A Generic Dialog Agent for Information Retrieval Based on Automated Planning Within a Reinforcement Learning Platform

Vishal Pallagani, Biplav Srivastava
Artificial Intelligence Institute, University of South Carolina
{vishalp@mailbox., biplav.s@}sc.edu

Abstract

With easy availability of large data sets online, like product catalogs and open data, a common business problem is to allow users to search them for information using natural interfaces. Dialog systems provide such an interface where a user can type or speak to the system and ask for information, and the system navigates the ambiguity of request, the complexity of content (size, hierarchy, schema) and usage considerations (response time, dialog length) to create a series of conversation leading to the system providing user the appropriate information. However, current learning-based methods to build a dialog agent require large training data, are data specific, and hard to scale while a user's interaction spans querying of multiple data sources. In this paper, we present a novel and generic approach for dialog for information retrieval based on automated planning within a reinforcement learning (RL)-based platform, ParlAI. The approach allows us to seamlessly scale to new data sources and to explore various planning and RL integration strategies. For instance, the planner performs search for response strategies that are controlled, goal-oriented, and across multiple turns without prior training data, while the RL is used to automate selection of data sources during an interaction. One can also just use the RL for end-to-end training or select a data source and use only the planner. We demonstrate the viability of our approach using the large data sets of UNSPSC and ICD-10, and a simple phone directory.

Introduction

Data is omnipresent online, and in many cases, available as hierarchical or relational data source. One of the mundane tasks users perform with data sources is to look them up for desired information (query). The common method to do so is by using specialized languages like Structured Query Language (SQL), which is known to a limited few with database training. Another approach to obtain information is by the tedious process of sifting and sieving through the huge corpus of data. Both of these aforementioned methods are neither effective nor accessible to diverse users and data sources. We seek to make data accessible to users using the natural interface of dialogs. Information retrieval using dialog systems is analyzed in-depth in (Radlinski and Craswell 2017) where the authors note that such an interface is especially useful when the user needs help in framing the query and the right result may be a set with multiple relevant answers.

There is a long history of dialog systems going back to 1960s when they first appeared to answer questions or do casual conversation(McTear, Callejas, and Griol 2016). There are many approaches to build them including finite-space, frame-based, inference-based and statistical learning-based (Crook 2018; Clark, Fox, and Lappin 2010; Inouye 2004; Young et al. 2013), of which, finite-space and frame-based are most popular with mainstream developers. The recent trend in research is to train the dialog system from end-to-end, allowing error signal from the end output (system) utterance to be back-propagated to raw (user) input, so that the whole dialog can be jointly optimized (Bordes, Boureau, and Weston 2017).

Given the plethora of implementation methods, recent surveys for building dialog systems[1] are (Ali and Gonzalez 2016) where the authors summarize the different approaches for building conversation systems and identify challenges, and (Fung et al. 2020) which focuses on deep-learning based methods for building chatbots. However, major caveats with these systems are that (a) they need large corpus of training data, and (b) once trained and deployed, they do not offer an ability to control the flow of conversation which is desirable in high-stakes domains like health and law. This has lead to renewed interest in inference-based methods to control system behavior (Cohen 2019; Botea et al. 2019a; Muise et al. 2019a).

In this paper, we present a generic approach for dialogs for information retrieval based on automated planning within a reinforcement learning (RL)-based platform, ParlAI (Miller et al. 2018). ParlAI has a unified architecture built with RL at its core for sharing, training and evaluating dialogue models. ParlAI has been actively used to build various state-of-the-art dialog systems. We integrate a cloud based planner with ParlAI in a modular approach - providing user with multiple workflow strategies. For instance, the planner performs search for response strategies that are controlled, goal-oriented, and across multiple turns without prior training data, while the RL is used to automate selection of data sources during an interaction. One can also just use the RL

[1] We also refer to them as *chatbots* interchangeably.

for end-to-end training or select a data source and use only the planner.

The system is able to navigate the ambiguity of request, the complexity of content (size, hierarchy, schema) and usage considerations (response time, dialog length) to create a series of conversation leading to the system providing user the appropriate information. One of the highlights of the approach is that the system is general with respect to data sources. The user can start the conversation with no data source and select them one by one, and the system can answer queries across them seamlessly. We demonstrate the viability of our approach using the large data sets of UNSPSC and ICD-10, and a simple phone directory.

The major contributions of this paper are:

- Incorporating planning-based dialog response generation in a RL-based dialog system
- Demonstrating a general, planning-based approach for creating chatbots for information lookup with three distinct data sources.
- Exploring multiple RL and planning integration strategies
- Creating a test-bed to evaluate conversations between user and the chatbot.

In the rest of the paper, we start with preliminaries about dialog systems and ParlAI and then describe data sources. We then present our approach followed by an illustration of its operation. We present preliminary results to show its promise and conclude.

Preliminaries

In this section, we give background on how dialog systems are built and the ParlAI system.

Data Retrieving Dialog Systems

We first clarify the terminology. Two or more participants engaged in conversation lead to a *dialog*. A *dialog* is made up of *turns*, where each turn is a series of *utterances* by one or more participants playing one or more *roles* in the conversations. As examples, an on-line forum can have a single role of *users* while a customer support dialog may have the roles of *customer* and *support agent*. We restrict to the basic setting where there is one user and one automated system who alternate utterance at each turn.

The core problem in building chatbots is that of dialog management (DM), i.e., creating dialog responses to user's utterances. The system architecture of a typical data-consuming dialog manager (DM) is shown in Figure 1. Given the user's utterance, it is analyzed to detect their intent and a policy for response is selected. This policy may call for querying a database, and the result is returned which is used by response generator to create a response using templates. The system can dynamically create one or more queries which involves selecting tables and attributes, filtering values and testing for conditions, and assuming defaults for missing values. It may also decide not to answer a request if it is unsure of a query's result correctness.

ParlAI Dialog System

ParlAI is an open source RL-based framework for building dialog systems at scale. It is aimed to allow research community to share existing as well as new tasks for dialog as agents that learn on them. ParlAI is also integrated with Mechanical Turk, helping researchers collect and evaluate conversations taking place between humans and agents. A detailed description of the complete functional components of ParlAI can be found in (Miller et al. 2018). The main *classes* in the framework that we extensively reuse are:

- **world** - is analogous to the environment which houses single or multiple agents facilitating conversation to take place.
- **agent** - is anything that can perform actions in the world. An agents can be a learned-model, a planner or even a human.
- **teacher** - is a type of agent that teaches other agents, used, especially in an RL setting.

The ParlAI core consists of policy (for RL), worlds, agents, teachers, evaluation metrics and other resources to assist the in the reuse of these components in other applications built using the framework.

Datasets Used

Data Source - UNSPSC

One popular category of user dialogs in business setting is inquiry of information that is organized in a taxonomy. For instance, a person wants to ask about an offering on a company's site (e.g., *pliers* - product, *plumbers* - service) but does not know which particular type. An example of taxonomy is United Nations Standard Products and Services Code (UNSPSC[2]), which *"is an open, global, multi-sector standard for efficient, accurate classification of products and services"*. Table 1 gives a summary of the levels in the taxonomy and its data. There are 4,302 items arranged into class, family, segment and commodity (lowest level). The *characteristic* column conveys the nature of abstraction across levels by reporting number of distinct (dissimilar) values, unique (count=1) values and their percentage, followed by highest frequency at a given level.

So, if the user inquires about a product, the chatbot can refer to the taxonomy to confirm what the user wants. As example, a query on *pliers* may refer to a surgical tool (Code: 42292303) or a vehicle servicing equipment (Code: 25191716) or a hand tool (Code: 27112106), and many variations therein. Thus, there can be a variety of strategies for a chatbot to help the user. It can go from highest level to lowest, or reverse, or ask randomly from the tree. Furthermore, if there is a restriction on how many turns the chatbot can take on this query, we want to explore if there are querying strategies that are optimal.

We will refer to it as D1 or UNSPSC.

[2]https://www.unspsc.org/

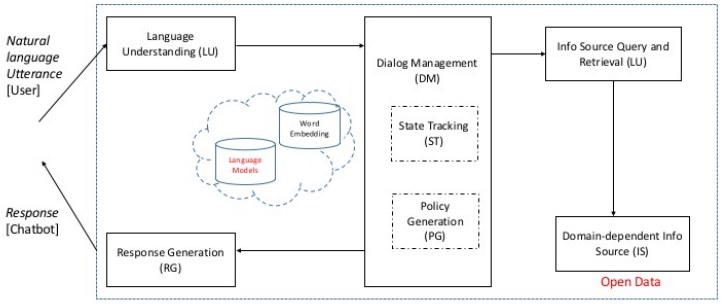

Figure 1: The architecture of a data-driven dialog system.

Level	Name	Characteristic
4 - Highest	Segment	36; 3 (0%); 825
3	Family	173; 23 (1%); 443
2	Class	826; 216 (5%); 49
1 - Lowest	Commodity	4302; 4302 (100%); 1

Table 1: Levels in UNSPSC Taxonomy.

Data Source - ICD

Healthcare domain has been laying emphasis on well organised and curated data to enable efficient communication between patients and doctors. An example of such taxonomy is the International Statistical Classification of Diseases and Related Health Problems (ICD[3]), from the World Health Organization. ICD provides a medical classification to identify diseases, signs and symptoms, abnormal findings, complaints, social circumstances, and external causes of injury or diseases (Hirsch et al. 2016). Medical practitioners extensively refer to ICD to obtain the codes for queries such as *injury of external jugular vein*. These codes form a basis of effective communication in the medical community.

Though there are web-based interfaces to query and find ICD codes, a dialog system would extend the access to users (here, medical practitioners, researcher, common people). In our work, we use ICD-10-CM which is the clinical modification of the 10th ICD revision by the United States. ICD-10-CM contains 94,766 codes, divided into 22 chapters as shown in Table 2. There are two major types of codes in ICD-10-CM, i.e., billable and non-billable. Every non billable-code can be broken down into atomic billable codes. Billable codes are the most specific, having no further child nodes under them, and can be used to indicate a diagnosis with utmost specificity. There are 72,621 billable codes in ICD-10-CM. On the contrary, non-billable codes can be further broken down into billable codes which offer more specificity to the diagnosis.

The user might start inquiring about a non-billable code (e.g., *injury of ulnar artery at wrist and hand level*). The agent would return the corresponding ICD-10-CM code (*S65.0*), but would also inform the user that there exists more specific diagnosis for his query. The user is given the liberty to make the decision and if opted for a billable code, the agent would perform multi-turn conversation and help the user in reaching one (e.g., *Laceration of ulnar artery at wrist and hand level of left arm, subsequent encounter* with code *S65.012D*).

Chapter	Block	Title
I	A00–B99	Certain infectious and parasitic diseases
II	C00–D48	Neoplasms
III	D50–D89	Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism
IV	E00–E90	Endocrine, nutritional and metabolic diseases
V	F00–F99	Mental and behavioural disorders
VI	G00–G99	Diseases of the nervous system
VII	H00–H59	Diseases of the eye and adnexa
VIII	H60–H95	Diseases of the ear and mastoid process
IX	I00–I99	Diseases of the circulatory system
X	J00–J99	Diseases of the respiratory system
XI	K00–K93	Diseases of the digestive system
XII	L00–L99	Diseases of the skin and subcutaneous tissue
XIII	M00–M99	Diseases of the musculoskeletal system and connective tissue
XIV	N00–N99	Diseases of the genitourinary system
XV	O00–O99	Pregnancy, childbirth and the puerperium
XVI	P00–P96	Certain conditions originating in the perinatal period
XVII	Q00–Q99	Congenital malformations, deformations and chromosomal abnormalities
XVIII	R00–R99	Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified
XIX	S00–T98	Injury, poisoning and certain other consequences of external causes
XX	V01–Y98	External causes of morbidity and mortality
XXI	Z00–Z99	Factors influencing health status and contact with health services
XXII	U00–U99	Codes for special purposes

Table 2: ICD-10 Chapters.

[3]https://www.cdc.gov/nchs/icd/index.htm

We will refer to this data source as D2 or ICD-10.

Data Source - Tiny Employee Directory

We also have created a relatively small data source - an employee directory. The data set consists of a employee name and their phone number pairs. The reasons behind using such a data set are that it is readily available at most organizations, easy to understand and test, and ideal for scalability and generalisation experiments that we plan for future.

We will refer to this data source as D3 or employee directory.

System Implementation and Evaluation

The overview of the proposed system is shown in Figure 2. It consists of three components: ParlAI Core, Planner and an Executor. The ParlAI core, provides the interface for the dialog agent to interact with the user. The Planner, along with ParlAI core help in *Intent Identification*. The learnt policy (present in ParlAI core) helps in the data *Source Selection* that matches the user's query. The Planner then generates a plan for the Executor, to perform *Information Retrieval* from the selected data source. The following subsections elaborate these features.

Planner

The proposed system is built using a cloud-based planner - Solver.Planning.Domains[4]. The planner helps in generating plans based on the domain and problem files written in Planning Domain Definition Language (PDDL). PDDL is a standard encoding language used for classical planning tasks. The domain and problem files help in capturing the planning tasks.

Domain The domain files consists of the actions and predicates needed for the planning task. The actions defined for the information lookup task help in getting the user query to facilitate conversation with the dialog agent. Once the user query is obtained, the check for existence of the data source and query type is performed. The query type can be either exact or partial. Exact meaning that the user query has a singular matching instance in the data. Partial, on the other hand, has multiple matching instances and need to be further disambiguated in order to satisfy the user's intent. Figure 3 captures the *CHECK_QUERY_TYPE* action defined in the information lookup task.

Problem The initial state of the planning task, along with the objects and the goal are specified in the problem file. Here, the goal refers to the recognised intent of the user. The problem and domain files together help in the generation of a plan to be followed by the dialog agent. Figure 4 shows a plan generated for the dialog agent when the system is yet to receive an input, and the received input is classified as *partial* by the *CHECK_QUERY_TYPE* action.

[4]http://solver.planning.domains

Intent Identification

The most crucial step in a dialog setting between the user and an automated system is identifying the intent. Based on the user's conversation with the system, the intent or what the user wants to achieve is established. Once the intent is identified, the dialog agent generates responses that steer towards realising the user's intent.

The dialog agent in our system is guided by the planner. The planner generates the questions to be posed to the user in order to glean information that would help in intent identification. An example user intent in our system is - *I want to know the UNSPSC code for pliers.* However, establishing intent from a single user dialog is an ideal situation, and usually requires multi-turn dialog exchanges. The planner, however, helps in posing the right questions that would help in identifying the intent in the least number of steps as possible. Instead of the planner for response generation, the proposed system also offers the option to use RL for response generation, which is the default operation of ParlAI. Once the intent is identified, the control is transferred to the source selector which is responsible for selecting the appropriate data source (here, *UNSPSC*).

Source Selection

Typically, a user manually selects the data he wishes to query. In addition to manual selection, our approach also offers the flexibility for automatically selecting the data source. ParlAI learns the policy to identify the data source based on the user's intent. For instance, if the recognised user intent is *code for pliers*, with no explicit mention to the data source, the system's learnt policy identifies *UNSPSC* to be the source to look for information.

However, when there is no data source available, the dialog agents performs general chit-chat but reminds the user for the access to data. The proposed approach has been tested on three data sources - D1, D2 and D3. The system is scalable, and can be used to query new data sets for performing the task of information lookup.

Information Retrieval

After the intent is identified and the data source is selected, the user can proceed to retrieve the information through exchange of dialogs. The user's queries are generally abstract in nature and would need further disambiguation in most of the cases. For instance, let us consider the recognised intent to be *code for pliers* and the data source selected is *UNSPSC*. The planner would now try to find the code for pliers, but, UNSPSC has 28 different kinds of pliers (*Brake Spring pliers, Surgical pliers, etc.*), categorised into 5 classes (*Vehicle Servicing, Orthodontic and Prosthodontic equipment, etc.*), belonging to 4 different families (*Hand Tools, Surgical Products, etc.*).

Since there is no exact match that satisfies the user's intent, the planner now generates a plan for disambiguation of the user's query. The disambiguation approach followed is to proceed from the highest hierarchy. Thus, the user is prompted by the dialog agent to select the family of the pliers he is looking for. If there is only plier belonging to

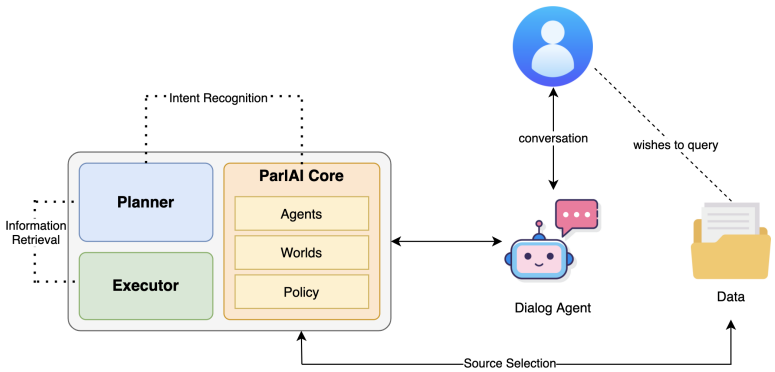

Figure 2: Proposed System Architecture

```
(:action CHECK_QUERY_TYPE
    :parameters (?x)
    :precondition (and (status-have-user-query ?x))
    :effect (and (when (is_query_exact ?x) (status-have-exact-query ?x))
                 (when (is_query_partial ?x) (status-have-partial-query ?x))
    )
)
```

Figure 3: An action in the PDDL Domain

```
GET_USER_QUERY user_dialogue
  GET_DB_ACCESS user_dialogue
    CHECK_QUERY_TYPE user_dialogue
      PERFORM_PARTIAL_MATCH_DISAMBIGUATION user_dialogue
        DISAMBIGUATE_HIGHEST_HIERARCHY_NODE user_dialogue
          DISAMBIGUATE_DEPENDENT_HIERARCHY_NODE user_dialogue
```

Figure 4: Plan generated by PDDL

that family, the planner guides the executor to retrieve the code from the data source. But, if further disambiguation is needed, the planner follows a similar approach making sure the user reaches his goal (here, to find the code for pliers) in optimal turns.

Experimental Results

This section presents the working of the proposed system, the evaluation metrics followed while testing and the obtained results.

Illustration

The working of the proposed dialog agent is illustrated in Figures 5 and 6. Figure 5 captures the task of information lookup performed on D1 data source. In this scenario, the user explicitly points to the data source for the system to perform lookup. Thus, the planner is the only active component here.

In Figure 6, RL is used to automatically identify the appropriate data source based on the user's query. We demonstrate this on data source D2. Based on the user's query, ICD-10-CM is automatically selected by the dialog agent to perform lookup.

The number of steps taken for the completion of information lookup is based on various factors such as the user's query being valid, i.e., have an instance present in the data source, the choices the user makes and the hierarchy present in the data source. In an ideal scenario, the maximum number of steps that need to be taken by the user to retrieve information from D1 is 3, and for D2 is 6.

Integration Strategies

In this paper, we explore three different Planning and RL integration strategies for response generation, source selection and information retrieval. We plan to explore more possibilities in future.

Planning without RL This strategy makes use of a planner alone to generate responses for the dialog agents as well as perform the task of information retrieval. PDDL is used to model the planning task and the generated plan is used to guide the dialog agent's responses in reaching the goal, i.e., the user's intent, data hierarchy, conversation control and data connectivity issues. However, the drawback of this approach is that the user must manually select the data source he wishes to query.

RL without Planning The second strategy is to make use of RL alone, with no integrated planning component. This is the regular ParlAI framework which learns the policy assuming training data is available about conversations taking place between a user and the dialog agent. The RL agent is not able to scale with data sources, control conversation unless examples are present in training data and handle dynamic connectivity situations.

Planning with RL Planning with RL is the third strategy, that is also used in our proposed system. Here, both the advantages of Planning and RL are exploited by overcoming their drawbacks. RL helps in automatically selecting the data source based on the user's query including a dummy, *No-data-source*, when no data source is registered in the system. The training phase will be small as the training data will only need the schema of the data sources. The Planner helps in guiding the dialog agent towards generating goal-oriented, multi-turn and controlled responses aimed at task completion.

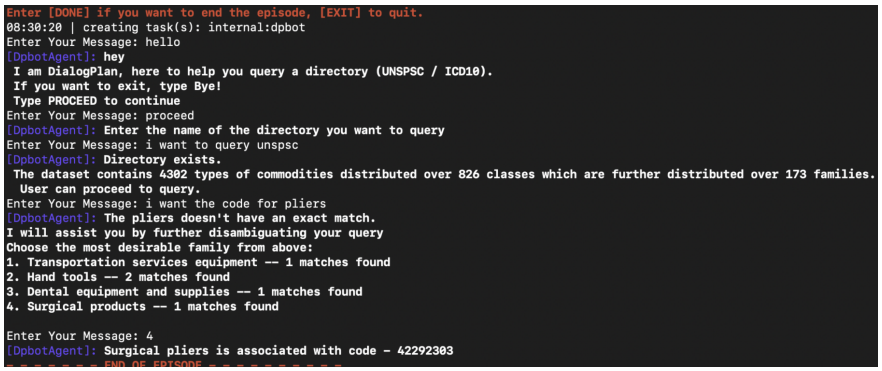

Figure 5: Information Lookup on D1. The user is searching for *pliers*.

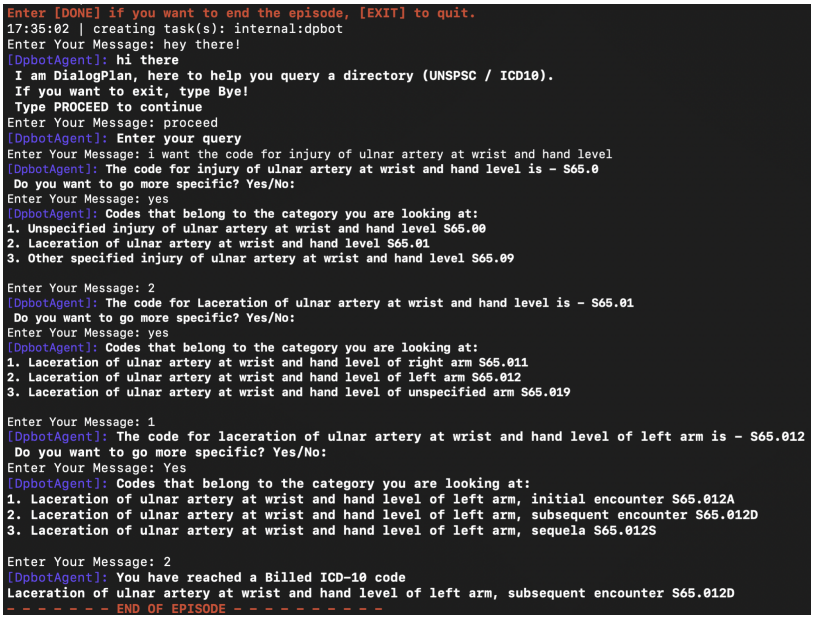

Figure 6: Information Lookup on D2. The user is searching for *laceration of ulnar artery at wrist and hand level*.

Evaluation Metrics and Examples

The proposed system is evaluated using three metrics to test the effectiveness of our approach. A comprehensive evaluation is left for future work. They metrics are:

- **Accuracy**: It refers to the percentage of correct utterances by the dialog agent that has helped the user in achieving his intent.
- **Intent Recognition**: Intent here refers to what the user wants to achieve at the end of the conversation with the dialog agent. Intent recognition is a metric for evaluating how well the dialog agent is able to understand user's needs.
- **Task Length**: The number of turns taken by the user and the dialog agent in completing the intent.

A detailed illustration of the system across data sources is shown in Table 3. The dialog agent saves the conversations with the user, along with the evaluation metrics such as accuracy, completion time and number of steps taken for intent completion in the form of tabular log files for future research.

Results

We evaluate the proposed generic dialog agent on the afore-mentioned data sources. The results of the conversation between the user and the agent are captured in Table 4. The accuracy obtained by the dialog agent in assisting fulfil his intent is 100%, given the query is in-scope. A query is said to be in-scope if it has an instance present in the data source. An out-of-scope query is one that does not have corresponding information to look for in the data source.

The task length varies on the user's intent for a given data source. The intent, however, can be changed over time. Consider the user's query for ICD-10-CM, *I want the code for injury of deep palmar arch*. The user has started off with just getting the code for a simpler diagnosis. If the user is satisfied with the agent's response - *The code for injury of*

		Data Source	Queries Tested	Completed Queries	Accuracy
User's Query	In-Scope	D1	20	20	100%
		D2	20	20	100%
		D3	5	5	100%
	Out-of-Scope	D1	10	10	-
		D2	10	10	-
		D3	3	3	-

Table 3: Evaluation of the System

User Query	**Final Bot Response**	**Task Length**	**Task Completion**
What is the code for lighters?	Lighters is associated with code - 12131707	1	Yes (in-scope)
I want the code for pliers	Surgical pliers is associated with code - 42292303	2	Yes (in-scope)
I want the code for pliers	Jewellers pliers is associated with code - 27112150	3	Yes (in-scope)
Code for taps?	Surgical taps is associated with code - 42292401	2	Yes (in-scope)
What is the code for nut cracker?	Nut cracker isn't present in the data-source.	1	No (out-of-scope)
Get me the code for hammers?	Claw Hammers is associated with code - 27111618	2	Yes (in-scope)
I want the code for injury of deep palmar arch	The code for injury of deep palmar arch is - S65.3	1	Yes (in-scope)
I want the code for injury of deep palmar arch	You have reached a Billed ICD-10 code. Laceration of deep palmar arch of unspecified hand, initial encounter S65.319A.	6	Yes (in-scope)
What is the code for heart attack?	Heart attack isn't present in the data-source.	1	No (out-of-scope)
What is the code for acute myocardial infarction?	The code for acute myocardial infarction is - I21	1	Yes (in-scope)
What is the phone number for Adam?	Adam's phone number is +1-902-453-xxxx	1	Yes (in-scope)
What is the phone number for John?	John is not present in the directory. Do you want to add his number?	1	No (out-of-scope)

Table 4: Detailed illustration of dialog agent's conversation with the user

deep palmar arch is - S65.3 and doesn't want to further proceed to reach a billable ICD code, then the task is said to be completed with length 1. However, if the user opts to view detailed diagnosis codes related to his query, the maximum length of task before the user reaches a billed code is 6.

We also would like to point out that the dialog agent currently is only capable of performing information lookup, and is not yet capable of identifying inherent relations. For instance, *myocardial infarction* is commonly known as *heart attack*. However, our system would not be able to assign the code for myocardial infarction with heart attack as that relation is not captured in the ICD-10-CM data source. However, we are aware that we can make the dialog agent be more intelligent by adding domain knowledge, which is a scope for future work.

The dialog agent is also tested on D3, the tiny curated employee dataset. The capabilities tested in this dataset are simple, and used for easier understanding of the system. The dialog agent helps in finding the corresponding phone number associated with an employee. If the query is out-of-scope, the dialog agent marks the task to be incomplete, but asks the user if he wishes to add a new entry to the data source, to make sure the same query does not lead to a failure again.

Related Work

Conversational Agents have been an active research area, especially in the recent times. (Cohen 2018) critiques the current approaches to build chatbots, bringing out contrastive differences between slot-filling and plan-based approaches. The significance of having a plan based dialogue agent that can perform both planning and plan recognition which would result in a chatbot that can plan, reason and converse is emphasised. The proposed work follows suite of using reasoning in a learning-based chatbot setting. (Botea et al. 2019b) tackles the problem of current chatbots being unable to perform multi-turn and complex conversations. They overcome this shortcoming by constructing dialogue plans automatically, which would be later plugged into a dialogue system to achieve goal-oriented conversations with the user. However, they do not consider information retrieval tasks where we try to optimize the number of conversational turns taken between the agent and the user in order to attain the desired answer. In (Nothdurft et al. 2015), the authors present a work on integrating a dialog system with a HTN planner for decision support. The problem is recommending personalized action chains (plans) to a person like exercises.

There is a well established literature on natural language querying (NLQ) (Li and Jagadish 2016) where a query by a user in a natural language is converted to a data source's query language like SQL. However, our approach is an alternative which exploits the advantages of a dialog system to disambiguate a user's query and help select from multiple similar results (Radlinski and Craswell 2017). (Muise et al. 2019b) proposes a paradigm shift from slot filling and dialogue trees to using automated planning for creating task-oriented chatbots. Planning helps in handling non-determinism, leading to creation of large and complex agents from compact declarative specifications. On the contrary, our approach adapts the use of a classical planner for information retrieval task and reason with information hierarchy and query ambiguity. Another line of work (Pasupat and Liang 2015) performs semantic parsing on semi-

structured tables in a question answering setting. It is a well presented work which encompasses both breadth of a knowledge source and depth of compositionality in semantic parsing. However, this approach is training data intensive, unlike ours. Also, the issue of how an ambiguous query is tackled is not elaborated.

Conclusion and Future Work

In this work, we have proposed a novel, generic, dialog agent based on automated planning within an RL-based framework for information retrieval. The approach allows us to seamlessly scale to new data sources and explore various planning and RL integration strategies. We have implemented the approach in ParlAI framework and demonstrated its working, along with preliminary evaluation on two large data sets - UNSPSC and ICD-10, and a simple directory. We explored different planning and RL strategies for building a multi-turn, task focused, and controlled dialog agent with little training data. In future, we will explore more integration possibilities and perform a comprehensive evaluation of the approach.

References

Ali, A.; and Gonzalez, A. 2016. Toward Designing a Realistic Conversational System: A Survey. In *Florida Artificial Intelligence Research Society Conference*. URL https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS16/paper/view/12776.

Bordes, A.; Boureau, Y.-L.; and Weston, J. 2017. Learning End-to-End Goal-Oriented Dialog. In *Proc. ICLR*.

Botea, A.; Muise, C.; Agarwal, S.; Alkan, O.; Bajgar, O.; Daly, E.; Kishimoto, A.; Lastras, L.; Marinescu, R.; Ondrej, J.; Pedemonte, P.; and Vodolan, M. 2019a. Generating Dialogue Agents via Automated Planning. In *https://arxiv.org/abs/1902.00771*.

Botea, A.; Muise, C.; Agarwal, S.; Alkan, O.; Bajgar, O.; Daly, E.; Kishimoto, A.; Lastras, L.; Marinescu, R.; Ondrej, J.; et al. 2019b. Generating dialogue agents via automated planning. *On Arxiv at: https://arxiv.org/abs/1902.00771* .

Clark, A.; Fox, C.; and Lappin, S. 2010. Handbook of Computation Linguistics and Natural Language Processing. In *Wiley, ISBN: 978-1-405-15581-6*.

Cohen, P. 2019. Foundations of Collaborative Task-Oriented Dialogue: What's in a Slot? In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, 198–209. Stockholm, Sweden: Association for Computational Linguistics. doi:10.18653/v1/W19-5924. URL https://www.aclweb.org/anthology/W19-5924.

Cohen, P. R. 2018. Back to the future for dialogue research: A position paper. *On Arxiv at: https://arxiv.org/abs/1812.01144* .

Crook, P. 2018. Statistical Machine Learning for Dialog Management: its history and future promise. In *AAAI DEEP-DIAL 2018 Workshop, at https://www.dropbox.com/home/AAAI2018 - DEEPDIALWorkshop/Presentations-Shareable?preview= Invited1-PaulCrook-AAAI_DeepDialog_Feb2018.pdf*.

Fung, P. N.; Chen, Y.-N. V.; Lin, Z.; and Madotto, A. 2020. Deeper Conversational AI. In *Neurips Tutorial, https://nips.cc/Conferences/2020/Schedule?showEvent=16657*.

Hirsch, J.; Nicola, G.; McGinty, G.; Liu, R.; Barr, R.; Chittle, M.; and Manchikanti, L. 2016. ICD-10: history and context. *American Journal of Neuroradiology* 37(4): 596–599.

Inouye, R. B. 2004. Minimizing the Length of Non-Mixed Initiative Dialogs. In Leonoor van der Beek, Dmitriy Genzel, D. M., ed., *ACL 2004: Student Research Workshop*, 7–12. Barcelona, Spain: Association for Computational Linguistics.

Li, F.; and Jagadish, H. 2016. Understanding natural language queries over relational databases. *ACM SIGMOD Record* 45(1): 6–13.

McTear, M.; Callejas, Z.; and Griol, D. 2016. Conversational Interfaces: Past and Present. In *The Conversational Interface. Springer, DOI: https://doi.org/10.1007/978-3-319-32967-3_4*.

Miller, A. H.; Feng, W.; Fisch, A.; Lu, J.; Batra, D.; Bordes, A.; Parikh, D.; and Weston, J. 2018. ParlAI: A Dialog Research Software Platform. *On Arxiv at: https://arxiv.org/abs/1705.06476* .

Muise, C.; Chakraborti, T.; Agarwal, S.; Bajgar, O.; Chaudhary, A.; Lastras-Montano, L. A.; Ondrej, J.; Vodolan, M.; and Wiecha, C. 2019a. Planning for Goal-Oriented Dialogue Systems. In *https://arxiv.org/abs/1910.08137*.

Muise, C.; Chakraborti, T.; Agarwal, S.; Bajgar, O.; Chaudhary, A.; Lastras-Montano, L. A.; Ondrej, J.; Vodolan, M.; and Wiecha, C. 2019b. Planning for goal-oriented dialogue systems. *arXiv preprint arXiv:1910.08137* .

Nothdurft, F.; Behnke, G.; Bercher, P.; Biundo, S.; and Minker, W. 2015. The Interplay of User-Centered Dialog Systems and AI Planning. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 344–353. Prague, Czech Republic: Association for Computational Linguistics. doi:10.18653/v1/W15-4646. URL https://www.aclweb.org/anthology/W15-4646.

Pasupat, P.; and Liang, P. 2015. Compositional semantic parsing on semi-structured tables. *On Arxiv at: https://arxiv.org/abs/1508.00305* .

Radlinski, F.; and Craswell, N. 2017. A Theoretical Framework for Conversational Search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, CHIIR '17, 117–126. New York, NY, USA: Association for Computing Machinery. ISBN 9781450346771. doi:10.1145/3020165.3020183. URL https://doi.org/10.1145/3020165.3020183.

Young, S.; Gašić, M.; Thomson, B.; and Williams, J. D. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5): 1160–1179.