] for more details

TABLE I: Summary of DM approaches with respect to Dialogue state, DST and DP dimensions

Method	Sub-Method(s)		Dimensions		
			Dialogue state	Dialogue state tracking	Dialogue policy
Handcrafted	R-Based	pattern/ response	Not represented	-	Purely handcrafted rules
		Pre/Post	User intent with slot-value pairs	Purely handcrafted rules	
	SM-Based			SM with handcrafted rules	
	Activity-Based			Activity-Based model with handcrafted rules	
	Frame-Based			Frame-Based model with handcrafted rules	
Data-Driven	SL Models		Mostly represented as Binary/multinomial distribution over multiple states, each consists of a set of slot-value pairs	Statistical models Deep Learning models	DST-Depend Policy: Deep Learning models End-to-end model: Seq2Seq models, Memory networks
	RL models	Flat RL	Mostly represented as Multinomial distribution over multiple possible states, each consists of a set of slot-value pairs Rarely represented as a compact representation of a set of features especially in DRL	Bayesian inference models Handcrafted rules Deep Learning models	Value-based methods Policy gradient methods
		HRL			
Hybrid models			Either represented as a true state consisting of slot-value pairs or a multinomial distribution over multiple possible states	Combining a rule based and a SL based model Combining multiple SL models	Combining a SL model with domain knowledge and rules Combining multiple SL/Rule-based models

Context support

Similar to SL approaches, conversation history is the main feature that has been widely considered by RL-based approaches. Many RL-based approaches [73], [104], [88], [93], [92], [79], [105], [82], [89], [90] to DST and DP support short-term history as they leverage information only from the last conversation turn. This renders them only tailored for single-turn conversation. Indeed, the success of such approaches is based on the fact that the user always provides utterances along with all required information the DST/DP needs. Otherwise, the DP is supposed to explicitly request any missing information from the user making the conversation less natural and not intelligent. Despite that, there are many others that support long-term history by taking either the previous dialog state [77], [72], [76], [78], [75], [27] or a set of particular features [83], [80], [81] (especially all the previously filled slots and past system actions) in the tracking of the current dialogue state. Although this contextual information is exploited mainly to improve the task of DST, the DP is naturally influenced since it acts on a context-aware dialogue state which results in deciding more personalized actions. In addition to the conversation history, leveraging domain-specific knowledge has been also considered by some RL approaches such as [91], where graph structure between the slots and their domains are integrated into the DP learning.

C. Hybrid approaches

The hybrid approach to DM rests on the idea of combining multiple approaches either handcrafted or data-driven in order to capitalize on the benefits of each.

Dialogue state tracking

Attempts that have been made for hybrid DST models can be summarized into two key ideas: (i) combining a rule-based model with a SL model (ii) combining multiple SL models. The first ideas (i) in turn is performed using two possible ways. The straightforward possibility relies on applying a rule-based model in parallel with the SL model and taking the outputs union of both as a final dialogue state. The hybrid DST model proposed by [106] is an example of such approach which rely on support vector machine (SVM) classifiers per slot-value pair as a SL model and a rule-based model. The latter includes a set of rules most of them are automatically generated using general rule templates and then selected according to their state tracking precision on the training dataset. The second possibility allows the rule-based model to be improved using training data by including a set of parameters that can be learned by any ML model. An example of this approach is the hybrid DST model of Vodolán et al. [109], [110], which defines a set of rules to update the probability distribution related to the dialogue state, where the probability for each slot is parameterized by two parameters. These latter have been learnt automatically using an LSTM network. All those approaches have been proven effective for DST with results outperformed the performances of rule-based and ML models if applied separately. But these results, while encouraging, have been achieved within datasets of less challenging conversation, as they consider only one domain whose slots and values are fixed in an ontology in advance. Therefore, it is still not clear

TABLE II: Summary of DM approaches with respect to the context features

Feature	Approaches
Conversation history	Rule-based:[15], [16], [20], [27] ; FSM-Based:[33], [32], [31]; Activity-based: ManyChat, Chatfuel and FlowXO; Frame-Based: [37], [36], Amazon Alexa, Google actions, DialogFlow; SL-based: [40], [41], [42], [44], [45], [39], [43], [46], [47], [55], [3], [49], [62]; RL-based: [88], [93], [92], [79], [105], [82], [89], [90], [77], [72], [76], [78], [75], [27], [83], [80], [81] ; Hybrid: [106], [107], [108], [54], [109], [110]
User profile	rule-based: [15], [18]; FSM-Based: [32]; Activity-based: ManyChat, Chatfuel and FlowXO; Frame-Based: [37], [36], Amazon Alexa, Google actions, DialogFlow;
External knowledge	Frame-Based: Google actions, Amazon Alexa, DialogFlow; SL-based: [53], [65], [64]; RL-based: [91]; Hybrid: [107];

whether these approaches will still robust and scalable against complex conversations that go beyond multiple domains and characterized by a dynamic change of slots. Also, the fact that DST is limited to the vocabulary used by the ontology to denote slots and their values make these approaches not able to deal with natural language variation. This shows clearly the inability of these approaches to track unseen slot-value pairs.

The second idea relies on combining multiple SL models. Most of leading SL-based hybrid approaches have attempted to combine the benefits of both using predefined ontology-based methods and open-vocabulary methods that either dynamically generate a candidate set of slot values or pointing them directly from input utterances. Examples include [54], [111], [112], [113], [114], [115]. The aim is to allow DST over unknown slot values that are not defined in a domain ontology. For instance, Hyst [54] proposed by Amazon learns for each slot the optimal DST method that has the highest accuracy between ontology-based DST model and open-vocabulary DST model. In doing this, it seeks to mitigate the scalability issue suffered from the ontology-based method and the low performance of the open Vocabulary method in certain cases.

The other hybrid approaches relied on ensemble-learning (EL), where the output of many DST models is synthesized to improve performance beyond any single model. The simplest EL method is score averaging, where the final DST output is obtained by averaging the output distributions of multiple trackers [116]. Its key advantage is that it does not require extra training data. A notable example is the RNN-based DST model proposed by Henderson et al. [46], where outputs of at least 6 individual RNNs trained with varying hyper-parameters are averaged to give the final distributions. The other considered EL method is known as Stacking (used in [116], [65]), where the outputs of the constituent DST models are fed to a new classifier that makes a final prediction, requiring a validation set for training. Both methods have been investigated in the DSTC2 [116] which demonstrated that the most accurate DST can be achieved by combining multiple trackers.

Dialogue Policy

There are two methods commonly adopted to learn the dialogue policy using a hybrid approach. The first method consists of integrating a SL model with handcrafted domain knowledge and rules. Hybrid Code Networks proposed by Williams et al. [107] is an example of such approach. Here, the DP is modeled using a RNN model augmented with handcrafted constraints and rules via software and action templates. The motivation behind this is that some simple operations like sorting a list of database results would be better implemented in a few lines than using thousands of dialogs to learn them. In addition, in some cases, programmed constraints are essential to identify the permitted actions among which the policy can choose. For example, in the dialing domain scenario, if a target phone number has not yet been identified, the API action to place a phone call may be masked [107].

The second method relies on applying multiple DP models simultaneously, whereby the next system action is decided by the policy model that predicts it with the highest confidence. RASACore [108] is an example of such a method that utilizes multiple dialogue policies, including rule-based, neural networks, and embedding models. Although all these policies are well packaged into software modules that can be reused easily by interested parts, their parallel execution may engender additional cost in terms of processing time and resources.

Context support

Similar to ML approaches, including SL and RL, hybrid approaches [106], [107], [108], [54] support the modeling of context, mostly, by leveraging the conversation history feature. An LSTM model over past user utterances and system replies is commonly used to encode the conversation history because of its ability to model long-term dependency or in another term can remember information across several turns. Moreover, other hybrid approaches [109], [110] utilize the current turn along with the previous dialogue state as input each time they update the new dialogue state so that it implicitly considers information from previous turns. This has been adopted to simplify the learning process of a dialog state that is aware of context without considering the entire conversation history.

IV. Summary and Future directions

In this section, we summarise DM approaches by discussing their key strengths and shortcomings (Table III) and then identify opportunities for future research directions.

A. Summary

Handcrafted approaches provide a straightforward way to design dialogue management that helps for the rapid prototyping of conversational systems. They are especially suitable for conversation scenarios with clearly-defined flow and goals and for domains concerned with strict adherence to business rules. For example, in domains like security, safety, and healthcare, conversational systems have to be able to adequately predict the expected and unexpected usage scenarios. An important feature of Handcrafted approaches, especially the activity-based/SM-based models, is the conversation flow traceability, which helps to track the interpretation of each user input and system actions for further fixes/improvements. Without being surprised, these approaches, however, suffer from poor domain portability and naturally comes at a high development cost as a lot of handcrafted features including DST/DP rules, state/action spaces, and interaction models are required.

In addition, the fact that DST model maintains a single hypothesis for the dialogue state makes the handcrafted approaches not robust when facing unforeseen user inputs or confronted with NLU recognition errors. In such situations, the DP often decides to reply by “I didn’t understand” to push users either to rephrase their inputs or start a new interaction. This is one of the reasons that can lead to the frustration of the user which may cause her to abandon the interaction.

Data-driven approaches, including SL and RL based models, reduce the development and maintenance cost of DM by automatically learning the dialogue state and policies [2], at the expense of the effort to get training data. In particular, the adoption of deep neural models along with word embedding techniques in DST notably facilitates the learning of dialogue state features (such as slots, slot values, history variables), which avoids defining a full ontology in advance, or even obviating it completely. Thus, the scalability of DST is improved in the sense that it is able to deal with changing dialogue

TABLE III: Summary of strengths and weaknesses of DM approaches

Method	Sub-Method(s)	Pros	Cons
Handcrafted	All methods	Are easy to implement and incorporate domain/business knowledge Do not require any training data Enable the conversation traceability	Have high development and maintenance cost Have limited scalability and robustness Do not consider user feedbacks to adapt the dialogue policy
Data-Driven	SL Models	Learning of state and action spaces and their features can be fully automated Able to deal with the natural language variations Require less effort to be adapted to new domains	Require acquiring sufficient and high-quality training data Do not consider user feedbacks to adapt the dialogue policy
	RL models – Flat RL	Robust and able to deal with uncertainty in user utterances Can adapt to different user behaviors Feature-engineering of state/action spaces can be alleviated with the adoption of DL models	Limited to small-scale conversation domains Require acquiring sufficient and high-quality training data
	RL models – HRL		Requires domain-specific knowledge to specify good task hierarchies Require acquiring sufficient and high-quality training data
Hybrid models	Rule/ML model	Reduce the learning complexity and the amount of training data Provide control over conversation flow	Feature-engineering of state/action spaces Suffer from poor domain portability
	Multiple Rule/ ML models	Improve performance	Require powerful resource settings, sufficient and high-quality training data, and development effort

domains involving the addition of new slots and values that are not in the training set. However, it has to be noted that many DST models do not consider intent in the state tracking, which makes the exploration of slot-value pairs be performed in a large search space. Tracking first the intent will give the DST model the opportunity to have a prior prediction on slot-value pairs it looks for, therefore making its task much easier without involving complex computations that typically result in slower learning. Furthermore, regarding SL-based models to DP, they implemented either as a DST-depend model that requires an explicit representation of the dialogue state or as an end-to-end model that reads directly from a user utterance and produces a system action. A DP as DST-depend model has the separation of concerns advantages (e.g., the errors that it may face with are clear and can be easily resolved), but it suffers from the strong dependence on the quality of the DST model. In addition, many SL-based works implemented such a model assume a fixed set of actions. Whereas, by adopting Seq2Seq models or end-to-end memories, the end-to-end DP approach dispenses the need for any handcrafting of state and action spaces. This approach, however, heavily relies on the quantity and quality of data and does not provide clear integration of API invocations. In both approaches, the DP has been trained using supervised learning, thus it does not have the capability to be adapted to accommodate user feedback during interaction.

Such an issue is widely resolved using RL since it offers the possibility to improve policy performance over time by acting on reward/punishment signals that can be either elicited or obtained explicitly from users. In this approach, the DM problem is cast either by MDP or POMDP while the DST can be done using multiple models including rule-based, BI, and DL models. DL models outperform the performance of Rule-based/BI models and offer a DST without resorting to features engineering. Based on the dialogue state, the policy selects the next action using either value-based methods or policy-based methods. Recently, these methods integrated with DL models giving rise to the Deep RL approach. Such integration notably alleviates the effort of feature engineering related to state/actions and facilities the policy learning on original state and action spaces instead of using summary spaces as in the traditional RL methods. More interestingly, the application of HRL empowers these methods to be applied to complicated conversation tasks that span across multiple topics and domains. The success of this approach, however, highly requires specifying a good task hierarchy that still requires domain-specific knowledge and careful engineering [117].

Hybrid approaches involving diverse combinations of handcrafted and data-driven models can be considered as an important step to improve the performance of DM and increase its capability to generalize. In particular, approaches combining ML model with a set of rules grant more flexibility to application developers to control conversation flow and ensure that it is adequately aligned with business rules. In addition, it has been proven that they can reach performances comparable to purely ML models with less training data. While such a hybrid approach may require an amount of developer effort, it can be seen as very useful in practical settings where collecting realistic dialogues for a new domain can be expensive. On the other hand, approaches combining multiple ML/rule models can provide a potentially stronger DM solution. However, applying multiple ML models at the same time may amplify the need for training data and powerful resource settings.

Finally, with regard to the *Context support*, we observed that SL and RL approaches have provided advanced capabilities than handcrafted approaches since contextual information and dependencies among them are learned automatically. Although moderns commercial systems like Amazon Alexa, Google actions, DialogFlow, and many others provide bot developers with sophisticated and easy-to-use IDEs, designing context-aware dialog managers remains a challenging task. It requires very good expertise especially if the conversation scope may span across multiple domains. Additionally, regarding the selected context features, we noted that the conversation history has been the widely covered feature in the selected DM state-of-art approaches. Leveraging the conversation history has

proven essential to hold complex and multi-turn conversations, as processing missing information in the dialogue state may require the DST model to go beyond multiple previous turns up to the last turn. Leveraging knowledge from user profiles, on the other hand, has been mostly considered by commercial frame-based and activity-based systems. Leveraging external knowledge, including domain-specific or general knowledge was the less covered feature.Although our survey focuses on context support in DST and DP tasks within task-oriented dialog systems, it should be noted that the use of external knowledge to effectively understand context has received more considerable attention in non-task-oriented conversation systems, including question answering and recommendation systems. Many of those systems are developed using end-to-end models [118], [119], [120], with the aim of generating more informative and engaging responses. Three approaches, including RNN, MemNN and Transformer based models, have been shown to be useful for supporting external knowledge [121].

B. *Future directions*

Despite the aforementioned achievements, designing and engineering scalable and robust dialogue management techniques that offer human-like conversational prowess remains a deeply challenging problem. Next, we highlight some possible research directions:

- **Automated generation and formal verification of DM models.** Handcrafted DM approaches are still the suitable choices for conversations where user Inputs/options can be known a prior and in general for any application domain where determinism property is required. However, current approaches suffer from high development and maintenance costs. Therefore, having automated mechanisms that better leverage existing resource models such as ontologies, knowledge graphs, business processes, etc., is a key step toward semi or even fully automated generation of handcrafted DM models. Furthermore, formal verification of such models may still be required to ensure their reliability.
- **Supporting more complex conversation tasks.** Solving complicated conversation tasks that span across multiple topics and domains is currently an active field of research. While HRL approaches have shown promising results in dealing with such an issue, they require expert knowledge to design the hierarchical structure of tasks. Thus, learning the natural hierarchy of tasks automatically represents a key research directive that can contribute to facilitating the effective adoption of the HRL. Furthermore, the same conversation task can involve executing different API calls due to the variability in user requirements, e.g., preferences for quality of service like price or others vary from one user to another. In this context, there is a need to strengthen dialogue management with flexible mechanisms that able to dynamically discover and select API/Services that adequately match with user requirements.
- **Handling conversation breakdowns.** Problems including misunderstandings, inappropriate responses, disagreements, complaints, rejection of offers represent persistent concerns that may cause breakdowns in conversations and therefore may negatively impact the user experience. Conversational systems are still not able to deal with these breakdowns due to the ambiguous nature of natural language and the large variation of user utterances. Therefore, it is essential to endow dialogue management techniques with effective strategies that are able to identify potential breakdowns in conversation when occur and repair them automatically.
- **Data control and quality.** The success of data-Driven and hybrid approaches highly dependent on the availability of high-quality training data, which still an open issue. Acquiring high-quality training data requires effective methodologies and processes for selecting, pipelining, tuning, and controlling data acquisition tasks. This represents a key research area that can affect the performance of dialogue management and therefore enhancing the quality of conversational systems.
- **Towards explainable conversations**. Explainability is an important aspect to make the conversation more human-like [122]. In this context, the dialog management component should be able to explain their decisions and behaviors. This is not only useful in allowing developers to inspect their models, but also in offering higher transparency to end-users, allowing them to inquire about the reasons behind the decisions of a conversational system.

Acknowledgement. This work was supported by the PICASSO (IDEX/FEL/2018/01 - 18IA102UDL) project at LIRIS lab.

REFERENCES

[1] H. Chen, X. Liu, D. Yin, and J. Tang, "A survey on dialogue systems: Recent advances and new frontiers," *SIGKDD Explor. Newsl.*, vol. 19, no. 2, p. 25–35, 2017.

[2] M. McTear, Z. Callejas, and D. Griol, *The Conversational Interface: Talking to Smart Devices*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[3] Z. Zhang, M. Huang, Z. Zhao, F. Ji, H. Chen, and X. Zhu, "Memory-augmented dialogue management for task-oriented dialogue systems," *ACM Transactions on Information Systems (TOIS)*, 2019.

[4] J. G. Harms, P. Kucherbaev, A. Bozzon, and G. J. Houben, "Approaches for dialog management in conversational agents," *IEEE Internet Computing*, vol. 23, no. 2, pp. 13–22, 2019.

[5] P. Su, P. Budzianowski, S. Ultes, M. Gasic, and S. J. Young, "Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management," in *18th SIGdial*, 2017.

[6] Z. Peng and X. Ma, "A survey on construction and enhancement methods in service chatbots design," *CCF Transactions on Pervasive Computing and Interaction*, 2019.

[7] S. A. Abdul-Kader and D. J. Woods, "Survey on chatbot design techniques in speech conversation systems," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 7, 2015.

[8] S. Hussain, O. Ameri Sianaki, and N. Ababneh, "A survey on conversational agents/chatbots classification and design techniques," in *Web, Artificial Intelligence and Network Applications*, 2019, pp. 946–956.

[9] K. Ramesh, S. Ravishankaran, A. Joshi, and K. Chandrasekaran, "A survey of design techniques for conversational agents," in *Information, Communication and Computing Technology*, 2017.

[10] I. V. Serban, R. Lowe, P. Henderson, L. Charlin, and J. Pineau, "A survey of available corpora for building data-driven dialogue systems: The journal version," *Dialogue and Discourse*, vol. 9, no. 1, pp. 1–49, 2018.

[11] J. Deriu, A. Rodrigo, A. Otegi, G. Echegoyen, S. Rosset, E. Agirre, and M. Cieliebak, "Survey on Evaluation Methods for Dialogue Systems," *CORR*, no. 1, 2019.

[12] A. Fadhil and G. Schiavo, "Designing for health chatbots," *CoRR*, vol. abs/1902.09022, 2019.

[13] J. Gao, M. Galley, and L. Li, "Neural approaches to conversational AI," *Foundations and Trends in Information Retrieval*, 2018.

[14] D. Jurafsky, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, third edition draft ed., 2019.

[15] C. Smith, N. Crook, J. Boye, D. Charlton, S. Dobnik, D. Pizzi, M. Cavazza, S. Pulman, R. S. de la Camara, and M. Turunen, "Interaction strategies for an affective conversational agent," in *Intelligent Virtual Agents*, 2010.

[16] G. R. Sankar, J. Greyling, D. Vogts, and M. C. du Plessis, "Models towards a hybrid conversational agent for contact centres," in *Proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding the Wave of Technology*, 2008, p. 200–209.

[17] C. Chakrabarti and G. F. Luger, "A semantic architecture for artificial conversations," in *The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems*, 2012, pp. 21–26.

[18] R. E. Banchs, R. Jiang, S. Kim, A. Niswar, and K. H. Yeo, "AIDA: Artificial intelligent dialogue agent," in *Proceedings of the SIGDIAL 2013 Conference*, 2013, pp. 145–147.

[19] J. Boye, "Dialogue management for automatic troubleshooting and other problem-solving applications," in *8th SIGDial workshop on discourse and dialogue*, 2007.

[20] D. Bohus and A. I. Rudnicky, "Ravenclaw: dialog management using hierarchical task decomposition and an expectation agenda," in *INTERSPEECH*, 2003.

[21] R. Wallace, *The elements of aiml style*. ALICE A. I. Foundation, 2003.

[22] J. Weizenbaum, "Eliza — a computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 26, no. 1, p. 23–28, 1983.

[23] K. M. Colby, *Human-Computer Conversation in A Cognitive Therapy Program*. Springer US, 1999, pp. 9–19.

[24] M. F. McTear, "Spoken dialogue technology: Enabling the conversational user interface," *ACM Comput. Surv.*, vol. 34, no. 1, p. 90–169, 2002.

[25] C. Thorne, "Chatbots for troubleshooting: A survey," *Language and Linguistics Compass*, vol. 11, 2017.

[26] J. D. Williams, A. Raux, and M. Henderson, "The dialog state tracking challenge series: A review," *D&D*, vol. 7, pp. 4–33, 2016.

[27] Z. Wang and O. Lemon, "A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information," in *Proceedings of the SIGDIAL Conference*. Metz, France: Association for Computational Linguistics, 2013, pp. 423–432.

[28] D. Burgan, "Dialogue systems & dialogue management," DST Group Edinburgh: Edinburgh, Australia, Tech. Rep., 2017.

[29] D. Jurafsky and J. H. Martin, *Speech and Language Processing: Chatbots & Dialogue Systems*, third draft ed., 2020. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[30] R. J. L. John, N. Potti, and J. M. Patel, "Ava: From data to insights through conversations," in *CIDR-8th Biennial Conference on Innovative Data Systems Research*, 2017.

[31] E. Fast, B. Chen, J. Mendelsohn, J. Bassen, and M. S. Bernstein, "Iris: A conversational agent for complex tasks," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, 2018, p. 1–12.

[32] N. Bradley, T. Fritz, and R. Holmes, "Context-aware conversational developer assistants," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2018, pp. 993–1003.

[33] R. J. Leo John, J. M. Patel, A. L. Alexander, V. Singh, and N. Adluru, "A natural language interface for dissemination of reproducible biomedical data science," in *MICCAI-Medical Image Computing and Computer Assisted Intervention*. Springer International Publishing, 2018, pp. 197–205.

[34] A. Koller, T. Baumann, and A. Köhn, "Dialogos: Simple and extensible dialog modeling," *Fachbereich Informatik*, 2018.

[35] J. Singh, M. H. Joesph, and K. B. A. Jabbar, "Rule-based chabot for student enquiries," *Journal of Physics: Conference Series*, vol. 1228, 2019.

[36] D. Bobrow, R. Kaplan, M. Kay, D. Norman, H. S. Thompson, and T. Winograd, "Gus, a frame-driven dialog system," *Artif. Intell.*, vol. 8, pp. 155–173, 1977.

[37] O. Lemon, A. Bracy, A. Gruenstein, and S. Peters, "The witas multi-modal dialogue system i," in *INTERSPEECH*, 2001.

[38] R. Catizone, A. Setzer, and Y. Wilks, "Multimodal dialogue management in the COMIC project," in *Proceedings of the 2003 EACL Workshop on Dialogue Systems: interaction, adaptation and styes of management*, 2003. [Online]. Available: https://www.aclweb.org/anthology/W03-2705

[39] H. Ren, W. Xu, Y. Zhang, and Y. Yan, "Dialog state tracking using conditional random fields," in *Proceedings of the SIGDIAL Conference*. Association for Computational Linguistics, 2013, pp. 457–461.

[40] S. Lee, "Structured discriminative model for dialog state tracking," in *Proceedings of the SIGDIAL Conference*. Association for Computational Linguistics, 2013, pp. 442–451.

[41] J. Williams, "Multi-domain learning and generalization in dialog state tracking," in *Proceedings of the SIGDIAL Conference*. Association for Computational Linguistics, 2013.

[42] ——, "A critical analysis of two statistical spoken dialog systems in public use," in *Proceedings IEEE Workshop on Spoken Language Technology (SLT)*. IEEE Spoken Language Technology Workshop, 2012.

[43] A. Metallinou, D. Bohus, and J. Williams, "Discriminative state tracking for spoken dialog systems," in *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, 2013.

[44] M. Henderson, B. Thomson, and S. Young, "Deep neural network approach for the dialog state tracking challenge," *SIGDIAL- 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Proceedings of the Conference*, pp. 467–471, 2013.

[45] N. Mrkšić, D. Ó Séaghdha, T.-H. Wen, B. Thomson, and S. Young, "Neural belief tracker: Data-driven dialogue state tracking," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Jul. 2017, pp. 1777–1788. [Online]. Available: https://www.aclweb.org/anthology/P17-1163

[46] M. Henderson, B. Thomson, and S. J. Young, "Word-based dialog state tracking with recurrent neural networks," in *15th SIGDIAL Conference*, 2014.

[47] N. Mrkšić, D. Ó Séaghdha, B. Thomson, M. Gašić, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young, "Multi-domain dialog state tracking using recurrent neural networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Jul. 2015, pp. 794–799.

[48] L. Žilka and F. Jurčíček, "Lectrack: Incremental dialog state tracking with long short-term memory networks," in *Text, Speech, and Dialogue*. Springer International Publishing, 2015, pp. 174–182.

[49] X. Yang and J. Liu, "Dialog state tracking using long short-term memory neural networks," in *INTERSPEECH*, 2015.

[50] A. Rastogi, R. Gupta, and D. Hakkani-Tur, "Multi-task learning for joint language understanding and dialogue state tracking," in *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, 2018, pp. 376–384.

[51] C.-S. Wu, A. Madotto, E. Hosseini-Asl, C. Xiong, R. Socher, and P. Fung, "Transferable multi-domain state generator for task-oriented dialogue systems," in *ACL*, 2019.

[52] L. Chen, B. Lv, C. Wang, S. Zhu, B. Tan, and K. Yu, "Schema-guided multi-domain dialogue state tracking with graph attention neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 7521–7528, 2020.

[53] S. Zhu, J. Li, L. Chen, and K. Yu, "Efficient context and schema fusion networks for multi-domain dialogue state tracking," in *Findings of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, pp. 766–781.

[54] R. Goel, S. Paul, and D. Hakkani-Tür, "Hyst: A hybrid approach for flexible and accurate dialogue state tracking," in *Interspeech*, 2019.

[55] A. Rastogi, D. Hakkani-Tur, and L. Heck, "Scalable multi-domain dialogue state tracking," in *Proceedings of IEEE ASRU*, 2017.

[56] A. Rastogi, X. Zang, S. K. Sunkara, R. Gupta, and P. Khaitan, "Schema-guided dialogue state tracking task at dstc8," in *AAAI Dialog System Technology Challenges Workshop*, 2020.

[57] S. McLeod, I. Kruijff-Korbayova, and B. Kiefer, "Multi-task learning of system dialogue act selection for supervised pretraining of goal-oriented dialogue policies," in *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*. Association for Computational Linguistics, 2019, pp. 411–417.

[58] P. Su, M. Gasic, N. Mrksic, L. M. Rojas-Barahona, S. Ultes, D. Vandyke, T. Wen, and S. J. Young, "Continuously learning neural dialogue management," *CoRR*, 2016.

[59] T. H. Wen, D. Vandyke, N. Mrkšíc, M. Gašíc, L. M. Rojas-Barahona, P. H. Su, S. Ultes, and S. Young, "A network-based end-to-end trainable task-oriented dialogue system," in *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, vol. 1, 2017, pp. 438–449.

[60] T. Zhao, A. Lu, K. Lee, and M. Eskenazi, "Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability," in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. Association for Computational Linguistics, 2017, pp. 27–36.

[61] C. D. Manning and M. Eric, "A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue," in *EACL*, 2017.

[62] J. Perez and F. Liu, "Dialog state tracking, a machine reading approach using memory network," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2017, pp. 305–314.

[63] S. Yang, R. Zhang, and S. Erfani, "Graphdialog: Integrating graph knowledge into end-to-end task-oriented dialogue systems," *arXiv preprint arXiv:2010.01447*, 2020.

[64] L. Zhou and K. Small, "Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering," *arXiv preprint arXiv:1911.06192*, 2019.

[65] Y. Murase, Y. Koichiro, and S. Nakamura, "Associative knowledge feature vector inferred on external knowledge base for dialog state tracking," *Computer Speech & Language*, vol. 54, pp. 1–16, 2019.

[66] I. Casanueva, P. Budzianowski, P.-H. Su, N. Mrkšić, T.-H. Wen, S. Ultes, L. Rojas-Barahona, S. Young, and M. Gašić, "A benchmarking environment for reinforcement learning based task oriented dialogue management," in *31st Conference on Neural Information Processing Systems*, 2017.

[67] M. Fazel-Zarandi, S.-W. Li, J. Cao, J. Casale, P. Henderson, D. Whitney, and A. Geramifard, "Learning robust dialog policies in noisy environments," *arXiv preprint arXiv:1712.04034*, 2017.

[68] P. Lison, "A hybrid approach to dialogue management based on probabilistic rules," *Computer Speech and Language*, vol. 34, no. 1, pp. 232 – 255, 2015.

[69] S. Young, M. Gašić, B. Thomson, and J. D. Williams, "Pomdp-based statistical spoken dialog systems: A review," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1160–1179, 2013.

[70] M. S. Henderson, "Discriminative methods for statistical spoken dialogue systems," Ph.D. dissertation, University of Cambridge, 2015.

[71] J. D. Williams and S. Young, "Partially observable markov decision processes for spoken dialog systems," *Computer Speech & Language*, vol. 21, no. 2, pp. 393 – 422, 2007.

[72] J. Williams, P. Poupart, and S. Young, "Factored partially observable markov decision processes for dialogue management," *Proceedings of the 4th Workshop on Knowledge and Reasoning in Practical Dialog Systems*, 01 2005.

[73] B. Zhang, Q. Cai, J. Mao, E. Chang, and B. Guo, "Spoken dialogue management as planning and acting under uncertainty," in *EUROSPEECH 2001 Scandinavia, 7th European Conference on Speech Communication and Technology*, 2001, pp. 2169–2172.

[74] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1, pp. 99 – 134, 1998.

[75] J. D. Williams, "Incremental partition recombination for efficient tracking of multiple dialog states," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010, pp. 5382–5385.

[76] M. Gašić and S. Young, "Effective handling of dialogue state in the hidden information state pomdp-based dialogue manager," *ACM Trans. Speech Lang. Process.*, vol. 7, no. 3, 2011.

[77] B. Thomson and S. Young, "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems," *Computer Speech and Language*, vol. 24, no. 4, p. 562, Mar. 2010.

[78] T. H. BUI, M. POEL, A. NIJHOLT, and J. ZWIERS, "A tractable hybrid ddn–pomdp approach to affective dialogue modeling for probabilistic frame-based dialogue systems," *Natural Language Engineering*, vol. 15, no. 2, p. 273–307, 2009.

[79] G. Xu, H. Lee, M. Koo, and J. Seo, "Optimizing policy via deep reinforcement learning for dialogue management," in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2018, pp. 582–589.

[80] T. Zhao and M. Eskenazi, "Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning," in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, 2016, pp. 1–10.

[81] Z. C. Lipton, J. Gao, L. Li, X. Li, F. Ahmed, and L. Deng, "Efficient exploration for dialog policy learning with deep BBQ networks \& replay buffer spiking," *CoRR*, 2016.

[82] H. Cuayahuitl, "Simpleds: A simple deep reinforcement learning dialogue system," in *Dialogues with Social Robots*, 2017.

[83] S. Singh, D. Litman, M. Kearns, and M. Walker, "Optimizing dialogue management with reinforcement learning: Experiments with the njfun system," *Journal of Artificial Intelligence Research*, vol. 16, p. 105–133, 2002.

[84] D. J. Litman, M. S. Kearns, S. Singh, and M. A. Walker, "Automatic optimization of dialogue management," in *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*, 2000.

[85] M. A. Walker, "An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email," *Journal of Artificial Intelligence Research*, vol. 12, p. 387–416, Jun 2000.

[86] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[87] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3–4, p. 229–256, 1992.

[88] P. Shah, D. Hakkani-Tur, and L. Heck, "Interactive reinforcement learning for task-oriented dialogue management," in *NIPS 2016 Deep Learning for Action and Interaction Workshop*, 2016.

[89] M. Fatemi, L. El Asri, H. Schulz, J. He, and K. Suleman, "Policy networks with two-stage training for dialogue systems," in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, 2016, pp. 101–110.

[90] H. Cuayáhuitl and S. Yu, "Deep reinforcement learning of dialogue policies with less weight updates," in *INTERSPEECH*, 2017.

[91] L. Chen, Z. Chen, B. Tan, S. Long, M. Gasic, and K. Yu, "Agentgraph: Toward universal dialogue management with structured deep reinforcement learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, pp. 1378–1391, 2019.

[92] B. Peng, X. Li, L. Li, J. Gao, A. Celikyilmaz, S. Lee, and K.-F. Wong, "Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017, pp. 2231–2240.

[93] P. Budzianowski, S. Ultes, P.-H. Su, N. Mrkšić, T.-H. Wen, I. Casanueva, L. M. Rojas-Barahona, and M. Gašić, "Sub-domain modelling for dialogue management with hierarchical reinforcement learning," in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, 2017, pp. 86–92.

[94] A. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 13, 2003.

[95] J. Schatzmann and S. Young, "The hidden agenda user simulation model," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 4, pp. 733–747, 2009.

[96] I. Gür, D. Hakkani-Tür, G. Tür, and P. Shah, "User modeling for task oriented dialogues," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 900–906.

[97] X. Li, Z. C. Lipton, B. Dhingra, L. Li, J. Gao, and Y.-N. Chen, "A user simulator for task-completion dialogues," *arXiv preprint arXiv:1612.05688*, 2016.

[98] E. Ie, C.-w. Hsu, M. Mladenov, V. Jain, S. Narvekar, J. Wang, R. Wu, and C. Boutilier, "Recsim: A configurable simulation platform for recommender systems," *arXiv preprint arXiv:1909.04847*, 2019.

[99] J. Gao, M. Galley, L. Li *et al.*, "Neural approaches to conversational ai," *Foundations and Trends® in Information Retrieval*, vol. 13, no. 2-3, pp. 127–298, 2019.

[100] P. Shah, D. Hakkani-Tür, B. Liu, and G. Tür, "Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and online reinforcement learning," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, Jun. 2018, pp. 41–51.

[101] B. Peng, X. Li, J. Gao, J. Liu, K.-F. Wong, and S.-Y. Su, "Deep dyna-q: Integrating planning for task-completion dialogue policy learning," *arXiv preprint arXiv:1801.06176*, 2018.

[102] Z. Zhang, X. Li, J. Gao, and E. Chen, "Budgeted policy learning for task-oriented dialogue systems," *arXiv preprint arXiv:1906.00499*, 2019.

[103] Z. Lipton, X. Li, J. Gao, L. Li, F. Ahmed, and L. Deng, "Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[104] Y. Yin, L. Shang, X. Jiang, X. Chen, and Q. Liu, "Dialog state tracking with reinforced data augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[105] G. Gordon-Hall, P. J. Gorinski, G. Lampouras, and I. Iacobacci, "Show us the way: Learning to manage dialog from demonstrations," in *The Eight Dialog System Technology Challenge (DSTC-8) Workshop at AAAI*, 2020.

[106] K. Sun, S. Zhu, L. Chen, S. Yao, X. Wu, and K. Yu, "Hybrid dialogue state tracking for real world human-to-human dialogues," in *Interspeech*, 2016, pp. 2060–2064.

[107] J. D. Williams, K. Asadi, and G. Zweig, "Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

[108] Rasa Technologies, "The rasa core dialogue engine," https://rasa.com/docs/rasa/core/about/- Last accessed on 2020-03-15.

[109] M. Vodolán, R. Kadlec, and J. Kleindienst, "Hybrid dialog state tracker with ASR features," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2017, pp. 205–210.

[110] M. Vodolán, R. Kadlec, and J. Kleindienst, "Hybrid dialog state tracker," in *Proceedings of the Machine Learning for SLU & Interaction NIPS Workshop*, 2015.

[111] J.-G. Zhang, K. Hashimoto, C.-S. Wu, Y. Wan, P. S. Yu, R. Socher, and C. Xiong, "Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking," 2020.

[112] S. Lei, S. Liu, M. Sen, H. Jiang, and X. Wang, "Zero-shot state tracking and user adoption tracking on schema-guided dialogue," in *Dialog System Technology Challenge Workshop at AAAI*, 2020.

[113] S. Gao, A. Sethi, S. Agarwal, T. Chung, and D. Hakkani-Tur, "Dialog state tracking: A neural reading comprehension approach," 2019.

[114] P. Xu and Q. Hu, "An end-to-end approach for handling unknown slot values in dialogue state tracking," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 1448–1457.

[115] Y. Wang, Y. Shen, and H. Jin, "A bi-model approach for handling unknown slot values in dialogue state tracking," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8019–8023.

[116] M. Henderson, B. Thomson, and J. D. Williams, "The second dialog state tracking challenge," in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, Jun. 2014, pp. 263–272.

[117] Y. Flet-Berliac, "The promise of hierarchical reinforcement learning," *The Gradient*, 2019.

[118] M. Ghazvininejad, C. Brockett, M.-W. Chang, B. Dolan, J. Gao, W.-t. Yih, and M. Galley, "A knowledge-grounded neural conversation model," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[119] P. Christmann, R. Saha Roy, A. Abujabal, J. Singh, and G. Weikum, "Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 729–738.

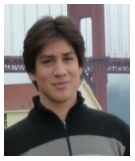
[120] X. Zhao, L. Wang, R. He, T. Yang, J. Chang, and R. Wang, "Multiple knowledge syncretic transformer for natural dialogue generation," in *Proceedings of The Web Conference 2020*, 2020, pp. 752–762.

[121] W. Zheng and K. Zhou, "Enhancing conversational dialogue models with grounded knowledge," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 709–718.

[122] S. Amershi and al., "Guidelines for human-ai interaction," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, p. 1–13.

Hayet Brabra received the Ph.D. degree from the Polytechnic Institute of Paris, Palaiseau, France, in 2020.

She is a Research Fellow with Claude Bernard University Lyon 1, France. Her research interests include Semantic Web, Cloud Computing and Cognitive Services.



Marcos Baez received the Ph.D. degree in computer science from University of Trento, Italy, in 2012.

He is a Research Fellow with Claude Bernard University Lyon 1, France. His research interests include Web Engineering, Crowdsourcing and Human-AI Interaction.

Boualem Benatallah received the Ph.D. degree in computer science from the University of Grenoble, Grenoble, France, in 1996.

He is a Scientia Professor at UNSW Sydney. His research interests include Web Services, Crowdsourcing, and Cognitive Services.

Walid Gaaloul received the M.S. and Ph.D. degrees in computer science from the University of Lorraine, Nancy, France, in 2002 and 2006, respectively, and the Habilitation degree from Pierre and Marie Curie University, Paris, France, in 2014.

He is a professor at Télécom SudParis. His research interests are on Business Process Management, Process Mining, Cloud Computing, Service Oriented Computing.

Sara Bouguelia is currently pursuing the Ph.D. degree with Claude Bernard University Lyon 1, Villeurbanne, France.

Her research interests include Cognitive Services.

Shayan Zamanirad is a research fellow at UNSW Sydney. He received his Ph.D. from UNSW. His research interests include Cognitive services and cloud computing. Contact him at shayan.zamanirad@unsw.edu.au.