Unsupervised Learning of KB Queries in Task-Oriented Dialogs

Dinesh Raghu[*1,2], **Nikhil Gupta**[†3], **and Mausam**[1]
[1]IIT Delhi, New Delhi, India
[2]IBM Research, New Delhi, India
[3]LimeChat, Gurgaon, India
diraghu1@in.ibm.com, nikhil@limechat.ai, mausam@cse.iitd.ac.in

Abstract

Task-oriented dialog (TOD) systems often need to formulate knowledge base (KB) queries corresponding to the user intent and use the query results to generate system responses. Existing approaches require dialog datasets to explicitly annotate these KB queries—these annotations can be time consuming, and expensive. In response, we define the novel problems of predicting the KB query and training the dialog agent, without explicit KB query annotation. For query prediction, we propose a reinforcement learning (RL) baseline, which rewards the generation of those queries whose KB results cover the entities mentioned in subsequent dialog. Further analysis reveals that correlation among query attributes in KB can significantly confuse memory augmented policy optimization (MAPO), an existing state of the art RL agent. To address this, we improve the MAPO baseline with simple but important modifications suited to our task.

To train the full TOD system for our setting, we propose a pipelined approach: it independently predicts when to make a KB query (query position predictor), then predicts a KB query at the predicted position (query predictor), and uses the results of predicted query in subsequent dialog (next response predictor). Overall, our work proposes first solutions to our novel problem, and our analysis highlights the research challenges in training TOD systems without query annotation.

1 Introduction

Task-oriented dialog (TOD) systems converse with users to accomplish specific tasks such as restaurant reservation (Henderson et al., 2014b), movie ticket booking (Li et al., 2017), or bus enquiry (Raux et al., 2005). In addition to the ability to converse, it is crucial for TOD systems to learn to formulate knowledge base (KB) queries based on user needs, and generate responses using the query results. An example TOD is shown in Figure 1, where during the conversation (at turn 2), the agent queries the KB based on the user needs, and then suggests the *Peking Restaurant* based on the retrieved results. Existing end-to-end approaches (Bordes and Weston, 2017; Madotto et al., 2018; Reddy et al., 2019) learn to formulate KB queries using manually annotated queries.

In real-world scenarios, human agents chat with users on messaging platforms. When the need to query the KB arises, the agent fires the query on a back-end KB application and uses the retrieved results to compose a response back on the messaging platform. The dialogs retrieved from these platforms contain just the user and the agent utterances, but the KB queries typically go undocumented. As existing approaches require KB annotations, they have to be manually annotated, which is expensive and hinders scalability. To eliminate the need for such annotations, we define the novel problem of training a TOD system without explicit KB query annotation. A key subtask of such a system is the unsupervised prediction of KB queries.

While there is no explicit query annotation, we observe that dialog data still offers *weak supervision* to induce KB queries—all the entities used by the agent in subsequent dialog should be returned by the correct query. For example, in Figure 1, the correct query should retrieve *Peking Restaurant* and its phone number. This suggests a

[*]D. Raghu is an employee at IBM Research. This work was carried out as part of PhD research at IIT Delhi.

[†]This work was done while Nikhil Gupta was a graduate student at IIT Delhi.

Transactions of the Association for Computational Linguistics, vol. 9, pp. 374–390, 2021. https://doi.org/10.1162/tacl_a_00372
Action Editor: Hang Li. Submission batch: 8/2020; Revision batch: 11/2020; Published 4/2021.

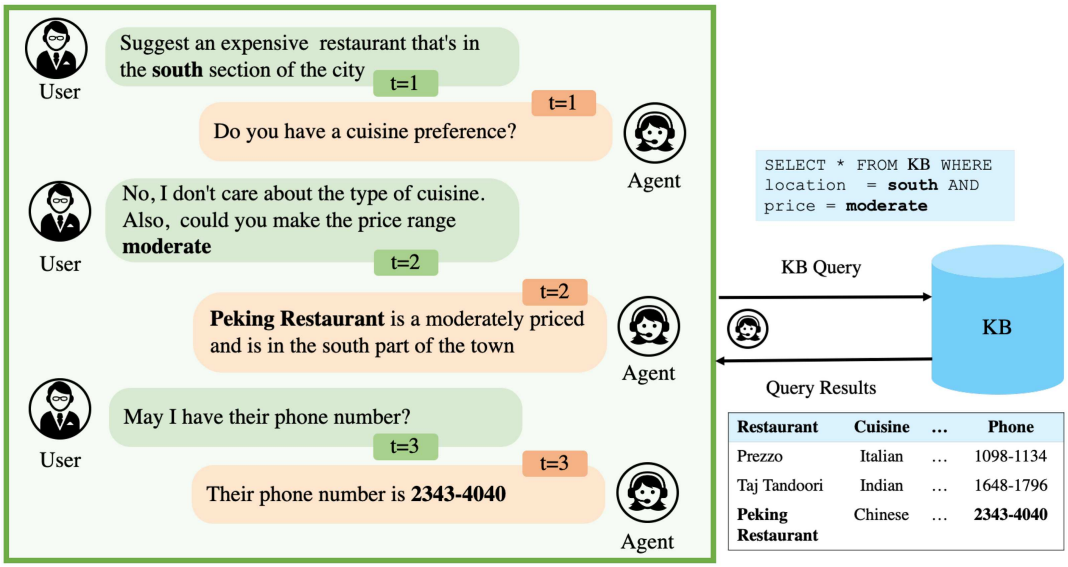

Figure 1: Example of a training task-oriented dialog. At turn 2, the agent first queries the KB based on the user requirement and then responds based on the retrieved results.

reinforcement learning (RL) formulation for query prediction, not unlike similar tasks in Question Answering (QA) (Artzi and Zettlemoyer, 2013) and conversational QA (Yu et al., 2019). However, there is one key difference between TOD and QA settings—in TOD only a few entities from the query results are used in subsequent dialog, whereas in QA all correct answers are provided at training time. This makes defining the reward function more challenging for our setting.

Our problem is further exacerbated by the issue of *correlated attributes*. Query attributes in TOD may exhibit significant correlation, like in the restaurant domain, cuisine and price range are often correlated. For example, most Japanese restaurants in a KB may be in expensive price range. As a result, presence and absence of expensive price range in the query could retrieve almost the same set of KB entities and hence similar rewards. This can confuse the weakly supervised query predictor. To counter this issue we present a baseline solution for KB query prediction by extending an existing policy optimization technique, memory augmented policy optimization (MAPO; Liang et al., 2018). Experiments show that our proposed modification significantly improves the query prediction accuracy.

To train a full TOD system without KB query annotation, we propose a pipelined solution. It uses three main components: (1) query position predictor predicts when a query must be made, (2) query predictor predicts the query at the turn predicted by position predictor, and (3) next response predictor generates next utterance based on dialog context, and predicted query's results. We train these components in a curriculum, due to the pipeline nature of the system. We find that overall our system obtains good dialog performance, and also learns to generalize to entities unseen during training.

We conclude with novel research challenges highlighted by our paper. In particular, we notice that even fully supervised TOD systems (trained with KB query annotation) suffer a significant loss in performance when they are evaluated with the protocol of using their own predicted queries and its query results (instead of using gold queries and gold results) in subsequent dialog. We believe that designing TOD systems with high performance under this evaluation protocol is the key next step towards making end-to-end TOD systems useful in real applications. We release all our resources for further research—training data, evaluation code, and code for TOD systems.[1]

2 Background & Related Work

Our work is on task-oriented dialogs and is closely related to the task of semantic parsing. We briefly discuss related work in both areas. Our query predictor is an extension of MAPO (Liang et al., 2018), which we describe in detail.

Task Oriented Dialogs: TOD systems are of two main types: traditional spoken dialog systems

[1] https://github.com/dair-iitd/mb-mapo.

(SDS) and end-to-end TOD systems. SDS (Wen et al., 2017; Williams et al., 2017) use hand-crafted states and state annotations on every utterance in the dialogs—a significant human supervision. End-to-end TOD systems (Reddy et al., 2019; Wu et al., 2019; Raghu et al., 2019) do not require state annotations but just the KB query annotations. There exist approaches (Chen et al., 2013, 2015) to induce state annotations in SDS, but we are the first to induce query annotations in end-to-end TOD systems. TOD systems cannot be learned with just the state annotations, additional state to KB query mapping/annotation is required. But no further annotations are needed to learn TOD system when query annotations are available.

We build on recent architectures, which use memory networks to store previous utterances and query results, and a generative copy decoder to construct the agent utterances (Madotto et al., 2018; Reddy et al., 2019; Raghu et al., 2019). An alternative approach maintains the entire KB in its neural model, bypassing the need for KB queries entirely (Dhingra et al., 2017; Eric et al., 2017). Unfortunately, such approaches can only work with small KBs. In contrast, our approach is scalable and does not impose restrictions on KB size.

Semantic Parsing: Semantic parsing is the task of mapping natural language text to a logical form (or program) (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005). To alleviate the need for gold program annotations, weakly supervised approaches have been proposed (Artzi and Zettlemoyer, 2013; Berant et al., 2013; Pasupat and Liang, 2015; Neelakantan et al., 2017; Haug et al., 2018). These weakly supervised approaches solve two main problems: (1) exploring large search space to find correct logical programs and (2) spurious program problem—where the policy learns an incorrect program that fetches the correct answer. To explore large search space better, Guu et al. (2017) used randomized beam search, Liang et al. (2017) and MAPO used systematic search. There exists several approaches to overcome the spurious query problem: crowd sourcing (Pasupat and Liang, 2016), spreading the probability mass over multiple reward earning programs (Guu et al., 2017), clustering similar natural language inputs and using their abstract representations (Goldman et al., 2018), and using overlap between the text spans in input and the generated programs (Wang et al., 2019; Dasigi et al., 2019; Misra et al., 2018). MAPO samples from a buffer of systematically explored reward earning programs based on their likelihood in the current policy (Liang et al., 2018) to tackle the spurious program problem. Our query prediction approach follows this literature, except in a dialog setting. As our approach is an extension of MAPO, we inherit the ability to effectively tackle the two issues.

These methods use an RL formalism, in which a logical form (query, in our case) $\mathbf{a}$ is predicted by an RL-policy $\pi_\theta(\mathbf{a}|c)$. Here, θ are the parameters of the RL agent, and c is the input, for example, a question (in our case, a dialog context). The policy π_θ is trained by maximizing the expected reward:

$$\mathcal{O}_{ER} = \mathbb{E}_{\mathbf{a}\sim\pi_\theta(\mathbf{a}|c)} R(\mathbf{a}|c, y) = \mathbb{E}_{\mathbf{a}\sim\pi_\theta(\mathbf{a})} R(\mathbf{a})$$

where R is the reward function and y is the gold answer for c. For simplicity, we drop the dependence of π_θ and R on c and y. REINFORCE (Williams, 1992) can be used to estimate the gradient of the expected reward. Using N queries sampled i.i.d. from the current policy, the gradient estimate can be expressed as:

$$\nabla_\theta \mathcal{O}_{ER} = \frac{1}{N} \sum_{k=1}^{N} \nabla_\theta \log \pi_\theta(\mathbf{a}_k) R(\mathbf{a}_k) \qquad (1)$$

When the search space is large and the rewards are sparse, relying on just the on-policy samples often leads to poor search space exploration. To overcome this, search is added on top of policy samples (Liang et al., 2017). We build on MAPO, which uses *systematic exploration* to identify non-zero reward queries for each training data and stores them in a memory buffer $\mathcal{B}$. The expected reward is computed as a weighted sum of two expectations: one over the queries inside the buffer $\mathcal{B}$ and the other over the remaining queries:

$$\begin{aligned} \mathcal{O}_{ER} &= \sum_{\mathbf{a}\in\mathcal{B}} \pi_\theta(\mathbf{a}) R(\mathbf{a}) + \sum_{\mathbf{a}\in\mathcal{A}-\mathcal{B}} \pi_\theta(\mathbf{a}) R(\mathbf{a}) \\ &= \pi_\mathcal{B} \mathbb{E}_{\mathbf{a}\sim\pi_\theta^+(\mathbf{a})} R(\mathbf{a}) + (1-\pi_\mathcal{B}) \mathbb{E}_{\mathbf{a}\sim\pi_\theta^-(\mathbf{a})} R(\mathbf{a}) \end{aligned} \qquad (2)$$

where $\mathcal{A}$ is the set of all possible queries, $\pi_\mathcal{B} = \sum_{\mathbf{a}\in\mathcal{B}} \pi_\theta(\mathbf{a})$ is total probability of all queries in the buffer, and $\pi_\theta^+(\mathbf{a})$ and $\pi_\theta^-(\mathbf{a})$ are the normalized probability distributions inside and outside the buffer respectively. The gradient of the first term is estimated exactly by enumerating all queries in the buffer, while the gradients of

the second term is estimated as in Equation 1 by sampling i.i.d queries from the current policy and rejecting them if they are present in the buffer $\mathcal{B}$.

A randomly initialized policy is likely to assign small probabilities to the queries in the buffer, and hence negligible contribution to the gradient estimation. To ensure queries in the buffer contribute significantly to the gradient estimates, MAPO clips $\pi_{\mathcal{B}}$ to α. The modified gradient estimate is

$$\nabla_\theta \mathcal{O}^c_{ER} = \pi^c_{\mathcal{B}} \mathbb{E}_{\mathbf{a}\sim\pi^+_\theta(\mathbf{a})} \nabla_\theta \log \pi_\theta(\mathbf{a}) R(\mathbf{a}) + (1-\pi^c_{\mathcal{B}}) \mathbb{E}_{\mathbf{a}\sim\pi^-_\theta(\mathbf{a})} \nabla_\theta \log \pi_\theta(\mathbf{a}) R(\mathbf{a}) \quad (3)$$

where $\pi^c_{\mathcal{B}} = \max(\pi_{\mathcal{B}}, \alpha)$. When the training begins, $\alpha > \pi_{\mathcal{B}}$ resulting in gradient estimates biased towards the queries in the buffer. Once the policy stabilizes, $\pi_{\mathcal{B}}$ gets larger than α and the estimates becomes unbiased.

Experience Replay: Experience replay (Lin, 1992) stores past experience in a buffer and reuses that to stabilize training and improve sample efficiency. Prioritized experience replay (Schaul et al., 2016) assigns priorities to the experiences and samples them based on the priorities to efficiently learn using them. MAPO and our approach use replay buffers that store non-zero reward queries. These queries include both past experiences and queries identified using systematic search. The sampling strategy is similar to prioritized experience replay, where the priorities are defined by the probability of the query in the current policy.

3 Task Definition & Baseline System

We first define our novel task of learning TOD system using *unannotated* dialogs – we name it uTOD. We then describe the KB query predictor in detail. Finally, we describe our proposed baseline uTOD system, which uses the KB query predictor.

3.1 Problem Definition

We represent a dialog d between a user u and an agent s as $\{c^u_1, c^s_1, c^u_2, c^s_2, \ldots, c^u_m, c^s_m\}$ where m denotes the number of turns in the dialog. Our goal is to train a uTOD system, which, for all turns i, takes the partial dialog-so-far $\{c^u_1, c^s_1, \ldots, c^u_{i-1}, c^s_{i-1}, c^u_i\}$ as input and predicts the next system response $\hat{c}^s_i$. Such a uTOD system is trained by the training data $\mathcal{D}$ comprising complete dialogs $\{d_j\}_{j=1}^{|\mathcal{D}|}$, and an associated knowledge base KB.

System responses in a dialog often use certain entities from KB. To accomplish this, a uTOD system can fire an appropriate query $\mathbf{a}$ to KB and fetch the query results E^a, and use those to generate its response. No explicit supervision is provided on the gold queries, at either training or test time. That is, a uTOD system neither knows which query was fired to KB, nor knows when it was fired. In this work we assume that the system can make only one KB query per dialog. Our overall uTOD baseline uses a pipeline of multiple components: a query position predictor, a query predictor, and a next response predictor.

3.2 KB Query Predictor

We first assume that we have access to the specific turn number $q, 1 \leq q \leq m$, where a KB query gets fired by the system (an assumption we relax in the next section). Thus, formally, we are given the *dialog context* $c = \{c^u_1, c^s_1, c^u_2, c^s_2, \ldots, c^u_q\}$, and KB, and the goal of query predictor is to output $\mathbf{a}$ that appropriately captures the user intent. We use the term *subsequent dialog* to refer to all utterances that follow the KB query–$\{c^s_q, c^u_{q+1}, \ldots, c^s_m\}$. In Figure 1, the subsequent dialog starts from the second agent utterance. Let E^s be the set of KB entities present in the subsequent dialog. In Figure 1, $E^s = \{$*Peking Restaurant*, *2343-4040*$\}$. We train the query predictor using RL, by providing feedback based on the predicted query's ability to retrieve the entities in E^s.

Our proposed baseline follows the literature on semantic parsing with weak supervision using RL, and treats the query prediction as equivalent to learning policy $\pi_\theta(\mathbf{a}|c)$ that takes the dialog context c as the input and generates a KB query $\mathbf{a} = a_1 a_2 \cdots a_T$. The KB query is a sequence of T actions, where each action is a word predicted by the query predictor. For a given context, the set of all possible actions is a union of a set of keywords[2] in the SQL query language, set of field names in KB, the set of all words in the context c and an <*eoq*> token to indicate the end of query. The query is generated auto-regressively as:

$$\pi_\theta(\mathbf{a}|c) = \prod_{t=1}^{T} \pi_\theta(a_t | a_{1:t-1}, c) \quad (4)$$

As the environment is deterministic, at each time t, the RL state can be fully defined by the dialog context c and the actions predicted so far $a_{1:t-1}$.

[2] {SELECT, *, =, FROM, AND, WHERE}.

RL gradients can be computed using REINFORCE or the MAPO algorithm.

The policy network is implemented with a standard encoder-decoder architecture for TOD systems. The context is encoded using a multi-hop memory encoder (Sukhbaatar et al., 2015) with a bag of sequences memory (Raghu et al., 2019). In the bag of sequences memory, the context is represented as a set of utterances and each utterance as a sequence of words. Each utterance is encoded using a bi-directional GRU (Cho et al., 2014).

The KB query is generated one word at a time by a copy-augmented sequence decoder (Gu et al., 2016). The search space is the output space of this variable length, copy-augmented sequence decoder. At each time step, the decoder computes a copy distribution over words in the dialog context and a generate distribution over words KB field names and SQL keywords. Finally, a soft gate (See et al., 2017) is used to combine the two distributions and a word is sampled from the combined distribution. By allowing only copying words from the dialog context to the values in the SQL WHERE clauses, we reduce the decoder's ability to predict spurious queries. Following Ghazvininejad et al. (2016), we use beam search guided by the SQL grammar to generate only syntactically correct queries.

3.2.1 Reward Functions

We now describe a vanilla reward function considered for training the RL agent. For a given (c, E^s) pair, the query $\mathbf{a}$ is predicted using c and the reward is computed using E^s. Since partial query isn't meaningful, the reward is computed only at time $t = T$, when the complete query is generated.

Let E^a be the set of entities retrieved by the query $\mathbf{a}$, then the reward at time T must ensure that all entities present in the subsequent dialog are retrieved by the query. Moreover, the query should be penalized for retrieving more than required entities. This can be operationalized as follows:

$$R(\mathbf{a}|c, E^s) = \mathbb{1}_{\text{recall}(E^s, E^a)=1}.\text{prec}(E^s, E^a) \quad (5)$$

The recall based indicator ensures that all entities are retrieved, and precision penalizes retrieval of a large number of entities. While a reasonable first solution, our initial experiments showed that MAPO with this reward function does not achieve satisfactory results. We now discuss the challenging aspects of our task, and our improved baseline, *multi buffer* MAPO, to resolve those issues.

3.2.2 Multi Buffer MAPO (m$\mathcal{B}$-MAPO)

Our problem can become particularly challenging if KB has correlated attributes. To understand this, consider two types of KB queries: (1) partial intent query and (2) complete intent query. A partial query is one that is partially correct, namely, captures a part of the user's intent, and does not include any incorrect clauses. A complete query contains all (and only) correct clauses expressed in the intent. For example (a) in Table 1, the first two queries are complete ones, whereas rest are partial. Query 4 in column (b) is a spurious query that can retrieve the same result as the complete intent query. This spurious query is not a partial query, as it does not capture any of the user needs. Our query decoder allows only words to be copied from the context when predicting the query, thus reducing the policy network's ability to learn spurious queries.

Because of the nature of weak supervision in uTOD (we get to see only a subset of entities in subsequent dialog), it is possible that multiple queries can achieve non-zero rewards for a dialog context. It is further possible that a *specific* partial query can achieve non-zero rewards in many dialog contexts. Example, the query ''*price=moderate*'' fetches non-zero rewards for both intents in Table 1. Such phenomena can confuse an RL agent, to the extent that it may end up incorrectly learning a single partial query as the best query for many contexts.

This problem is further exacerbated if some query attributes in KB are correlated. For example (a) in Table 1, let us assume that KB has 8 *chinese* restaurants out of which 7 have *moderate* price range, that is, cuisine and price range are highly correlated. Here, results of the partial query ''cuisine=*chinese*" would contain only one additional restaurant compared to the complete query. As a result, they will receive almost the same reward during training. This could potentially get extreme in certain cases, where presence or absence of an attribute makes no difference, giving little signal to an RL agent. As the number of attributes in an intent increases, the problem can get harder and harder. In our preliminary experiments using MAPO with reward from Equation 5, the RL agent often produced partial queries leading to further errors in subsequent dialog.

User Intent	**(a)**	user needs a restaurant that serves Chinese with moderate price range	**(b)**	user needs a Japanese restaurant in moderate price range
KB Queries		1. cuisine=*chinese* AND price=*moderate* 2. price=*moderate* AND cuisine=*chinese* 3. cuisine=*chinese* 4. price=*moderate*		1. cuisine=*japanese* AND price=*moderate* 2. price=*moderate* AND cuisine=*japanese* 3. cuisine=*japanese* 4. phone=*98232-66789*

Table 1: Summary of dialogs and a few examples of non-zero reward KB queries for the corresponding dialogs. For simplicity, the SELECT clauses are removed from the KB queries.

In our datasets intents can be described by SELECT queries with multiple WHERE clauses; thus, every partial query is a prefix of some complete query (e.g., query 3 in Table 1). Upon further analysis of model behavior, we observed that MAPO often maintained both partial prefix queries and complete queries, but still learned a model to output the partial ones.

To understand this surprising observation, we first note that auto-regressive decoders, due to probability multiplication at every decode step, generally prefer *shorter* grammatical sentences (partial prefix query in our case) to longer ones. This is particularly likely to happen when the decoder gets randomly initialized at the start of training. A partial query in the buffer that is assigned a high probability makes a higher contribution in the gradient estimation. Moreover, because of correlated attributes, this query may have a fairly high reward. This results in the model believing it to be a good query, and changing the parameters to increase its probability further. The only way RL can break out of this vicious cycle is if the complete query is explored often—however, MAPO's in-buffer sampling probabilities are proportional to network-assigned probabilities, leading to an ineffective in-buffer exploration. This issue is general, but gets extreme in our problem due to a combination of (1) real-valued (not just 0/1) rewards, (2) prefix queries getting high rewards due to correlated attributes, and (3) prefix queries getting sampled more often due to decoder's bias in favoring shorter queries.

In response, our proposed baseline extends MAPO to maintain multiple buffers, so that complete intent queries with high rewards can be prioritized over other queries during gradient estimation. We use two buffers with MAPO: a buffer $\mathcal{B}_h$ to store all queries with the highest reward for a dialog context and a buffer $\mathcal{B}_o$ to store all other queries whose rewards are non-zero and less than the highest. The expected reward is now computed as:

$$\begin{aligned}\mathcal{O}_{ER} &= \sum_{\mathbf{a}\in\mathcal{B}_h}\pi_\theta(\mathbf{a})R(\mathbf{a}) + \sum_{\mathbf{a}\in\mathcal{B}_o}\pi_\theta(\mathbf{a})R(\mathbf{a}) \\ &\quad + \sum_{\mathbf{a}\in\mathcal{A}-(\mathcal{B}_h\cup\mathcal{B}_o)}\pi_\theta(\mathbf{a})R(\mathbf{a}) \\ &= \pi_{\mathcal{B}_h}\mathbb{E}_{\mathbf{a}\sim\pi_\theta^{h+}(\mathbf{a})}R(\mathbf{a}) + \pi_{\mathcal{B}_o}\mathbb{E}_{\mathbf{a}\sim\pi_\theta^{o+}(\mathbf{a})}R(\mathbf{a}) \\ &\quad + (1-\pi_{\mathcal{B}_h}-\pi_{\mathcal{B}_o})\mathbb{E}_{\mathbf{a}\sim\pi_\theta^{-}(\mathbf{a})}R(\mathbf{a}) \qquad (6)\end{aligned}$$

where π_θ^{h+}, π_θ^{o+} and π_θ^- are the normalized probability distributions of queries in $\mathcal{B}_h$, queries in $\mathcal{B}_o$, and all other queries, respectively. $\pi_{\mathcal{B}_h} = \sum_{\mathbf{a}\in\mathcal{B}_h}\pi_\theta(\mathbf{a})$ and $\pi_{\mathcal{B}_o} = \sum_{\mathbf{a}\in\mathcal{B}_o}\pi_\theta(\mathbf{a})$ are the total probabilities assigned by the policy π_θ of all queries in $\mathcal{B}_h$ and $\mathcal{B}_o$ respectively. As the complete intent queries mostly have the highest rewards for a given context, they are placed in $\mathcal{B}_h$, and as the partial prefix queries usually have rewards less than the complete queries, they are placed in the other buffer $\mathcal{B}_o$. To ensure that all non-zero reward queries are explored, each buffer is made to contribute to the gradient estimation. To ensure that each buffer contributes significantly to the gradient estimation, we clip $\pi_{\mathcal{B}_h}$ using $\max(\pi_{\mathcal{B}_h}, \alpha_h)$ and $\pi_{\mathcal{B}_o}$ using $\min(\max((1-\pi^c_{\mathcal{B}_h})\alpha_o, \pi_{\mathcal{B}_o}), (1-\pi^c_{\mathcal{B}_h}))$. α_h and α_o are hyperparameters whose value can be between 0 and 1. α_h ensures that the highest reward buffer gets assigned a certain weight during gradient estimation. α_o ensures that the queries in $\mathcal{B}_o$ are assigned at least a certain fraction of probability mass unused by the queries in $\mathcal{B}_h$. The estimator is biased when the training starts, and can becomes unbiased when $\pi_{\mathcal{B}_h} > \alpha_h$ and $\pi_{\mathcal{B}_o} > (1-\pi_{\mathcal{B}_h}))\alpha_o$. Based on the clipped probabilities, the gradients are estimated as:

$$\begin{aligned}\nabla_\theta\mathcal{O}^c_{ER} &= \pi^c_{\mathcal{B}_h}\mathbb{E}_{\mathbf{a}\sim\pi_\theta^{h+}(\mathbf{a})}\nabla_\theta\log\pi_\theta(\mathbf{a})R(\mathbf{a}) \\ &\quad + \pi^c_{\mathcal{B}_o}\mathbb{E}_{\mathbf{a}\sim\pi_\theta^{o+}(\mathbf{a})}\nabla_\theta\log\pi_\theta(\mathbf{a})R(\mathbf{a}) \\ &\quad + (1-\pi^c_{\mathcal{B}_h}-\pi^c_{\mathcal{B}_o})\mathbb{E}_{\mathbf{a}\sim\pi_\theta^{-}(\mathbf{a})}\nabla_\theta\log\pi_\theta(\mathbf{a})R(\mathbf{a}) \qquad (7)\end{aligned}$$

Finally, if a new query is found that has a higher reward than queries in $\mathcal{B}_h$, then the buffers are

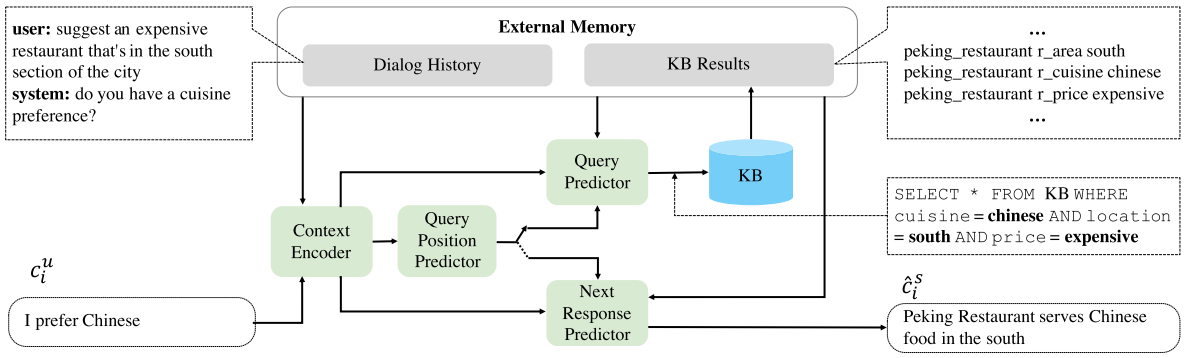

Figure 2: The architecture of the uTOD system.

updated: That query is added to $\mathcal{B}_h$, and all other queries in that buffer are removed and added to $\mathcal{B}_o$. The proposed approach can be used for estimating the policy gradients for any deterministic environments with discrete actions and non-binary rewards.

3.3 uTOD System

We now describe the remaining parts of the uTOD system that learns to predict the next response using unannotated dialogs. The system has three main components: (1) query position predictor, (2) query predictor (as described in the previous section), and (3) next response predictor. The system architecture is shown in Figure 2. The position predictor is a binary classifier that decides whether the KB query is to be predicted at the given turn or not. The query predictor generates a KB query at the turn predicted by the position predictor. Finally, the next response predictor takes the dialog context along with the query results (if a KB query has been made) and predicts the next system response.

In order to train the position predictor, each train dialog must be annotated with the turn at which the KB query is to be made. As there are no gold labels available, we heuristically provide this supervision. We identify the turn ($\tilde{q}$) at which the agent response contains a KB entity that was never seen in the dialog context. We mark $\tilde{q}$ as the heuristic label for training position predictor.[3] For example, in Figure 1, $t = 2$ is the turn at which the agent response contains a KB entity *Peking Restaurant* that was never used in the dialog context.

The position predictor takes the dialog context as input and encodes it using a multi-hop memory encoder, as described in Section 3.2. The encoder output is then passed through a linear layer followed by sigmoid function to generate the probability of the binary label. At training time, KB query predictor learns to generate KB queries at the turns ($\tilde{q}$) annotated by the heuristic. During test time, a query is generated at the turn ($\hat{q}$) predicted by the position predictor.

The query predictor can be seen as annotating all the train dialogs with KB queries. Thus, after this annotation, any existing end-to-end TOD system can be used for next response prediction. For our experiments, we use BoSsNet as the underlying response predictor (Raghu et al., 2019). BoSsNet takes as input (i) the dialog context and (ii) the ground truth KB query results (during both train and test) to generate the response. As we assume the ground truth KB queries are unavailable, we use the results of the predicted query results instead.

Since our system uses a pipeline of components, it is trained in a training curriculum, which is divided into three phases. In *phase-1*, the query predictor is trained. Once each train dialog is annotated with KB queries, the next response predictor is trained in *phase-2*. Finally, in *phase-3* the position predictor is trained. We train the position predictor after the response predictor as we observed that initializing the position predictor encoder with the weights of the response predictor encoder yields better position prediction performance.

4 Experimental Setup

We now describe the datasets used, the comparison algorithms and the evaluation metrics for our task. We release all software and datasets from our experimental setup for further research.

[3]This heuristic labeling matched 80% of gold labels in our datasets.

	CamRest	DSTC2
Train Dialogs	406	1279
Val Dialogs	135	324
Test Dialogs	135	1051
Avg. no. of turns	4.06	7.94
Rows/Fields in KB	110/8	108/8
Vocab Size	1215	958

Table 2: Statistics of CamRest and DSTC2 datasets.

4.1 Datasets

We perform experiments on two task-oriented dialog datasets from the restaurant reservation domain: CamRest (Wen et al., 2016) and DSTC2 (Henderson et al., 2014a). CamRest676 is a human human dialog dataset with having just one KB query annotated in them. DSTC2 is a human-bot dialog dataset. We filtered the dialogs from this dataset that had more than one KB query. Table 2 summarizes the statistics of the two datasets. As CamRest and DSTC2 are originally designed for dialog state tracking, we use the versions that are suitable for end-to-end learning, made available by Raghu et al. (2019) and Bordes and Weston (2017), respectively. Both these datasets have KB query annotations. We remove these annotations from the training dialogs to create datasets for our task. We use these annotations for evaluating the performance of various algorithms.

4.2 Comparison Baselines

We compare various gradient estimation techniques to train the policy network (defined in Section 3.2) for the task of KB query prediction as follows.

REINFORCE (Williams, 1992): uses on-policy samples to estimate the gradient as in Equation 1, with the reward function defined in Equation 5.

BS-REINFORCE (Guu et al., 2017): uses all the queries generated by beam search to estimate the gradients. The reward for each query is a product of the reward defined in Equation 5 and the likelihood of the query in the current policy.

RBS-REINFORCE (Guu et al., 2017): uses all the queries generated by ϵ-greedy randomized beam search (RBS) to estimate the gradients. At every time step, RBS samples a random continuation as opposed to a highest scoring continuation with a probability ϵ.

MAPO (Liang et al., 2018): uses on-policy samples and a buffer of non-zero reward queries to estimate the gradients. It uses the reward function defined in Equation 5.

m$\mathcal{B}$-MAPO: the proposed approach, multi buffer MAPO, which uses a buffer with highest reward queries and a buffer with other non-zero reward queries to estimate the gradients. This algorithm uses the reward function defined in Equation 5.

SL: uses the gold KB queries as direct supervision. We train the policy network proposed in Section 3.2 with cross entropy loss.

SL+RL: we combine weak supervision using RL as a secondary source of knowledge on top of supervised cross entropy loss for better training. This is analogous to the benefits of additional prior knowledge (e.g., as constraints) on top of supervision for small datasets (Nandwani et al., 2019). In our experiments, *total loss* = *Cross-Entropy* $-\lambda.\mathcal{O}_{ER}$. Equation 6 is used for computing $\mathcal{O}_{ER}$.

We also compare the full uTOD dialog engines, built using these RL approaches for query prediction; we name them uTOD$^{\text{REINFORCE}}$, uTOD$^{\text{MAPO}}$, and uTOD$^{\text{mMAPO}}$. We also compare these to a fully supervised TOD system that uses KB query annotations during training (aTOD). We report two variants of the aTOD system based on the supervised query predictors used: aTOD$^{\text{SL}}$ and aTOD$^{\text{SL+RL}}$.

4.3 Evaluation Metrics

As we have gold annotations, we evaluate the KB query predictor and position predictor separately, in addition to the overall TOD system. KB query predictors are evaluated based on accuracy—the fraction of dialogs where the gold KB queries are predicted. We also report two other metrics: total reward and the PIQ ratio—the fraction of dialogs where a partial intent query was outputted, that is, the predicted query captured only a subset of gold query attributes.

A query position predictor's performance is also measured using accuracy. The predictor is considered correct only if it predicts 1 at the turn at which query is made and 0 for all turns before that. We also compute *turn difference* as the absolute difference between the turn at which the classifier predicts true and the turn corresponding to the gold label in the annotated dialog. The smaller the average turn difference, the better is the classifier.

The next response prediction is evaluated based on its ability to match the gold responses at every turn. We use standard metrics of BLEU (Papineni et al., 2002) and entity F1 to measure the similarity between predicted and gold responses. Entity F1 is the average of F1 scores computed for each response. We also report Entity F1 KB for entities that can only be copied from the KB results, to emphasize importance of using query results in subsequent dialog.

We note that our setup for evaluating dialog engines looks similar to supervised TOD systems, but has a subtle but important difference. In standard supervised dialog evaluations, for a given turn, the full dialog context is provided and the system is evaluated on correctly predicting the next utterance. Thus, even if the system made an incorrect query at an earlier turn, the subsequent turns will be shown correct query and correct KB query results.

However, since our goal is to assess the response prediction without query annotation, we remove such annotation from the test dialogs also. That is, for the test dialogs subsequent to making the query, all responses are generated based on the *predicted* KB queries at predicted position $\hat{q}$ and their query results. This makes the evaluation much more realistic, but also quite challenging for current dialog engines. This is the key reason why aTOD performance in Table 5 is much lower than results reported in the BossNet paper (Raghu et al., 2019).

We perform two human evaluation experiments to compare (1) *informativeness* – the ability to effectively use the results to generate responses and convey the information requested by the user, and (2) *grammar* – ability to generate grammatically correct and fluent responses. Both the dimensions were annotated on a scale of (0–2). As the primary focus of our work is to evaluate the ability of a TOD system to effectively use the annotated query results, we only collect judgements for responses that occur *after* the KB query. We sampled 100 random dialog-context from CamRest dataset and collected judgments from 2 judges for 4 systems, namely, uTOD$^{\text{REINFORCE}}$, uTOD$^{\text{MAPO}}$, uTOD$^{\text{m}\mathcal{B}-\text{MAPO}}$, and aTOD. We collected a total of 1600 labels from the judges.

4.4 Implementation Details

We implemented our system using TensorFlow (Abadi et al., 2016). We identify hyperparameters based on the evaluation of the held-out validation sets. We sample word embedding, hidden layer, and cell sizes (es) from $\{32, 64, 128, 256, 512\}$, learning rates ($lr$) from $\{10^{-3}, 25 \times 10^{-4}, 5 \times 10^{-4}, 10^{-4}\}$, α_o and λ from increments of 0.1 between [0.1, 0,9], ϵ from $\{0.05, 0.1, 0.15, 0.2\}$, and α_h from increments of 0.1 between [0.5, 0,9]. The hyper-parameters that $(\alpha_h, \alpha_o, \lambda, \epsilon, es, lr)$ achieved the best validation rewards were $(0.5, 0.1, 0, 0.15, 256, 5\times10^{-4})$ and $(0.6, 0.1, 0.1, 0.15, 256, 25 \times 10^{-4})$ for DSTC2 and CamRest, respectively. As our response predictor is BossNet, we used the best performing hyper-parameters reported by Raghu et al. (2019) for each dataset. Total accumulated validation rewards is used as a early stopping criteria for training the query predictor and BLEU for the training the response predictor.

5 Experiments

Our experiments evaluate three research questions:

1. *Query Predictor Performance:* How does the performance of m$\mathcal{B}$-MAPO compare to other gradient estimation techniques?

2. *Query Position Predictor Performance:* How does the proposed position predictor perform on the two datasets?

3. *Next Response Predictor Performance:* How do responses from TOD systems trained with unannotated dialogs compare to the ones trained with KB query annotated dialogs?

5.1 Query Predictor Performance

To measure the query predictor performance, we generate the KB queries at the gold position during train and test. We only use the gold queries during test to measure the performance. Table 3 reports the KB query prediction accuracy, PIQ ratio, and total test rewards achieved by various gradient estimation techniques. m$\mathcal{B}$-MAPO significantly outperforms MAPO on both datasets. The performance gain comes from m$\mathcal{B}$-MAPO's ability to address correlated attributes in KB and the frequent sampling of highest reward queries from the buffer to prevent the policy from learning common partial intent queries. Compared to MAPO, m$\mathcal{B}$-MAPO reduces the PIQ ratio by 55% on DSTC2 and 54% on CamRest, and achieves considerably higher rewards.

The failure of REINFORCE highlights that using just the on-policy samples to estimate policy

	Accuracy		PIQ Ratio		Total Test Rewards	
	DSTC2	**CamRest**	**DSTC2**	**CamRest**	**DSTC2**	**CamRest**
REINFORCE	0.00	0.00	0.00	0.00	0.00	0.00
BS-REINFORCE	0.00	0.00	0.00	0.004±0.008	0.00	0.01±0.03
RBS-REINFORCE	0.00	0.002±0.005	0.00	0.02±0.05	0.00	0.24±0.26
MAPO	0.25±0.01	0.10±0.01	0.61±0.01	0.54±0.05	77.13±3.6	11.8±0.5
m$\mathcal{B}$-MAPO	**0.68±0.03**	**0.62±0.03**	**0.06±0.03**	**0.09±0.02**	**169.98±7.6**	**23.4±1.1**
SL	0.78±0.02	0.59±0.06	0.09±0.02	0.09±0.03	175.9±2.7	21.7±1.4
SL+RL	0.78±0.02	0.66±0.04	0.09±0.02	0.09±0.03	175.9±2.7	23.7±1.3

Table 3: Accuracy of KB query prediction of m$\mathcal{B}$-MAPO and other algorithms on CamRest and DSTC2 on 10 runs. Partial intent query (PIQ) ratio is the fraction of partial intent queries predicted.

gradients is inadequate for exploring large combinatorial search spaces with sparse rewards. Both MAPO and m$\mathcal{B}$-MAPO use queries in the buffer, which are explored using systematic search. These guide the policy towards the parts of the search space that are likely to yield non-zero rewards.

We notice that, surprisingly, m$\mathcal{B}$-MAPO achieves slightly better accuracy than even the supervised (SL) baseline on CamRest dataset. Further analysis reveals that on this dataset, supervised learner achieves a train accuracy of 95%, while m$\mathcal{B}$-MAPO achieves only 75% (but higher test accuracy). This suggests that the supervised learner is overfitting, which is conceivable since CamRest is a relatively small dataset (406 train dialogs, see Table 2). Because m$\mathcal{B}$-MAPO is learning with weak supervision, it is solving a much harder problem, which makes it harder to overfit. Moreover, sometimes m$\mathcal{B}$-MAPO may simply learn partial intent queries if they generalize better. This added flexibility helps avoid overfitting in the small dataset. Our error analysis reveals that 25% of queries generated by supervised learner had non-entity words as values in the WHERE clause. For example,‘‘please" was predicted as a cuisine. This shows the model has learned to pick up spurious signals to predict certain attributes. In contrast, only 11% to 13% of the queries predicted by MAPO and m$\mathcal{B}$-MAPO exhibited this problem. To prevent the supervised learner from overfitting, SL+RL combines the weak supervision using RL as a secondary source of knowledge on top of cross entropy loss. This increased the accuracy by 7 points and percent of queries with non-entity words reduced from 25% to 11%.

On the other hand, compared to supervised learner, m$\mathcal{B}$-MAPO is 10 accuracy points lower on DSTC2. The difference in performance can be attributed to two factors. First, DSTC2 is a larger dataset, which enables supervised learner to train well. Second, in this dataset, there are 12% of dialogs where overconstrained queries (those that have even more attributes than the gold) fetch better rewards than the gold. For instance, in Table 1 example (a), say the query ‘‘cuisine=*chinese* AND price=*moderate* AND location=*west*" fetches all the entities mentioned in subsequent dialog. Even though it has one additional attribute compared to the actual user intent (or gold query), it will likely contain fewer unused entities than the gold query. Thus m$\mathcal{B}$-MAPO will assign a higher reward to this overconstrained query, and will encourage the training to output this. This confuses m$\mathcal{B}$-MAPO, leading to a significant performance gap from supervised learner.

Dynamics of the Total Buffer Probabilities: $\pi_{\mathcal{B}_h}$ and $\pi_{\mathcal{B}_o}$ are the sum of probabilities of all the queries in buffers $\mathcal{B}_h$ and $\mathcal{B}_o$, respectively. We now discuss the dynamics of $\pi_{\mathcal{B}_h}$ and $\pi_{\mathcal{B}_o}$ during training. We define average $\pi_{\mathcal{B}_h}$ and average $\pi_{\mathcal{B}_o}$ as the average of total buffer probabilities across all the examples in train. Figure 3 shows the average $\pi_{\mathcal{B}_h}$ and average $\pi_{\mathcal{B}_o}$ after each train epoch on DSTC2. Queries in B_o are typically shorter (partial queries) compared to the queries in B_h and so they get assigned a higher probabilities when the policy is randomly initialized. Hence during initial epochs average $\pi_{\mathcal{B}_o}$ is higher than average $\pi_{\mathcal{B}_h}$. But as the training proceeds, the policy learns to assign higher probability to the (longer) queries in B_h as a result of clipping the buffer probabilities defined by α_h. The gradients are biased towards the queries in B_h during the first few epochs, but as the policy converges we can see that the average $\pi_{\mathcal{B}_h}$ reaches close to α_h (0.5 for DSTC2) making the gradient estimates unbiased.

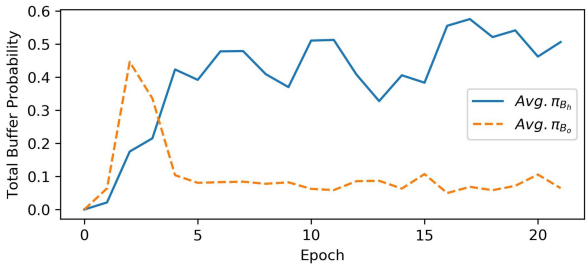

Figure 3: Dynamics of total buffer probabilities $\pi_{\mathcal{B}_h}$ and $\pi_{\mathcal{B}_o}$ while training on DSTC2.

	Accuracy		ATD	
	w/o PT	w/ PT	w/o PT	w/ PT
CamRest	0.47	**0.54**	0.72	**0.56**
DSTC2	0.21	**0.28**	2.26	**2.20**

Table 4: Position predictor performance with and without pre-training (PT). ATD = average turn difference.

5.2 Query Position Predictor Performance

To study the performance of the query position predictor, we use the gold positions to train the classifier and to measure the performance during test. Table 4 shows the accuracy and the average turn differences for the task of query position prediction. The predictor is evaluated in two settings: one initialized with random weights (w/o PT) and one with the encoder network pre-trained for the task of response prediction (w/ PT). The response prediction tasks helps the network identify parts of dialog context that are crucial for generating the response. This additional information helps improve the accuracy by 7 points on both datasets.

The overall performance for CamRest is acceptable, since average turn difference is less than 1. The errors are due to natural variations in the policy followed by the agent (human in the Wizard of Oz study)—sometimes the agent fires the query before all possible attributes are specified by the user, and at other times, the agent requests for missing attributes and only then fires a query. However, performance in DSTC2 is somewhat limited. In addition to natural agent policy variations, there are artifacts of speech data, because DSTC2 are speech transcripts of human-bot conversations. For exmaple, the bot often re-confirms an already specified attribute to reduce speech recognition errors. As DSTC2 dataset uses transcripts and not actual speech, this re-affirmation is redundant, but is still present. This confuses the model since it has to decide between whether to make a query or request a re-affirmation.

5.3 TOD System Performance

To study the TOD system performance, we neither use the gold positions nor the gold queries. We use the heuristic defined in Section 3.3 to label the positions. These labelled positions are used to train the position predictor and as the position at which the query predictor generates the query during train. At test time, the query predictor generates the query at the predicted position.

We study the performance of uTOD systems (next response prediction) trained using the dialogs whose queries were predicted by various query prediction algorithms. For this evaluation, test dialogs are first divided into (context, response) pairs. The system predicts a response for each turn and the predicted response is compared with the gold response. When predicting responses in the subsequent dialog, the results of the predicted query are appended to the context. For the aTOD system, results from gold queries are used during train, and the results of predicted queries are used during test.

Entities in responses can be divided into two types: context entities and KB entities. Context entities are present in the dialog context. For example, in Figure 1, agent utterance in turn 2 has two context entities (‘‘moderate’’ and ‘‘south’’) and one KB entity (‘‘Peking Restaurant’’). For the response to contain the correct KB entity, all of position predictor, query predictor, and next response predictor must work together. To assess this, we report *KB Entity F1*, which judges the match between gold and predicted KB entities used in the utterance.

Table 5 shows the performance of various TOD systems on two datasets. We see that the uTOD system trained using m$\mathcal{B}$-MAPO is only a few points lower than aTOD system. This underscores the value of our gradient computation scheme. m$\mathcal{B}$-MAPO is significantly better than MAPO; our analysis reveals that MAPO frequently returns partial queries which have a larger set of results —this confuses the response predictor, reducing KB Entity F1 substantially.

Most entities predicted by REINFORCE are just context entities copied from the context and their contribution dominates the entity F1 (all) score. As REINFORCE fails completely in generating

	DSTC2			CamRest		
	Ent. F1 KB	**Ent. F1** All	**BLEU**	**Ent. F1** KB	**Ent. F1** All	**BLEU**
uTOD$^{\text{REINFORCE}}$	0.11±0.01	0.36±0.02	**51.52±0.91**	0.02±0.01	0.34±0.02	**15.09±0.56**
uTOD$^{\text{MAPO}}$	0.14±0.01	0.36±0.01	46.27±1.63	0.14±0.01	0.31±0.03	12.90±0.75
uTOD$^{\text{m}\mathcal{B}-\text{MAPO}}$	**0.21±0.02**	**0.38±0.02**	47.52±1.27	**0.23±0.02**	**0.35±0.02**	13.11±0.86
aTOD$^{\text{SL}}$	0.24±0.02	0.41±0.02	48.35±1.58	0.25±0.03	0.36±0.02	13.80±1.04
aTOD$^{\text{SL+RL}}$	0.24±0.02	0.41±0.02	48.35±1.58	0.29±0.03	0.41±0.02	14.68±0.85

Table 5: Performance of various uTOD systems and aTOD system on 10 runs.

	DSTC2	CamRest
uTOD$^{\text{REINFORCE}}$	0.02±0.01	0.00±0.00
uTOD$^{\text{MAPO}}$	0.08±0.01	0.10±0.01
uTOD$^{\text{m}\mathcal{B}-\text{MAPO}}$	**0.17±0.02**	**0.21±0.02**
aTOD$^{\text{SL+RL}}$	0.20±0.01	0.28±0.02

Table 6: KB entity F1 achieved by various TOD systems on OOV test set.

	Info.	**Grammar**
uTOD$^{\text{REINFORCE}}$	0.20	0.89
uTOD$^{\text{MAPO}}$	0.36	1.18
uTOD$^{\text{m}\mathcal{B}-\text{MAPO}}$	**0.64**	**1.38**
aTOD$^{\text{SL+RL}}$	1.08	1.39

Table 7: Human evaluations on CamRest.

correct queries, the response predictor is forced to memorize the KB entities rather than inferring them from the query results. This results in very low KB entity F1. As the dialog context often has no query results, the system's only objective becomes generation of good language. Due to this, it achieves a higher BLEU score compared to other systems.

Human Evaluation: We report the human evaluation[4] results on 100 random context-response pairs for CamRest dataset in Table 7. uTOD$^{\text{m}\mathcal{B}-\text{MAPO}}$ outperforms other uTOD baselines on both informativeness and grammar. It was surprising to see uTOD$^{\text{REINFORCE}}$ perform poorly on grammar. Further investigation showed that often the responses generated were missing entities that made them look incomplete. For example, the response ''is a restaurant in the moderate part of town . would you like their phone number'' has a missing restaurant name at the start of the sentence. We measure the inter-annotator agreement using Cohen's Kappa (κ) (Cohen, 1960). The agreement was substantial for informativeness ($\kappa = 0.62$) and moderate ($\kappa = 0.45$) for grammar.

Disentanglement Study: TOD systems can learn to predict KB entities in the response either by inferring (copying) them from the query results or memorizing (generating) them. Only systems that learns to copy them from the results will generalize to entities unseen during train. To test this ability to generalize, we follow Raghu et al. (2019) and construct an OOV test set (and a corresponding KB) such that it contains entities unseen during train. Table 6 shows KB entity F1 of various TOD systems on this OOV test set. The low numbers for uTOD$^{\text{REINFORCE}}$ confirm that it memorizes KB entities. uTOD$^{\text{m}\mathcal{B}-\text{MAPO}}$ achieves best scores, and is 3–7 points short of supervised TOD.

Qualitative Example We qualitatively compare the performance of various TOD systems using the example shown in Table 8. The example demonstrates the ability of uTOD$^{\text{m}\mathcal{B}-\text{MAPO}}$ to generate the correct query and use the KB results to generate an appropriate response. uTOD$^{\text{MAPO}}$ generates a partial query that has a larger set of results—this confuses the response predictor, and hence an incorrect restaurant is copied in the response. uTOD$^{\text{REINFORCE}}$ fails to generate a valid query and so the response predictor generates a random restaurant memorized during train. The response generated by all approaches contain the correct context entities (*cheap* and *south*), but only responses of uTOD$^{\text{m}\mathcal{B}-\text{MAPO}}$ and uTOD$^{\text{SL}}$ contain the appropriate KB entity requested by the user.

6 Discussion & Research Challenges

We now discuss novel research questions identified in this research.

[4]We used two in-house (non-author) judges. One was an expert in dialog systems and the other was a novice.

	KB Results (Restaurant | Food | Area | Pricerange) *nandos* | *portuguese* | *south* | *cheap* *the_lucky_star* | *chinese* | *south* | *cheap*
User-1	hello, i am looking for a *cheap* restaurant on the *south* side of town .
Gold	SELECT * FROM KB WHERE area = south AND pricerange = cheap there is a place called *nandos* that specializes in *portuguese* food on the *south* side of town .
uTOD$^{\text{REINFORCE}}$	SELECT * FROM table WHERE area = i *chiquito_restaurant_bar* is *cheap* in the *south* of town . would you like their phone number
uTOD$^{\text{MAPO}}$	SELECT * FROM table WHERE pricerange = cheap *the_missing_sock* is a *cheap* restaurant in the *south* . would you like their phone number
uTOD$^{\text{m}\mathcal{B}-\text{MAPO}}$	SELECT * FROM table WHERE area = south AND pricerange = cheap *nandos* are *portuguese* in the *south south* would you like their phone number
aTOD$^{\text{SL+RL}}$	SELECT * FROM table WHERE area = south AND pricerange = cheap *the_lucky_star* is a *cheap* restaurant in the *south south* would you like their phone number

Table 8: Queries and responses generated by uTOD$^{\text{m}\mathcal{B}-\text{MAPO}}$ and other TOD systems on a dialog from CamRest. For simplicity, only the fields used in the dialog are mentioned in the KB results. Entities are italicized.

Training		**DSTC2**			**CamRest**		
Q.Position Predictor	**Query Predictor**	**Ent. F1** KB	**Ent. F1** All	**BLEU**	**Ent. F1** KB	**Ent. F1** All	**BLEU**
Predicted	Predicted	0.24±0.02	0.41±0.02	48.35±1.58	0.29±0.03	0.41±0.02	14.68±0.85
Oracle	Predicted	0.32±0.04	0.40±0.02	48.52±1.31	0.32±0.02	0.40±0.02	14.17±0.70
Predicted	Oracle.	0.32±0.03	0.41±0.03	48.94±1.80	0.37±0.04	0.44±0.03	14.63±0.82
Oracle	Oracle	0.38±0.03	0.41±0.02	49.79±1.80	0.39±0.04	0.45±0.03	14.84±0.94

Table 9: Performance gap for supervised TOD systems when trained in different evaluation settings.

Can we train an end-to-end differentiable uTOD system? Our proposed approach for uTOD uses a pipeline of three components trained separately, which can lead to cascading errors. We believe that a single end-to-end neural architecture will likely obtain superior performance. But, this is a technical challenge, since it will require one model to make two discrete decisions: when to query and what to query, complicating the RL problem.

Can we bridge the gap between aTOD performance with and without oracle queries at test time? Our work exposes a critical weakness of fully supervised TOD systems. When evaluated in a setting where the TOD system only has access to its own predicted query and that query's results, the overall performance drops drastically. Table 9 shows the performance of our aTOD system, where at test time, both for query position and the query, the system prediction is used (Prediction) or the gold value is used (Oracle). We observe that using oracle values gets 14-point KB Entity F1 gains over predicted values. Since our setting is very realistic to assess the usability of current dialog systems, this result highlights the research challenge of improving supervised TOD systems in our setting. We also emphasize the importance of KB Entity F1 as an important metric—it better assesses the value offered to the end user, given that these datasets contain mostly informational tasks.

Can we design a uTOD system to handle dialogs that require more than one query per dialog? Our current work makes the assumption of a single query per dialog. This is because most supervised TOD datasets also make only one query per dialog. Given that our task is harder than those, it was important to define it such that existing ML machinery of TOD can feasibly learn our task. At the same time, extending the task definition and creating datasets for the multiple queries per dialog case is straightforward. An important future research challenge will be to design an end-to-end dialog system that can handle a variable number of KB queries in a single dialog, without explicit

query (or query position) annotation. We believe that this is best studied only after substantial progress on our specific task definition.

7 Conclusion

We define the novel problem of learning TOD systems without explicit KB query annotation and the associated subtask of unsupervised prediction of KB queries. We also propose first baseline solutions for these tasks. Our best query prediction baseline extends the existing RL approach MAPO to include multiple query buffers at training time. We also present a pipeline architecture that trains different components in a curriculum to obtain the final TOD system. Our detailed evaluation shows that our approaches achieve much better performance than simpler baselines, though there is some gap when compared to supervised approaches. We study the results further to identify research challenges for future research. We will release all resources for use by the research community.

Acknowledgments

We thank Gaurav Pandey, Danish Contractor, Dhiraj Madan, Sachindra Joshi, and the anonymous reviewers for their comments on an earlier version of this paper. This work is supported by IBM AI Horizons Network grant, an IBM SUR award, grants by Google, Bloomberg, and 1MG, a Visvesvaraya faculty award the Government of India, and the Jai Gupta chair fellowship by IIT Delhi. We thank the IIT Delhi HPC facility for computational resources.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, and et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62. **DOI:** https://doi.org/10.1162/tacl_a_00209

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations*.

Yun-Nung Chen, William Yang Wang, Anatole Gershman, and Alexander I. Rudnicky. 2015. Matrix factorization with knowledge graph propagation for unsupervised spoken language understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 483–494.

Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky. 2013. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 120–125. IEEE. **DOI:** https://doi.org/10.1109/ASRU.2013.6707716

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46. **DOI:** https://doi.org/10.1177/001316446002000104

Pradeep Dasigi, Matt Gardner, Shikhar Murty, Luke Zettlemoyer, and Eduard Hovy. 2019. Iterative search for weakly supervised semantic parsing. In *Proceedings of the 2019 Conference*

of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 2669–2680, Minneapolis, Minnesota. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/N19-1273

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–495. **DOI:** https://doi.org/10.18653/v1/P17-1045

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Dialog System Technology Challenges, Saarbrücken, Germany, August 15-17, 2017*, pages 37–49.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191. **DOI:** https://doi.org/10.18653/v1/D16-1126

Omer Goldman, Veronica Latcinnik, Ehud Nave, Amir Globerson, and Jonathan Berant. 2018. Weakly supervised semantic parsing with abstract examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1809–1819, Melbourne, Australia. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P18-1168

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640. Association for Computational Linguistics.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1051–1062.

Till Haug, Octavian-Eugen Ganea, and Paulina Grnarova. 2018. Neural multi-step reasoning for question answering on semi-structured tables. In *European Conference on Information Retrieval*, pages 611–617. Springer. **DOI:** https://doi.org/10.1007/978-3-319-76941-7_52

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272. **DOI:** https://doi.org/10.3115/v1/W14-4337

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Word-based dialog state tracking with re-current neural networks. In *In Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299. **DOI:** https://doi.org/10.3115/v1/W14-4340

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pages 733–743. Asian Federation of Natural Language Processing.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33. **DOI:** https://doi.org/10.18653/v1/P17-1003

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V. Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *Advances in Neural Information Processing Systems*, pages 9994–10006.

Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, 8(3–4):293–321. **DOI:** https://doi.org/10.1007/BF00992699

A. Madotto, C. S. Wu, and P. Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P18-1136

Dipendra Misra, Ming-Wei Chang, Xiaodong He, and Wen-tau Yih. 2018. Policy shaping and generalized update equations for semantic parsing from denotations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2442–2452. **DOI:** https://doi.org/10.18653/v1/D18-1266

Yatin Nandwani, Abhishek Pathak, Mausam, and Parag Singla. 2019. A primal dual formulation for deep learning with constraints. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 12157–12168.

Arvind Neelakantan, Quoc V. Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2017. Learning a natural language interface with neural programmer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics. **DOI:** https://doi.org/10.3115/1073083.1073135

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics. **DOI:** https://doi.org/10.3115/v1/P15-1142

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–32, Berlin, Germany. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P16-1003

Dinesh Raghu, Nikhil Gupta, and Mausam. 2019. Disentangling language and knowledge in task-oriented dialogs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1239–1255.

Antoine Raux, Brian Langner, Dan Bohus, Alan W. Black, and Maxine Eskénazi. 2005. Let's go public! Taking a spoken dialog system to the real world. In *INTERSPEECH*.

Revanth Gangi Reddy, Danish Contractor, Dinesh Raghu, and Sachindra Joshi. 2019. Multi-level memory for task oriented dialogs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3744–3754. **DOI:** https://doi.org/10.18653/v1/N19-1375

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized experience replay. In *International Conference on Learning Representations*, Puerto Rico.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2440–2448.

Bailin Wang, Ivan Titov, and Mirella Lapata. 2019. Learning semantic parsers from denotations with latent structured alignments and abstract programs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3765–3776. **DOI:** https://doi.org/10.18653/v1/D19-1391, **PMID:** 31933765

T. H. Wen, D. Vandyke, N. Mrkšíc, M. Gašíc, L. M. Rojas-Barahona, P. H. Su, S. Ultes, and S. Young. 2017. 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*, volume 1, pages 438–449. **DOI:** https://doi.org/10.18653/v1/E17-1042, **PMCID:** PMC5677129

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. Conditional generation and snapshot learning in neural dialogue systems. In *EMNLP*, pages 2153–2162, Austin, Texas. ACL.

Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. 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 (Volume 1: Long Papers)*, volume 1, pages 665–677. **DOI:** https://doi.org/10.18653/v1/P17-1062, **PMID:** 28243779

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256. **DOI:** https://doi.org/10.1007/BF00992696

Chien-Sheng Wu, Richard Socher, and Caiming Xiong. 2019. Global-to-local memory pointer networks for task-oriented dialogue. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S. Lasecki, and Dragomir Radev. 2019. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979. **DOI:** https://doi.org/10.18653/v1/D19-1204

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1050–1055. AAAI Press.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pages 658–666, Arlington, Virginia, USA. AUAI Press.