

DKAF: KB Arbitration for Learning Task-Oriented Dialog Systems with Dialog-KB Inconsistencies

Saley Vishal Vivek¹, Rocktim Jyoti Das¹, Dinesh Raghu^{1 2} and Mausam¹

¹ Indian Institute of Technology, New Delhi, India

² IBM Research, New Delhi, India

Vishal.Vivek.Saley@cse.iitd.ac.in, rocktimjyotidas@gmail.com

diraghu1@in.ibm.com, mausam@cse.iitd.ac.in

Abstract

Task-oriented dialog (TOD) agents often ground their responses on external knowledge bases (KBs). These KBs can be dynamic and may be updated frequently. Existing approaches for learning TOD agents assume the KB snapshot contemporary to each individual dialog is available during training. However, in real-world scenarios, only the latest KB snapshot is available during training and as a result, the train dialogs may contain facts conflicting with the latest KB. These dialog-KB inconsistencies in the training data may potentially confuse the TOD agent learning algorithm.

In this work, we define the novel problem of learning a TOD agent with dialog-KB inconsistencies in the training data. We propose a **Dialog-KB Arbitration Framework (DKAF)** which reduces the dialog-KB inconsistencies by predicting the contemporary KB snapshot for each train dialog. These predicted KB snapshots are then used for training downstream TOD agents. As there are no existing datasets with dialog-KB inconsistencies, we systematically introduce inconsistencies in two publicly available dialog datasets. We show that TOD agents trained with *DKAF* perform better than existing baselines on both these datasets.

1 Introduction

A task-oriented dialog (TOD) system often requires information from a knowledge base (KB) to complete user goals like restaurant reservations, flight bookings, and calendar enquiry. This paper follows the recent line of research in *end-to-end* approaches (Wu et al., 2019; Qin et al., 2020; Raghu et al., 2021b), where dialog agents are trained using just the training dialogs and an associated KB, without any expensive dialog state annotation.

The KB contents typically change to reflect the transactions that happened during the user-agent dialogs. For example, in Figure 1, the KB snapshot K_1 can transform into K_2 when *La Margherita*

and *Prezzo* become unavailable due to reservations, and *Bangkok City* becomes available due to a cancellation. Due to this evolving nature of the KB, two dialogs which started with the same user goal can result in two different outcomes. For example, consider the dialogs d_1 and d_2 in Figure 1. In d_1 , the agent makes two recommendations from K_1 , whereas, in d_2 , no recommendation is feasible as K_2 has no restaurants that fit the user’s need.

Existing approaches for learning TOD agents assume the KB snapshot contemporary to each dialog is available during training. Such an assumption is limiting due to two reasons. First, KB snapshots are usually created at periodic intervals not after each KB transaction due to storage constraints. Second, dialogs used for training TOD models are often collected from messaging applications where human agents and users interact. Human agents often access the associated KB using a different application and so the KB queries fired during the dialog do not get logged with the dialogs (Raghu et al., 2021a). Without these KB query logs, it is difficult to reconstruct the contemporary KB.

As the contemporary KB snapshots are unavailable, a single KB snapshot (generally, the latest) is made available during training. When the latest KB snapshot gets associated with the train dialogs, the dialogs and the KB may portray diverging information resulting in *dialog-KB inconsistencies*. In the running example, K_T denotes the latest KB snapshot. Dialog d_1 disagrees with K_T , as *La Margherita* is missing from K_T . Dialog d_2 also disagrees with K_T , since K_T contains an Italian restaurant, contradicting agent response.

Dialog-KB inconsistencies hinder the learning of TOD agents. These inconsistencies can force the TOD agent to either learn spurious patterns (e.g., using d_2 and K_T may force the agent to ignore *Prezzo*) or memorizes responses (using d_1 and K_T , will force the agent to generate *La Margherita*) leading to poor generalization. To overcome these

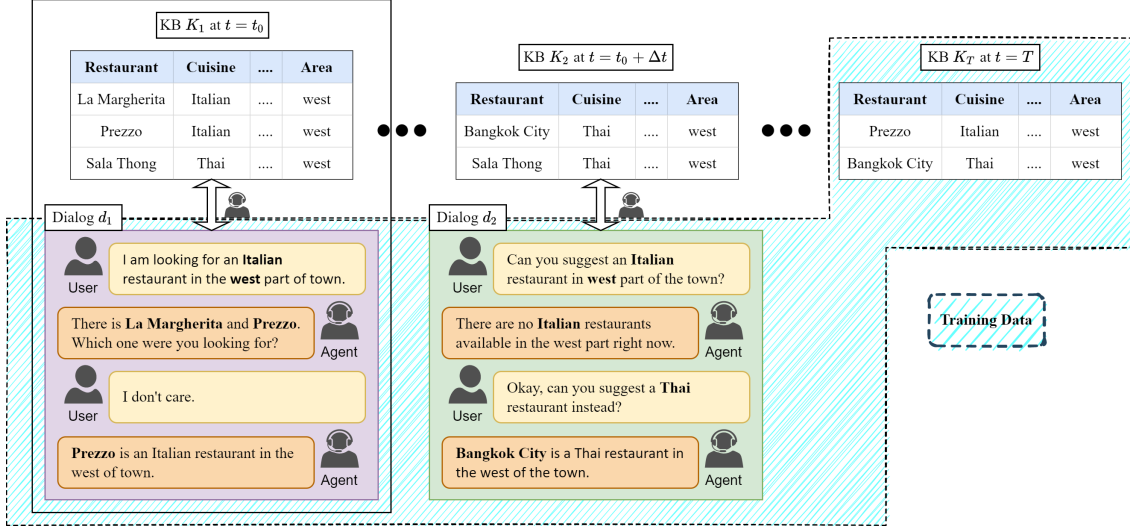


Figure 1: Figure shows snapshots of an evolving KB at times t_0 , $t_0 + \Delta t$ and T . Over time, restaurants in the KB is changing, which is reflected in the KB snapshots K_1 and K_2 at time t_0 and $t_0 + \Delta t$ respectively. Dialogs d_1 and d_2 are consistent with KB snapshots K_1 and K_2 . During training, KB snapshot K_T is associated with dialogs d_1 and d_2 resulting in dialog KB inconsistencies. Shaded region defines our problem setting.

challenges, we define the novel problem of end-to-end learning of TOD systems with dialog-KB inconsistencies in training data. We also propose *DKAF*, whose goal is to reduce the dialog-KB inconsistencies by predicting the contemporary KB for each dialog in the training corpus. These predicted KB snapshots and the associated dialogs can then be used to train any existing end-to-end TOD learning approaches.

Given a dialog, inconsistencies can be removed by inserting a new row in the KB based on the entities and relationships present in the dialog (e.g., adding *La Margherita* to K_T can make d_1 consistent with K_T). Inconsistencies can also be removed by deleting rows (e.g., removing *Prezzo* from K_T can make d_2 consistent). As dialogs offer *weak supervision* to reduce dialog-KB inconsistencies, we use distant supervision and reinforcement learning to train *DKAF*.

We construct two datasets by systematically infusing dialog-KB inconsistencies on bAbI (Bordes and Weston, 2017), and BiTOD (English) (Lin et al., 2021) datasets and refer to them as *inc-bAbI* and *inc-BiTOD* respectively. Our experiments show that *DKAF* reduces the dialog-KB inconsistencies and the overall TOD system trained with the KB predicted by *DKAF* outperforms existing state-of-the-art models on both the datasets. In summary,

1. We introduce the novel problem of training task-oriented dialog systems over data with

dialog-KB inconsistencies.

2. We present *DKAF* that alleviates dialog-KB inconsistencies by predicting the contemporary KB based on a given training dialog.
3. We systematically modify two publicly available datasets for the proposed task. Our experiments demonstrate that *DKAF* improves TOD performance on these datasets.

We release all resources for future research¹.

2 Related Work

Traditionally, dialog systems are modular (Young et al., 2013; Rojas-Barahona et al., 2016; Hosseini-Asl et al., 2020) with different modules for natural language understanding, dialog state tracking, and natural language generation. These models require hand-crafting of dialog states and require expensive intermediate annotations for training each component. On the other hand, end-to-end TOD models (Eric et al., 2017; Madotto et al., 2018; Raghu et al., 2021b, 2019; Wu et al., 2019) that directly predict system response given dialog history and the KB are becoming increasingly popular as they alleviate the need for expensive annotations. *DKAF* approach proposed in this work focuses on learning end-to-end TOD system when training data has dialog-KB inconsistencies.

Recent works on inconsistency in dialog generation by Nie et al. (2021); Qin et al. (2021, 2020)

¹<https://github.com/dair-iitd/DKAF>

study problem of detecting inconsistent dialog responses with respect to dialog history, user intent, the KB. Welleck et al. (2019) explores a similar problem but in domain of Persona-based dialog systems. Larson et al. (2020) studies the topology of annotation inconsistencies in crowd-sourced data for slot-filling models.

DKAF differs from these works in two key ways: (1) its objective is learning a TOD model when training data includes dialogs inconsistent with the KB and, (2) it explicitly resolves dialog-KB inconsistencies via a novel KB arbitration procedure.

3 Problem Definition

We first describe the task of learning an end-to-end TOD system. We denote a dialog between user u and agent a as $d = [u_1^u, u_1^a, u_2^u, u_2^a, \dots, u_m^u, u_m^a]$ where m denotes number of exchanges. Let $\{d_j\}_{j=1}^N$ be the set of N training dialogs. An end-to-end TOD system predicts agent response \hat{u}_i^a given dialog history $[u_1^u, u_1^a, u_2^u, u_2^a, \dots, u_i^u]$ and an associated KB K_T . This system is trained using $\{d_j, K_T\}_{j=1}^N$ where K_T is assumed to be consistent with all the training dialogs.

We now consider the setting where training dialogs are grounded in an evolving KB. Here, a training dialog d_j is consistent with its contemporary KB snapshot, K_j . However, at training time, a single KB snapshot K_T is available which gets associated with all training dialogs resulting in dialog-KB inconsistencies. So, we propose the task of learning end-to-end TOD system using $\{d_j, K_T\}_{j=1}^N$ with dialog-KB inconsistencies.

4 DKAF

To solve dialog-KB inconsistencies, we propose *DKAF* that updates K_T based on d_j such that the resultant KB snapshot \hat{K}_j resembles with K_j . A TOD system is then trained using $\{d_j, \hat{K}_j\}_{j=1}^N$. *DKAF*'s updates to K_T happen through a cascade of three models - row insertion, row deletion, and row completion. Each model takes the KBs resulting from the preceding model and performs modifications to them based on the training dialogs. Figure 2 highlights this process. We now describe each model in detail.

4.1 Row Insertion (RI)

Row insertion aims to extract rows from the dialogs that are missing from the training KB. For this, RI model predicts if a relation r holds between entities

e_1 and e_2 mentioned in a given dialog d . Following Zhang and Wang (2015), it infuses d with position indicators for e_1 and e_2 and encodes the resulting dialog using a hierarchical encoder (Sordani et al., 2015). Encoder feature vectors for a dialog and entities are then passed through classifier network for relation r . Thus, RI model uses training dialog to identify missing KB relationships (e_1, r, e_2) . Figure 2 showcases this where $(Bangkok\ City, cuisine, Thai)$ and $(Bangkok\ City, area, west)$ get added to the KB. We provide more details in B.2.

We form supervised data for training RI model with distant supervision and follow annotation scheme of Xu et al. (2013). Given a training dialog d , we form three sets - positive, negative and infer consisting of type-consistent relationships. For entities $e_1, e_2 \in d^2$, a relationship (e_1, r, e_2) is in positive set if it also exists in K_T . A relationship (e_1, r, e_2) is in negative set when its head entity e_1 exists in K_T but the relationship does not. We follow this conservative annotation to avoid to false negatives samples. We add all remaining relationships to infer set. We train RI model over the union of positive and negative sets from all training dialogs.

We apply RI model over infer set from training dialog d_j to obtain KB snapshot K_j^{ri} post insertion.

We note that (Yu et al., 2020) proposed a similar task of predicting relations among the individuals engaged and mentioned in dialogs from a popular TV series. However their approach is fully supervised while we use distant supervision.

4.2 Row Deletion (RD)

RD model predicts whether a row ρ from KB K (mis)aligns with a given dialog d . Here, ρ is misaligned if it disrupts agent reasoning in d . In figure 2, row for *Na Thai* is misaligned with d_j since it forces the TOD system to generate a factually incorrect response "*Sorry it is not available...*". Further, it hinders TOD system from producing *Sala Thong* as it is rated below *Na Thai*. We use RD model predictions to drop misaligned rows from the KB.

For input d , RD model computes dialog features using the dialog encoder given in Section 4.1. Recent works (Banerjee and Khapra, 2019; Yang et al., 2020) showcase the efficacy of GCNs in TOD modeling. Consequently, RD model includes an r-GCN (Schlichtkrull et al., 2018) KB encoder that com-

²can be identified by NER, though in this work, we assume this is known

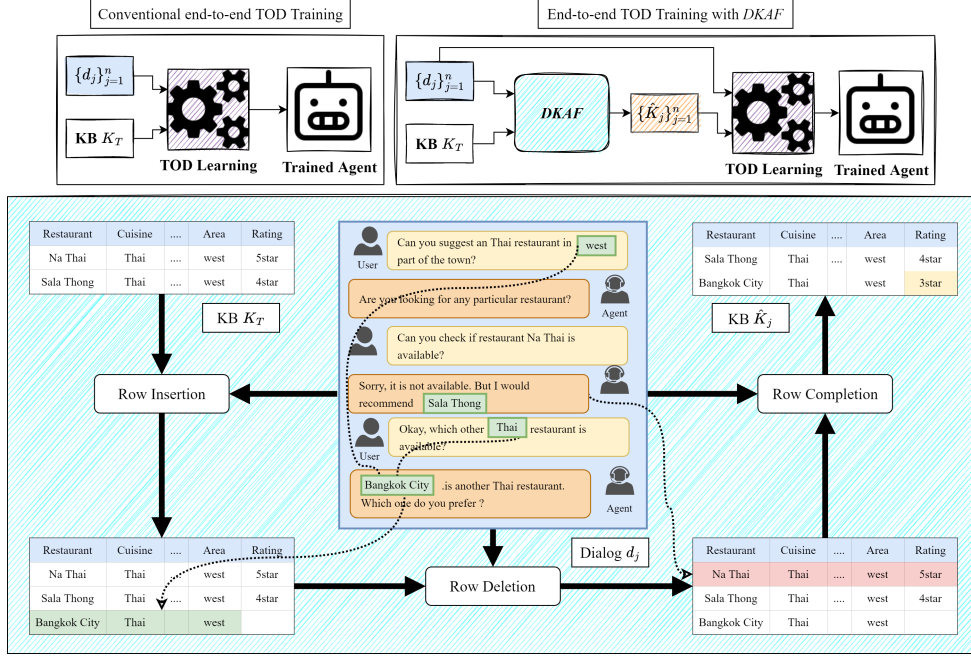


Figure 2: Comparison of conventional TOD learning (top-left) with TOD learning with *DKAF* (top-right). *DKAF* attempts to resolve dialog-KB inconsistencies by updating training KB K_T given a training dialog. Figure (bottom) shows *DKAF* in action with KB updates from row insertion, row deletion and row completion to training KB K_T .

putes KB entity features. Then, RD model reasons over KB entities using a memory network (Sukhbaatar et al., 2015) with dialog features as query input. Finally, it appends memory network output with features of a row (sum of constituent entity features). The resulting vector is fed to a feed-forward network that makes binary prediction. We provide further information in B.2

Training RD Model

We adopt reinforcement learning (RL) to train RD model due to lack of supervised dataset. We treat RD model as an RL agent that inputs a state (d, K, ρ) and takes action $a \in \{0, 1\}$ where $a = 0$ means ρ is misaligned with d . Given reward function $R_a(d, K, \rho)$, RL objective for training RD is

$$J_{RD} = \sum_{j=i}^N \frac{1}{|K_j^{ri}|} \sum_{\rho \in K_j^{ri}} R_a(d_j, K_j^{ri}, \rho)$$

We posit that a TOD system can provide an appropriate reward function for the task. In our running example, dropping *Na Thai* from the KB aids agent reasoning in the dialog causing likelihood of *Sala Thong* in the agent utterance to improve. Thus, likelihood score from a TOD system can guide RD tasks. We incorporate this insight using a novel masked entity modeling (MEM) task. Let e be an entity in the i^{th} utterance in given dialog d . We

form a masked dialog history H_e consisting of utterances till i^{th} utterance and replace entity e in i^{th} utterance with a $\langle mask \rangle$ token. Let E_a be the set of entities occurring in agent utterances d . MEM objective is then to maximize following likelihood

$$\mathcal{L}(d, K) = \prod_{e \in E_a} P(e|H_e, K) \quad (1)$$

Now we derive reward function for RD model as

$$R_0(d, K, \rho) = \text{sgn}[\mathcal{L}(d, K \setminus \{\rho\}) - \mathcal{L}(d, K)]$$

$$R_1(d, K, \rho) = -R_0(d, K, \rho)$$

Note that, deleting a conflicting row improves the likelihood in equation 1 thus incurs a positive reward otherwise a negative reward.

Inspired by recent works (Wu et al., 2019; Raghu et al., 2021b; He et al., 2020b), we design our MEM model as a dual pointer network where $P(e|H_e, K)$ is modelled as probability of copying masked entity e from H_e tokens and KB entities. We discuss MEM model in detail in appendix B.2.

We train both MEM and RD models using $\{d_j, K_j^{ri}\}_{j=1}^N$. We train RD using MAPO algorithm (Liang et al., 2018), since our action space is discrete and state transitions deterministic. We use predictions from RD model over (d_j, K_j^{ri}, ρ) states from each d_j to obtain snapshot K_j^{rd} post deletion.

4.3 Row Completion (RC)

RI model adds new rows to the KB, which can be incomplete since fields like rating of restaurants need not occur explicitly in the dialog. Yet, these fields can be crucial for TOD system. Rating can be necessary, for example, when agent selects the restaurant from the KB based on its rating. We call fields like rating latent fields and RC model aims to deduce the values for such fields from the dialog. For example in figure 2, RI should predict a rating *3star* or lower for *Bangkok City*.

We consider entity e_s in dialog d such that e_s is not related to any entity in KB K via latent field type r . RC model aims to predict target entity for the partial relationship (e_s, r) given d . It infuses d with position indicators for e_s and encodes resulting dialog using dialog encoder. Similar to 4.2, it computes KB entity features using KB encoder and reasons over them using memory network. Finally, it appends memory network output with e_s encoding and feeds it to a feed-forward network that predicts the target entity $e_t \in E_r$. Here, E_r is the set of valid target entities for r based on the task ontology. We provide more details in B.2

Similar to 4.2, we treat RC model as RL agent that observes state (d, e_s, r, K) and takes an action $e_t \in E_r$. We use following reward function to train the model

$$R_{e_t}(d, e_s, r, K) = \begin{cases} 1 & \text{if } e_t = \arg \max_{e \in E_r} \mathcal{L}(d, K \cup \{e_s, r, e\}) \\ 0 & \text{otherwise} \end{cases}$$

For training dialog d_j , we create state space $\{(d_j, e_s, r, \tilde{K}_j^{rd})\}$ where entity $e_s \in d_j$, r is a latent field and \tilde{K}_j^{rd} is formed by dropping any relationships (e_s, r, e) from K_j^{rd} . We train RC model using MAPO over state-spaces combined over training dialogs. Finally, the trained RC model makes prediction over incomplete rows in K_j^{rd} to get final snapshot \hat{K}_j .

5 Experimental Setup

5.1 Datasets Construction

Existing TOD datasets make a simplistic assumption that KB contents do not change over time. Hence, all dialogs in these datasets are consistent with the KB. To study our problem, we systematically induce dialog-KB inconsistencies in two existing TOD datasets, namely bAbI dialog (Bordes and Weston, 2017) & BiTOD (English) (Lin

et al., 2021) and refer to them as *inc-bAbI* and *inc-BiTOD*, respectively. bAbI dialog dataset consists of synthetically generated dialogs from the restaurant reservation domain. BiTOD is a human-generated multi-domain dialog dataset with dialogs in English and Chinese. For our experiments, we only use the English dialogs from hotel, restaurant, and attraction domains. For more details on these datasets please refer to Appendix A.

We follow a two-step procedure to simulate the dialog-KB inconsistencies. In the first step, we generate an evolving KB by modifying its contents over time and maintaining a snapshot with timestamp associated with it. To generate an evolving KB, we add a binary random variable, named *available*, to indicate the availability of each KB entry as illustrated in Figure 3.

For restaurants, we wanted our simulator to reflect real-life scenarios where restaurants are often available during afternoons but are busy during peak hours (like evening and breakfast). To this end, we use the Yelp dataset³. Yelp provides the number of customers that have checked in into a restaurant at any given hour of the day for any day of the week. We use this data to simulate the availability of restaurants in our KB. Given the time of the day and day of the week, we sample restaurant availability to be inversely proportional to the number of check-ins from Yelp data. In our simulation, we also mimic (a) maintenance breaks by making restaurants unavailable for a day with a probability of 0.05 and (b) permanent closures with a probability of 1e-5.

Unfortunately, for hotels we did not find any check-ins data. we set the availability of each KB entry following a Bernoulli distribution parameterized by a success probability p set to 0.75. Contrary to restaurants and hotels, attractions are generally available. Thus, we do not simulate their availability. Note that as entities are simulated differently, our dataset has a mixture of different evolving KB patterns.

In the second step, we assign a timestamp to each dialog and associate it with a corresponding KB snapshot. For example, the dialog d_j in Figure 3 is associated with the snapshot K_j . We then identify the KB entities present in the dialog (e.g., *Sala Thong* and *3 star* in d_j) and replace them with appropriate entities from the snapshot K_j that match the annotated dialog state (e.g., *cui-*

³<https://www.yelp.com/dataset>

sine=Thai, area=east). All modified dialogs and the last snapshot of the KB together form the inconsistent version of the dataset. Each modified dialog d_j will be consistent with its KB snapshot K_j but may not be consistent with the last snapshot used for training. To mimic real-world settings, we only induce inconsistencies in the train dialogs. The test dialogs remain consistent.

5.2 Algorithms

We compare our proposed approach against the following baselines: GLMP (Wu et al., 2019), CDNet (Raghu et al., 2019) and SimpleTOD (Hosseini-Asl et al., 2020). GLMP and CDNet are both end-to-end TOD models. SimpleTOD is GPT2 based model that requires belief state annotations. So, we adapt SimpleTOD to the end-to-end TOD setting. For more details please refer to Appendix D.1.

We train the baselines on *inc-bAbI* and *inc-BiTOD* datasets and identify the best-performing baseline. The best baseline is then trained in the following two settings:

Rule-based: A rule-based system performs KB arbitration for each dialog. Resulting KB snapshots are then used to train the TOD model. We defer the discussion of the rules in Appendix C.

DKAF: This is our proposed approach that performs KB arbitration for each dialog d_j with *DKAF*. The predicted KB snapshot and dialog $\{d_j, \hat{K}_j\}_{j=1}^N$ pairs to train the TOD model.

The training details are reported in Appendix D.

5.3 Evaluation Metrics

As *inc-bAbI* is synthetically generated, following Bordes and Weston (2017), we use exact string matching metrics: response accuracy (percentage of predicted responses that exactly match the gold response) and dialog accuracy (percentage of dialogs with all correctly predicted responses).

As *inc-BiTOD* is human-generated, we follow Wu et al. (2019) and use BLEU (Papineni et al., 2002) and Entity F1 (Eric et al., 2017) for measuring response prediction performance. Dialog-KB inconsistencies can cause models to learn incorrect KB reasoning patterns. To measure this effect, we also report KB Entity F1 from Raghu et al. (2021a) computed for entities that can only be inferred from KB. We also perform human evaluation for *inc-BiTOD* along two dimensions: (i) *Relevance*: how useful are the responses given the dialog and KB, and (ii) *Naturalness*: how human-like are the pre-

dicted responses. Each dimension is annotated on a Likert scale of 0-4 (Likert, 1932a).

6 Results

We answer the following research questions in our experiments:

1. *Performance Study*: How effective is *DKAF* in fixing the dialog-KB inconsistencies?
2. *Ablation Study*: What is the performance gain from each component of *DKAF*?
3. *Incremental Analysis*: How robust is *DKAF* to the number of inconsistent dialogs in the train data?

6.1 Performance Analysis

Table 1 reports the response prediction performance on *inc-bAbI* and *inc-BiTOD* datasets. We first discuss the performance of baseline models. We then integrate *DKAF* into the best-performing model - SimpleTOD and discuss how well *DKAF* mitigates the effect of dialog-KB inconsistencies.

Baseline Performance: We observe that dialog-KB inconsistencies affect baseline models in varying degrees. On *inc-bAbI* dataset, SimpleTOD achieves the best performance with 90.6% dialog accuracy. Whereas, GLMP and CDNet perform poorly with dialog accuracy of 73.6% and 66.8%.

SimpleTOD also achieves the best performance on *inc-BiTOD* dataset across all the metrics. This is expected, especially in the human-generated *inc-BiTOD* dataset, as SimpleTOD is built on top of GPT2. We select SimpleTOD for our further experiments with *DKAF*.

Efficacy of *DKAF*: We report the performance of SimpleTOD + Rule-based and SimpleTOD + *DKAF* in table 1. In *inc-bAbI* dataset, SimpleTOD + *DKAF* shows improvement over SimpleTOD model with 8.6% gain in dialog accuracy. SimpleTOD is also the best-performing model across all baselines. To analyze the results of *DKAF*, we compare the number of dialog-KB inconsistencies in *inc-bAbI* before and after *DKAF* arbitration. *DKAF* performs total of 239 insertions and 207 deletion in *inc-bAbI* causing inconsistencies to drop from 35.8% to 2.8% validating effectiveness of *DKAF* in resolving the inconsistencies.

SimpleTOD + Rule-based, on contrary, performs worse even compared to SimpleTOD baseline. Rule-based arbitration performs 239 insertions and 1014 deletions to *inc-bAbI* reducing the inconsistency rate to 0%. Yet, this does not re-

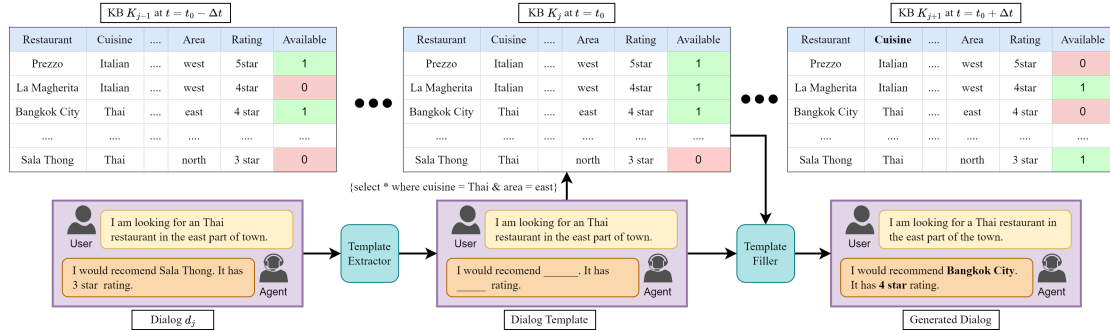


Figure 3: Figure shows the simulation pipeline used for generating datasets.

Model	<i>inc-bAbI</i>		<i>inc-BiTOD</i>		
	Dialog Acc.	Response Acc.	BLEU	Ent. F1	KB Ent. F1
GLMP	73.6	97.87	15.29	0.674	0.633
CDNet	66.8	96.76	19.37	0.772	0.745
SimpleTOD	90.6	99.39	20.28	0.786	0.757
SimpleTOD + Rule-based	53.1	96.28	21	0.761	0.773
SimpleTOD + DKAF	99.2	99.94	24.91	0.819	0.833

Table 1: Performance of GLMP, CDNet and SimpleTOD on *inc-bAbI* and *inc-BiTOD* dataset. We report SimpleTOD in Rule-based and *DKAF* setting.

	Relevance	Naturalness
SimpleTOD	3.15	3.71
SimpleTOD + Rule-based	3.05	3.84
SimpleTOD + DKAF	3.36	3.74

Table 2: Human Evaluation on *inc-BiTOD*

sult in performance improvement over baselines. Here, excessive deletions due to rule-based arbitration upset reasoning patterns in the dataset more than dialog-KB inconsistencies. Note that domain experts can improve such rule-based system further by incorporating reasoning patterns peculiar to the domain. On other hand, *DKAF* makes achieves gains in performance with minimal domain-specific assumptions.

For *inc-BiTOD* dataset, SimpleTOD + *DKAF* outperforms SimpleTOD model in entity F1 and entity F1 KB metrics by a margin of 3.25 and 7.64 points. The gain in entity F1 KB is indicative of *DKAF*'s effectiveness in resolving inconsistencies. In total, *DKAF* makes 264 insertions and 207 deletions to *inc-BiTOD* which results in dialog-KB inconsistencies to drop from 23% to 6.94%. We find that resolving dialog-KB inconsistencies is much more challenging in human-generated dataset. As in *inc-bAbI*, SimpleTOD + Rule-based under-performs compared to SimpleTOD baseline in *inc-BiTOD* as well. Rule-based

arbitration results in 5.08% inconsistencies from 264 insertions and 2889 deletions.

Human Evaluation: We summarize the human evaluation results on the *inc-BiTOD* dataset in Table 2. We randomly sample 50 (dialog-context, response) pairs from *inc-BiTOD* and two human judges labelled responses generated by SimpleTOD, SimpleTOD + Rule-based and SimpleTOD + *DKAF* on relevance and grammar on a Likert scale (0-4) (Likert, 1932b). We observe that on relevance, SimpleTOD + *DKAF* outperforms both SimpleTOD (0.21) and SimpleTOD + Rule-based (0.31) baselines.

However, naturalness score of SimpleTOD + Rule-based is better than SimpleTOD and SimpleTOD + *DKAF*. Upon investigation, we found that the annotator favoured SimpleTOD+Rule-based due to minor grammatical errors. For example, the annotator preferred SimpleTOD+Rule-based because it used the preposition "from" instead of "on" before april 24 as shown below:

1. *SimpleTOD + Rule-based*: so you would like to book 4 rooms at mingdu hotel for 4 nights starting from april 24 ?
2. *SimpleTOD + DKAF*: so you would like to book 4 rooms at mingdu hotel for 4 nights starting on april 24 ?

We provide more details on human evaluation in Appendix H.

6.2 Ablation Experiments

Model	<i>inc-bAbI</i>	<i>inc-bAbI(M)</i>	<i>inc-BiTOD</i>
	Dlg Acc.	Dlg Acc.	KB Ent. F1
SimpleTOD	90.6	49.7	0.757
+ DKAF w/o RI	91.9	62.3	0.749
+ DKAF w/o RD	98	77.7	0.793
+ DKAF w/o RC	99	79.9	0.833
+ DKAF	99.1	88.6	0.833

Table 3: Ablation Results

We perform ablation for each component in *DKAF* to measure how each stage contributes to overall *DKAF* performance. Table 3 reports our results.

For both *inc-bAbI* and *inc-BiTOD*, excluding RI leads to a significant performance drop. In the case of *inc-BiTOD*, we observe that excluding RI also causes RD model to abstain from removing rows from the KB. Dropping RD results in performance drop of 1.1 points for *inc-bAbI* dataset and 0.04 for *inc-BiTOD*. This is expected as agent suggestions in both *inc-bAbI*, and *inc-BiTOD* follow rating orders, and row deletion restores this order by systematically deleting upsetting rows. This can be seen in examples given in table 15 and 17. We provide further details on why dropping RI leads to severe degradation in comparison to RD and RC in section 6.4.

Finally, excluding RC has a lower impact in *inc-bAbI*. In *inc-bAbI*, restaurant names carry much of its attributes include its rating. We posit that SimpleTOD tokenization allows model a direct access to this rating. For example, SimpleTOD tokenizer splits restaurant name *resto_rome_cheap_thai_2stars* in *inc-bAbI* into attributes (*rest, o,_, rome, _, che, ap, _, th, ai, _, 2, stars*). As a result, SimpleTOD can operate sufficiently well even in absence of the ratings.

To validate this, we modify *inc-bAbI* dataset where we replace the rating in restaurant names with random alphabets. For example, we replace *resto_rome_cheap_thai_2stars* with *resto_rome_cheap_thai_Qstars*. We report ablations on resulting dataset, named *inc-bAbI(M)*, in table 3. SimpleTOD performance significantly deteriorates in *inc-bAbI(M)* with a drop as high as 40.9 points compared to *inc-bAbI*. Note that *DKAF* improved performance by a margin of 38.9 points. Here, we observe that excluding RC leads to 8.7 point drop. On the other hand, *inc-BiTOD* does not have any such latent entities in the KB, thus

resulting in no change in performance.

6.3 Incremental Analysis

We create 5 variants *inc-bAbI* dataset with increasing inconsistency rates in our simulation. For each dataset variant, we train SimpleTOD and SimpleTOD + *DKAF* model. Figure 4 showcases the results. With an increasing number of dialog-KB inconsistencies, the performance of SimpleTOD model decreases sharply. On the other hand, SimpleTOD + *DKAF* is consistently able to recover from the performance drop with significant gains.

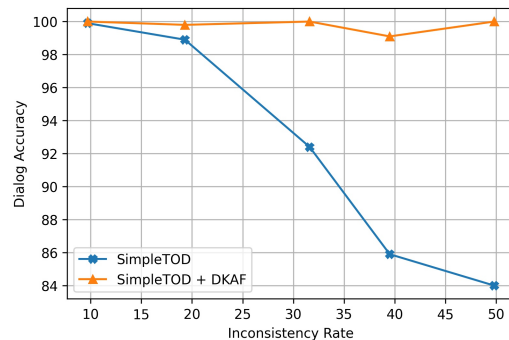


Figure 4: DKAF Incremental Analysis on *inc-bAbI*

6.4 Order of models in *DKAF*

In this section, we validate our choice of order among the different models in *DKAF*. As discussed in section 4.3, RC acts on the new rows introduced by RI, so RC will always follow RI. Consequently, (RI, RD, RC), (RI, RC, RD) and (RD, RI, RC) are the only possible permutations. We note the following observations regarding *DKAF*.

- *Row insertion assists the performance of row deletion and row completion.* Our reward functions are based on MEM likelihood of the entities occurring in the dialog (eq. 1). When an entity (say a restaurant) in a dialog is missing from the KB, eq 1 yields a very low likelihood value. Consequently, training of RD and RC is adversely affected as reward functions become uninformative on such dialogs. By ensuring that training dialogs do not contain entities missing from the training KB, RI assists the training of RD and RC.
- *RD assists training of RC.* Among row deletion and completion, RL training of RC is challenging due to larger action space. We thus run RD first to remove rows from the KB

Permutation	Dialog Acc.	Response Acc.
(RI, RD, RC)	88.6	99.70
(RD, RI, RC)	83.6	99.01
(RI, RC, RD)	86.8	99.17

Table 4: Different orderings of models in *DKAF*.

that disrupts the reasoning in the dialogs. This further helps RC during training.

We experiment with the these three orderings on *inc-bAbI(M)* dataset and report the results in table 4. (RI, RD, RC) outperforms the other two permutations as expected. We note that dropping RI leads training dialogs to contain entities missing from the KB. Further, it adversely affects the training of other *DKAF* models. Similarly, dropping RD leaves training KB with rows that upset dialog reasoning patterns and also disrupt RC training. Finally, dropping RC does not influence the preceding models. As a result, we expect dropping RI should cause a higher drop in performance followed by RD and RC as discussed in section 6.2.

6.5 *DKAF* Model Evaluations

We evaluate RI, RD, and RC models for their corresponding tasks. Table 5 summarizes our findings. For a given dialog d , we identify set R of rows by comparing training KB \mathcal{K}_T with contemporary KB \mathcal{K}_d for the dialog. We then use R to compute F1 for RI. We observe that RI model performs reasonably well in both *inc-bAbI* and *inc-BiTOD* datasets though we observe a performance drop in case *inc-BiTOD*. This is expected as *inc-BiTOD* is human-generated and provides a more challenging setting.

For RD model, we obtain set D_g of rows that occur in \mathcal{K}_T but are missing from \mathcal{K}_d . We compare rows D_p deleted by RD with D_g to compute row deletion F1. We find that performance of RD model is comparatively poor on both the datasets. RD task is difficult compared to RI due to lack of supervision. Further, RD requires understanding of complex reasoning patterns in the datasets. Our RL-based approach alleviates these challenges though there still remains margin for improvement. Nonetheless, we obtain significant performance gains with RD as discussed in 6.2.

We evaluate RC model on *inc-bAbI* dataset. In this case, we consider a prediction by the model to be correct if the predicted rating fits into the rating

Dataset	RI F1	RD F1	RC Acc
<i>incbAbI</i>	1.0 (1.0)	0.451	0.795
<i>incBiTOD</i>	0.708 (0.96)	0.398	

Table 5: *DKAF* model evaluation. F1 scores for relationship extraction are given in brackets.

order in the KB. We then report accuracy across all predictions of the RC model.

7 Conclusions

We define the novel task of end-to-end training of task-oriented dialog agents, when training data may have inconsistencies between dialog and accompanying KB. This scenario arises, when KB evolves over time, but only one final KB is attached with the data, instead of saving KB snapshots associated with each training dialog. We also contribute two datasets, curated by systematically modifying *bAbI* and *BiTOD* datasets, for our task.

Existing state-of-the-art TOD models, when trained on our datasets, can get quite confused. Our proposed solution, *DKAF*, hypothesizes corrections to KB for each dialog so that the KB becomes dialog-consistent. Since no explicit annotation is available, the modules for KB correction are trained via distant supervision and reinforcement learning. When trained on such corrected data, *DKAF*-based TOD models outperform vanilla TOD models in almost all settings. We release our code and data for further research on the topic.

Acknowledgements

This work is supported by IBM AI Horizons Network grant, grants by Google, Verisk, and IMG, an IBM SUR award, and the Jai Gupta chair fellowship by IIT Delhi. Vishal is supported by a Google Fellowship. We also thank the IIT Delhi HPC facility for its computational resources.

Limitations

DKAF model has only been tested on English data so far. At the moment, we curate new datasets by systematic modification of existing datasets. Our simulation strategy is limited as it does not capture real-world factors (e.g. COVID-19 pandemic) that have a drastic impact on restaurant availability. Finally, It would be interesting to find a real-world dataset and verify whether the proposed methods give similar performance gains on it or not.

References

- Suman Banerjee and Mitesh M. Khapra. 2019. Graph convolutional network with sequential attention for goal-oriented dialogue systems. *Transactions of the Association for Computational Linguistics*, 7:485–500.
- Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *ArXiv*, abs/1605.07683.
- Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.
- Mihail Eric, Lakshmi. Krishnan, François Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. *ArXiv*, abs/1705.05414.
- Zhenhao He, Yuhong He, Qingyao Wu, and Jian Chen. 2020a. Fg2seq: Effectively encoding knowledge for end-to-end task-oriented dialog. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8029–8033.
- Zhenhao He, Jiachun Wang, and Jian Chen. 2020b. Task-oriented dialog generation with enhanced entity representation. In *INTERSPEECH*.
- Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *ArXiv*, abs/2005.00796.
- Stefan Larson, Adrian Cheung, Anish Mahendran, Kevin Leach, and Jonathan K. Kummerfeld. 2020. Inconsistencies in crowdsourced slot-filling annotations: A typology and identification methods. In *COLING*.
- Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V. Le, and N. Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *NeurIPS*.
- Rensis Likert. 1932a. A technique for the measurement of attitude scales.
- Rensis Likert. 1932b. A technique for the measurement of attitudes. *Archives of psychology*.
- Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, Peng Xu, Feijun Jiang, Yuxiang Hu, Chen Shi, and Pascale Fung. 2021. Bitod: A bilingual multi-domain dataset for task-oriented dialogue modeling. *ArXiv*, abs/2106.02787.
- Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.
- Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *ACL*.
- Yixin Nie, Mary Williamson, Mohit Bansal, Douwe Kiela, and Jason Weston. 2021. I like fish, especially dolphins: Addressing contradictions in dialogue modeling. *ArXiv*, abs/2012.13391.
- 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 of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.
- Libo Qin, Tianbao Xie, Shijue Huang, Qiguang Chen, Xiao Xu, and Wanxiang Che. 2021. Don’t be contradicted with anything! ci-tod: Towards benchmarking consistency for task-oriented dialogue system. *ArXiv*, abs/2109.11292.
- Libo Qin, Xiao Xu, Wanxiang Che, Yue Zhang, and Ting Liu. 2020. Dynamic fusion network for multi-domain end-to-end task-oriented dialog. In *ACL*.
- 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 1239–1255. Association for Computational Linguistics.
- Dinesh Raghu, Nikhil Gupta, and Mausam. 2021a. Un-supervised learning of kb queries in task-oriented dialogs. *Transactions of the Association for Computational Linguistics*, 9:374–390.
- Dinesh Raghu, Atishya Jain, Mausam, and Sachindra Joshi. 2021b. Constraint based knowledge base distillation in end-to-end task oriented dialogs. In *FINDINGS*.
- Lina Maria Rojas-Barahona, Milica Gašić, Nikola Mrkšić, Pei hao Su, Stefan Ultes, Tsung-Hsien Wen, Steve J. Young, and David Vandryke. 2016. A network-based end-to-end trainable task-oriented dialogue system. In *Conference of the European Chapter of the Association for Computational Linguistics*.
- M. Schlichtkrull, Thomas Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. *ArXiv*, abs/1703.06103.
- Alessandro Sordani, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jianyun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*.
- Sainbayar Sukhbaatar, Arthur D. Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*.

Sean Welleck, Jason Weston, Arthur D. Szlam, and Kyunghyun Cho. 2019. Dialogue natural language inference. In *ACL*.

Chien-Sheng Wu, Richard Socher, and Caiming Xiong. 2019. Global-to-local memory pointer networks for task-oriented dialogue. *ArXiv*, abs/1901.04713.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *ACL*.

Shiquan Yang, Rui Zhang, and Sarah Monazam Erfani. 2020. Graphdialog: Integrating graph knowledge into end-to-end task-oriented dialogue systems. In *EMNLP*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*.

Steve J. Young, Milica Gasic, Blaise Thomson, and J. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101:1160–1179.

Dian Yu, Kai Sun, Claire Cardie, and Dong Yu. 2020. Dialogue-based relation extraction. *ArXiv*, abs/2004.08056.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *ArXiv*, abs/1508.01006.

A Dataset Details

Here we provide details for *inc-bAbI* and *inc-BiTOD* datasets. Table 6 shows the train, validation and test splits of the *inc-BiTOD* and *inc-bAbI*.

inc-bAbI consists of dialogs from restaurant domain where queries the agent for restaurants fitting user constraints. Agent gathers all user constraints and suggests fitting restaurants in descending order. User can further request for address or phone number for the restaurant of their choosing. The restaurant knowledge base consists of 1200 entries where each entry has 8 associated attributes. *inc-bAbI* dataset has with 35.8% inconsistent dialogs.

inc-BiTOD is a multi-domain dataset containing dialogs from hotel, restaurant and attraction domains. In *inc-BiTOD*, the agent suggests user (hotel, restaurant or attraction) based on user-provided constraints. There are 699 hotels, 1218 restaurants, and 305 attractions. A hotel, a restaurant, and an attraction have 9, 9, and 6 attributes respectively. *inc-BiTOD* dataset has 23% inconsistent dialogs. Note that we do not simulate attraction KB as they rarely change. We simulate availability of hotels using a Bernoulli process.

	<i>inc-bAbI</i>	<i>inc-BiTOD</i>		
		Hotel	Restaurant	Attraction
Train Dialogs	1000	865	465	283
Val Dialogs	1000	84	56	29
Test Dialogs	1000	142	64	45

Table 6: No. of dialogs in train, validation and test sets.

B DKAF Details

DKAF consists of four models - RI, RD, RC, and reward function. We first present component modules present in *DKAF* models followed by separate discussion on each model. Finally, we provide training details for *DKAF*.

B.1 Component Modules

Dialog Encoder

We use a hierarchical dialog encoder (Sordani et al., 2015) in all the *DKAF* models. Our design follows hierarchical attention mechanism from (Yang et al., 2016). Hierarchical dialog encoder consists of two components - utterance level encoder and dialog level encoder.

Let $d = [u_1^u, u_1^a, u_2^u, u_2^a, \dots, u_m^u, u_m^a] = [u_1, u_2, \dots, u_{2m-1}, u_{2m}]$ be a given dialog with m turns where u_i is i^{th} utterance in the dialog. Let $u_i = [w_{i1}, w_{i2}, \dots, w_{il_i}]$ where w_{ik} is encoding for k^{th} token in u_i and l_i is number of tokens in u_i . Each token is encoded as sum of its token embedding (initialised randomly) and token tag embedding. Here, token tag is the entity type if token is an entity, null otherwise.

Utterance level encoder computes feature vectors for each token in u_i as

$$[h_{i1}, h_{i2}, \dots, h_{il_i}] = BiGRU([w_{i1}, w_{i2}, \dots, w_{il_i}])$$

Encoding h_i for each utterance is then computed using Luong attention (Luong et al., 2015) as

$$\mathbf{h}_i = \sum_{k=1}^{l_i} \alpha_k h_{ik}$$

$$\alpha_k = softmax(g_u(h_{ik}))$$

where $g_u(h_{ik})$ is a feed-forward network. Dialog level encoder takes $[\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_{2m}]$ as input and computes dialog feature vector \mathbf{c} using Luong at-

tention as

$$[H_1, H_2, \dots, H_{2m}] = GRU([\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_{2m}])$$

$$\mathbf{c} = \sum_{i=1}^{2m} \beta_i H_i$$

$$\beta_i = \text{softmax}(g_d(H_i))$$

where g_d is another feed forward network. Note that the hierarchical dialog encoder outputs hidden vectors for each token in an utterance, each utterance, and the entire dialog.

KB Encoder

KB encoder treats input KB as a relational graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ where \mathcal{V} and \mathcal{E} are set entities and relationships in KB respectively. \mathcal{R} denotes a set of all relation types based on domain. KB encoder uses L -relation graph convolution (r-GCN) layers (Schlichtkrull et al., 2018) for computing the KB entity feature. It forms a set $Z_0 = \{z_e^0\}_{\forall e \in \mathcal{V}}$ of entity embeddings as input to the first r-GCN layer. l^{th} GCN layer updates the features for entity $e \in \mathcal{V}$ as

$$z_e^l = \sigma \left(\sum_{r \in \mathcal{R}} \sum_{e' \in \mathcal{N}_e^r} W_r^{(l)} z_{e'}^{(l-1)} + W_0^{(l)} z_e^{(l-1)} \right)$$

where \mathcal{N}_e^r is set of entities that are related to e in G via relationship type r . Matrices $W^{(l)}$ s are parameters of the r-GCN layer and σ is ReLU activation function. We use $\mathbf{Z} = \{z_e\}_{\forall e \in \mathcal{V}}$ to denote the output of the last (L^{th}) r-GCN layer.

Memory Network

Memory network performs k -hop reasoning (Sukhbaatar et al., 2015) over a memory using given input query q^0 . In our case, KB entity features \mathbf{Z} form the memory while query q^0 depends upon the model (RD, RC or MEM reward model). At l^{th} hop, the memory network refines the query vector using Luong attention as

$$o^{(l)} = \sum_{k=1}^{|Z|} \gamma_k z_k$$

$$\gamma_k = \text{softmax}(g^l(z_k || q^{(l-1)}))$$

$$q^{(l)} = q^{(l-1)} + o^{(l)}$$

where g^l is a feed-forward network at l^{th} hop and $||$ is concatenation operator. The output of the memory network is final query vector $\mathbf{q} = q^{(k)}$.

B.2 Model Architectures

Row Insertion (RI)

For a given input (d, e_1, e_2, r) , RI model infuses position indicators for entities e_1 and e_2 in d as in Zhang and Wang (2015). It then encodes utterances in the resulting dialog with utterance level encoder described in section B.1. For an utterance u_i in the dialog, RI model appends \mathbf{h}_i with position vectors pos_{i_1} and pos_{i_2} relative to utterances containing e_1 and e_2 respectively. The concatenated vector is then passed to the dialog level encoder which computes the dialog feature vector \mathbf{c} .

RI model concatenates dialog features \mathbf{c} and entity features h_{e_1} and h_{e_2} from the dialog encoder and feeds them to a classification layer for relation type r .

Row Deletion (RD)

For a given input (d, K, ρ) , RD model computes dialog features and KB features using dialog encoder and KB encoder respectively. It computes encoding for the input ρ as $z_\rho = \sum_{e \in \rho} z_e$. Finally, it sets initial query $q^0 = \mathbf{c}$ and reasons over KB entity encoding using memory network to get refined query vector \mathbf{q} . Finally, it concatenates vectors \mathbf{q}, z_ρ and passes the resultant through a binary classifier layer.

Row Completion (RC)

Let (d, e_s, r, K) be input to RC model. RC model infuses position indicators and position vectors with respect to e_s and encodes resulting dialog using dialog encoder. It encodes K using KB encoder. It forms initial vector $q^0 = f(\mathbf{c} || h_{e_s})$ where f is a feed-forward layer as input to memory network. Finally, it combines memory network output \mathbf{q} with entity features z_{e_s} and feeds the resultant to a feed-forward layer that performs predictions over E_r of possible target entities.

Masked Entity Model (MEM)

Recent works (Wu et al., 2019; He et al., 2020a; Raghu et al., 2021b; He et al., 2020b) use pointer networks that copy entities required in the agent response from dialog history tokens and KB entities. Consequently, we design our MEM model $P(e|H_e, K)$ as a dual pointer network as

$$P(e|H_e, K)$$

$$= \lambda P_{kb}(e|H_e, K) + (1 - \lambda) P_{ctx}(e|H_e, K)$$

Here P_{kb} and P_{ctx} compute probabilities for copying entity e from KB entities and tokens from

Epoch	0	5	10	15	20
Train Acc.	0.784	1.0	1.0	1.0	1.0
Val Acc.	0.775	1.0	1.0	1.0	1.0

Table 7: Progress of training and validation accuracy of RI on *inc-bAbI*

masked dialog history H_e respectively. λ is a soft-gate to select entity e from H_e and the KB.

MEM model consists of hierarchical dialog encoder, KB encoder and memory network discussed earlier. For a given input (H_e, K) , MEM model uses position indicators and features with respect to $\langle mask \rangle$ token and computes dialog features using dialog encoder. It encodes K using KB encoder. It forms initial query q^0 to memory network as concatenation dialog features c and $\langle mask \rangle$ token features h_m . It receives q as output of the memory network.

MEM model computes P_{kb} over KB entities using Luong attention between concatenated vector $(q||h_m)$ and KB entity encoding Z . Similarly, it computes P_{ctx} using Luong attention between $(q||h_m)$ and H_e token encoding from dialog encoder. Finally, it computes soft-gate $\lambda = g_2(q)$ where g_2 is a feed-forward network.

B.3 Training Details

We find that following hyper-parameter setting works decently across all *DKAF* models. We use input embedding size of 100, learning rate of $1e-4$ and batch size of 32. For RD, RC and MEM models, we use entity embedding size of 100 and 8 r-GCN layers in KB encoder and 8 hops reasoning in the memory network. We train RI, RD, RC and MEM models for 30, 200, 200 and 100 epochs. It takes around 4 hours to train *DKAF* for both *inc-bAbI* and *inc-BiTOD* datasets.

Since the problem assumes no annotated data, we use either distant supervision or reinforcement learning to train the models. We track the training progress of each model in *DKAF* as follows.

Row Insertion The RI model is relation classifier trained using distantly supervised data. We use classifier accuracy as a metric to measure progress during training. The training and validation accuracy of the RI models over epochs on the *inc-bAbI* dataset is shown in table 7.

Row Deletion We use RL to train the RD model. We report the average reward across epochs for *inc-bAbI* dataset in table 8.

Epoch	0	10	100	180	190
Avg. Reward	-0.590	-0.002	0.710	0.927	0.937

Table 8: Progress of average reward for RD on *inc-bAbI*

Epoch	0	10	100	180	190
Avg. Reward	-0.649	-0.255	0.272	0.674	0.883

Table 9: Progress of average reward for RC on *inc-bAbI*

Row Completion We use RL to train the row completion model as well. Here too, we report the average reward across epochs for *inc-bAbI* dataset in table 9:

B.4 DKAF Model Evaluations

Row Insertion F1: We measure efficacy of RI in extracting correct rows from given dialog d . Let K^{ri} denote KB obtained post row insertion. Let $R \subseteq R_d$ be the set of rows that participate in d . Note that RI can only extract rows from R . We compute F1 with following precision and recall $pr = |R \cap (K^{ri} \setminus K_T)| / |(K^{ri} \setminus K_T)|$ and $re = |R \cap (K^{ri} \setminus K_T)| / |R|$. We now report Macro F1 across all the dialogs.

Row Deletion F1: During simulation, we obtain set D_g of rows in K_T that are misaligned with the dialog. Let D_p denote RD’s predicted set of rows for deletion. We compute F1 with following precision and recall $pr = |D_p \cap D_g| / |D_p|$ and $re = |D_p \cap D_g| / |D_g|$. We now report Macro F1 across all the dialogs.

Row Deletion F1: Let K^{rd} denote KB obtained post row deletion. Then, $D_p = K^T \setminus K^{rd}$ is set of rows deleted by RD and $D_g = K_T \setminus K_d$ is gold deletion set. We compute F1 with following precision and recall $pr = |D_p \cap D_g| / |D_p|$ and $re = |D_p \cap D_g| / |D_g|$. Note that our $K_T \setminus K_d$ can also contain rows that may be neutral to the task (for example, non-participating restaurants in *inc-bAbI*). Consequently, the recall we get significantly underestimates the actual model performance.

Row Completion Accuracy: In *inc-bAbI*, the RC model introduces ratings to the newly added rows. Recommendations in *inc-bAbI* strictly follow the rating order (higher to lower) of the restaurants in KB. Consequently, we consider a prediction by the RC model to be correct if the predicted rating

fits into the rating order in the KB. We then report accuracy across all predictions of the RC model.

C Rule-based Baseline

We propose a rule-based KB correction framework with the least possible dataset-specific rules that can be applied to any dataset. The rules of the three components of the framework are as below. We use the same notations that are used to explain the different components of *DKAF*.

Row Insertion Let (e_1, r, e_2) be a candidate relationship as defined in 4.1 where e_1 and e_2 are entities in input dialog d . We use the following rules for deciding whether relation (e_1, r, e_2) to be added to KB.

1. If e_1 is missing in the KB, insert a new row for e_1 .
2. Add relationship (e_1, r, e_2) to the new row if e_2 is the closest type-consistent entity to e_1 in the dialog.
3. If e_2 is uniquely associated with some entity in KB (for example phone number of a restaurant), do not insert (e_1, r, e_2) to the new row.

Row Deletion We delete a row from the KB if none of the entities unique to that row occur in the dialog.

Row Completion Rules for row completion are highly dataset specific and require considerable domain expertise. Since *inc-bAbI* is a synthetic dataset, we can derive a reasonable rule for row completion. Here, we add the rating for newly added restaurants such that the order in which restaurants are suggested in the dialog is respected.

Such a rule-based system may not capitalize on fine-grained patterns present in the data for each domain. Note that with detailed domain knowledge, we can design a rule-based approach for row insertion (RI), row deletion (RD), and row completion (RC), which may work for resolving the dialog-KB inconsistencies to a reasonable extent. But such detailed domain-specific knowledge is not always available or may be expensive to collect for every dataset. In contrast, our proposed *DKAF* can be trained to solve dialog-KB inconsistency in any dataset without any extra domain information.

	lr	warmup	bs	epochs
<i>inc-bAbI</i>	3e-5	0.1	32	4
<i>inc-BiTOD</i>	3e-5	0.1	32	10

Table 10: Best Hyperparameters for SimpleTOD for *inc-bAbI* and *inc-BiTOD*

	learning rate	dropout	no. of hops
GLMP	1e-4	0.1	1, 3
CDNet	1e-4	0.05	3

Table 11: Best Hyperparameters for GLMP and CDNet for *inc-bAbI* and *inc-BiTOD*

D Training baseline models

We adapt SimpleTOD to end-to-end setting and implement it using HuggingFace library⁴. Please refer D for more details.

D.1 SimpleTOD for end-to-end TOD

We adapt the input representation given by Hosseini-Asl et al. (2020) to end-to-end TOD setting. Our encoding scheme is given in table 20. Encoded input is then tokenized using GPT2 tokenizer and passed to the model. During training, the model is optimized for log-likelihood of response given context and KB. During inference, model generates a system response provided context and KB using greedy decoding (Hosseini-Asl et al., 2020). For SimpleTOD, we performed grid search on four parameters: learning rate, warm ratio, batch-size and number of epoch for both *inc-bAbI* and *inc-BiTOD*. The hyperparameters for best performance are reported in table 10.

D.2 GLMP and CDNet

For CDNet and GLMP we are using the same hyper-parameters as mentioned in their respective original papers. The hyperparameters that give us the best results for both *inc-bAbI* and *inc-BiTOD* are mentioned in the table 11. For GLMP, we obtain the best performance at one of two values of number of hops mentioned in the table.

We use publicly available implementations for FG2Seq⁵ and CDNet⁶ baselines.

⁴<https://huggingface.co/>

⁵<https://github.com/scoyer/FG2Seq>

⁶<https://github.com/dair-iitd/AggNet>

	<i>inc</i> -bAbI	<i>inc</i> -BiTOD
GLMP	1 hours	0.5 hour
CDNet	9 hours	7 hours
SimpleTOD	4 hours	2.5 hours

Table 12: Average compute time for all the models for *inc*-bAbI and *inc*-BiTOD

	Response Acc.	Dialog Acc.
CDNet	96.33	64.9
CDNet + DKAF	98.34	79.8

Table 13: Incremental KB Analysis

E Compute Resources

All experiments were run on a single Nvidia V100 GPU with 32GB of memory. *DKAF* has an average runtime of 4 hours on both *inc*-bAbI and *inc*-BiTOD. The compute time for model training for all three models are mentioned in table 12. For SimpleTOD, *DKAF* modified versions of *inc*-bAbI and *inc*-BiTOD take, the same average compute time as the original datasets.

F Domain Specific Analysis

During our experiments, we found that *DKAF* exhibits the same trend across the three domains of *inc*-BiTOD dataset: hotels, restaurants, and attractions. We have compared the domain-wise results in table 14. It can be observed that SimpleTOD is the best baseline on *inc*-BiTOD dataset across all three domains. Also, SimpleTOD trained with *DKAF* gives us a gain in performance with the best Entity F1 and KB F1 across all domains. In contrast, rule-based KB correction is performing worse than even SimpleTOD, showing that more domain-specific rules are required to obtain better scores.

G Incremental KB Size Analysis

In this section, we conducted experiments to check the effect of increase in KB size on the efficacy of *DKAF*. For our experiments, we systematically increased the size of the KB in *inc*-bAbI dataset by adding new restaurants to the associated training KB. We reported the finding in table 13 which shows that there is a limited effect on the expected trend. Because of the constrained input sequence length of SimpleTOD we have conducted this experiment on CDNet.

H Human Evaluation Details

Our team of annotators consists of two graduate-level students who volunteered for this task. Each of them has completed a course in either Machine Learning or Natural Language Processing, equipping them with the necessary knowledge and expertise. We have great confidence in the quality of their annotations. Additionally, we conducted a thorough review of a selection of randomly chosen annotated samples and found them to be satisfactory. Inter-annotator agreement was $\kappa = 0.31$ (Cohen, 1960) for the relevance score.

A snapshot of the portal used for collecting human evaluation is shown in figure 5. And the instructions provided to the human annotators are listed below:

1. What is the task about?

There are 50 dialog context response pairs in the HTML file. Each context response pair dictates a scenario where user is enquiring the agent about hotels, restaurants and attractions to visit. User can optionally request for additional attributes like phone number and address and can make a booking. Agent is expected to suggest hotel, restaurant and attraction with the highest rating among available options. Each context response pair has an associated knowledge base (table) where rows corresponding to top-rated entities are highlighted. Along with the context response pair, there are outputs of different dialog systems (randomly shuffled). You are requested to annotate each system-generated output along two dimensions: relevance and grammar, using the following scale:

- (a) SA: Strongly Agree
- (b) A : Agree
- (c) N : Neutral
- (d) D : Disagree
- (e) SD: Strongly Disagree

2. How to judge relevance?

- (a) Strongly Agree - when the generated output conveys the intended information—correct entity (hotel/restaurant/attraction) and its attributes (address, phone, rating, etc). Also, when generated, output requests correct input from the user.

Model	Hotels				Restaurant			Attraction	
	Bleu	Ent. F1	KB Ent. F1	Ent. F1	KB Ent. F1	Ent. F1	KB Ent. F1	Ent. F1	KB Ent. F1
GLMP	15.29	0.6743	0.6326	0.6839	0.6316	0.6640	0.6279	0.6335	0.6502
CDNet	19.37	0.7717	0.7445	0.8188	0.7975	0.6879	0.6440	0.6788	0.6783
SimpleTOD	20.28	0.7862	0.7566	0.8255	0.7966	0.7118	0.6633	0.7233	0.7488
SimpleTOD + Rule-based	21	0.7611	0.7733	0.7996	0.8023	0.6890	0.7239	0.6962	0.7236
SimpleTOD + <i>DKAF</i>	24.91	0.8187	0.8330	0.8402	0.8616	0.7915	0.7677	0.7400	0.8232
SimpleTOD + <i>DKAF</i> w/o RI	19.92	0.7779	0.7488	0.8142	0.7891	0.7200	0.6737	0.6840	0.7034
SimpleTOD + <i>DKAF</i> w/o RD	23.48	0.7973	0.7924	0.8264	0.8226	0.7422	0.7185	0.7400	0.7949

Table 14: Domain Specific results of *inc*-BiTOD dataset

- (b) Agree – when generated output contains partial information (e.g., when user request address and phone number but output contains only address).
- (c) Neutral – when generated output is hard to decide whether its right or wrong.
- (d) Disagree - when the generated response is somewhat unacceptable (e.g., re-querying already known information like cuisine for restaurants and name of the user for booking).
- (e) Strongly Disagree – when the generated output contains incorrect information (entities or attributes) for given conversation context.

In some cases, generated output contains number of search results of the form #number. For example, there are #3 available hotels, I recommend *jw_marriott_hotel_hong_kong* which has a rating of 9.

Since KB provided does not contain this information, you are expected to ignore this term in your evaluation.

3. How to judge grammar?

The grammar of the response is independent of the dialog context or ground truth. A system output can be marked strongly disagree for relevance and still be marked strongly agree for grammar. You can make your own rules about what each rating in the scale means for grammar, but please be consistent with the rules you come up with.

4. Can I use any browser?

Please use only Firefox as other browsers don't allow you to save the annotations to a json file in your local disk. Before you start the annotation please enter about:config in address bar of Firefox and in the config page set `privacy.file_unique_origin` to False.

5. How do I send you the annotations back?

After you finish the annotating the file, please click the save annotations button at the bottom of the page. This should save a json file with the same name as the html file in the same folder as the html file. Please send me that json file.

I *inc*-bAbI Examples

Table 15 demonstrates *DKAF* updates to training KB given a dialog context. Comparison responses generated by SimpleTOD model with and without *DKAF* is shown in Table 16.

J *inc*-BiTOD Examples

Table 17 demonstrates *DKAF* updates to training KB given a dialog context. Table 18 and 19 compares responses generated by SimpleTOD model with and without *DKAF*.

Conversation-11 (Domain: hotels)

name	rating	location	stars	price_level	price_per_night	phone_number	ref_number	number_of_rooms
whole_sunshine_international_hotel	10	tsim_sha_tsui	1	cheap	765	852_6408_9185	ogd7jbpp	9
wing_wah_hostel	9	mong_kok	1	cheap	507	852_9157_9334	fjyhtqv5	0
dragon_hostel	8	mong_kok	1	cheap	761	852_8389_5138	pej3y5ma	1

user: i would like to look for a 1 star hotel .

agent: which location would you prefer ?

user: i am ok with any location .

		Relevance	Grammar
Algo-1	"would you have preferences for hotel rating ?"	<input type="radio"/> SA <input type="radio"/> A <input type="radio"/> N <input type="radio"/> D <input type="radio"/> SD	<input type="radio"/> SA <input type="radio"/> A <input type="radio"/> N <input type="radio"/> D <input type="radio"/> SD
Algo-2	"would you like a cheap or expensive hotel ?"	<input type="radio"/> SA <input type="radio"/> A <input type="radio"/> N <input type="radio"/> D <input type="radio"/> SD	<input type="radio"/> SA <input type="radio"/> A <input type="radio"/> N <input type="radio"/> D <input type="radio"/> SD

Figure 5: Figure shows a snapshot of the portal used for human evaluation

Utterances	
User	hi
Agent	hello what can i help you with today
User	can you book a table in london
Agent	i'm on it
User	<SILENCE>
Agent	any preference on a type of cuisine
User	with italian food
Agent	how many people would be in your party
User	we will be two
Agent	which price range are looking for
User	i am looking for a expensive restaurant
Agent	ok let me look into some options for you
User	<SILENCE>
Agent	api_call italian london two expensive
User	instead could it be for four
Agent	sure is there anything else to update
User	actually i would prefer in a moderate price range
Agent	sure is there anything else to update
User	actually i would prefer with british food
Agent	sure is there anything else to update
User	no
Agent	ok let me look into some options for you
User	<SILENCE>
Agent	api_call british london four moderate
User	<SILENCE>
Agent	what do you think of this option: resto_london_moderate_british_6stars
User	no i don't like that
Agent	sure let me find an other option for you
User	<SILENCE>
Agent	what do you think of this option: resto_london_moderate_british_4stars
User	no this does not work for me
Agent	sure let me find an other option for you
User	<SILENCE>
Agent	what do you think of this option: resto_london_moderate_british_1stars
User	let's do it
Agent	great let me do the reservation
User	what is the phone number of the restaurant
Agent	here it is resto_london_moderate_british_1stars_phone
User	do you have its address
Agent	here it is resto_london_moderate_british_1stars_address
User	thank you
Agent	is there anything i can help you with
User	no thanks
Agent	you're welcome

Training KB							
R_name	R_phone	R_cuisine	R_address	R_location	R_number	R_price	R_rating
resto_london_moderate_british_4stars	resto_london_moderate_british_4stars_phone	british	resto_london_moderate_british_4stars_address	london	four	moderate	4
resto_london_moderate_british_6stars	resto_london_moderate_british_6stars_phone	british	resto_london_moderate_british_6stars_address	london	four	moderate	6
resto_london_moderate_british_8stars	resto_london_moderate_british_8stars_phone	british	resto_london_moderate_british_8stars_address	london	four	moderate	8

DKAF updated KB							
R_name	R_location	R_cuisine	R_number	R_price	R_phone	R_address	R_rating
resto_london_moderate_british_4stars	london	british	four	moderate	resto_london_moderate_british_4stars_phone	resto_london_moderate_british_4stars_address	4
resto_london_moderate_british_6stars	london	british	four	moderate	resto_london_moderate_british_6stars_phone	resto_london_moderate_british_6stars_address	6
resto_london_moderate_british_1stars	london	british	four	moderate	resto_london_moderate_british_1stars_phone	resto_london_moderate_british_1stars_address	2

Table 15: *DKAF* updates to training KB given a dialog context. *DKAF* adds a new row for *resto_london_moderate_british_1stars* which appears in the dialog. *DKAF* deletes two rows *resto_london_moderate_british_8stars* that has rating higher than restaurants occurring in dialog upsetting dialog reasoning. *DKAF* also adds rating 2 to *resto_london_moderate_british_1stars* maintaining correct rating order. Insertion and deletions are marked in **green** and **red** respectively. Entities in dialog are marked in **blue**.

Utterances	
User	hi
Agent	hello what can i help you with today
User	can you book a table in madrid with indian food for four
Agent	i'm on it
User	<silence>
Agent	which price range are looking for
User	i am looking for a moderate restaurant
Agent	ok let me look into some options for you
User	<silence>
Agent	api_call indian madrid four moderate
User	instead could it be with spanish cuisine
Agent	sure is there anything else to update
User	actually i would prefer in rome
Agent	sure is there anything else to update
User	no
Agent	ok let me look into some options for you
User	<silence>
Agent	api_call spanish rome four moderate
User	<silence>
Agent	what do you think of this option: resto_rome_moderate_spanish_8stars
User	do you have something else
Agent	sure let me find an other option for you
User	<silence>

KB							
r_name	r_phone	r_cuisine	r_address	r_location	r_number	r_price	r_rating
resto_rome_moderate_spanish_8stars	resto_rome_moderate_spanish_8stars_phone	spanish	resto_rome_moderate_spanish_8stars_address	rome	four	moderate	8
resto_rome_moderate_spanish_7stars	resto_rome_moderate_spanish_7stars_phone	spanish	resto_rome_moderate_spanish_7stars_address	rome	four	moderate	7
resto_rome_moderate_spanish_6stars	resto_rome_moderate_spanish_6stars_phone	spanish	resto_rome_moderate_spanish_6stars_address	rome	four	moderate	6
resto_rome_moderate_spanish_5stars	resto_rome_moderate_spanish_5stars_phone	spanish	resto_rome_moderate_spanish_5stars_address	rome	four	moderate	5

Model	Response
SimpleTOD	what do you think of this option: resto_rome_moderate_spanish_6stars
SimpleTOD + DKAF	what do you think of this option: resto_rome_moderate_spanish_7stars
Gold	what do you think of this option: resto_rome_moderate_spanish_7stars

Table 16: SimpleTOD incorrectly suggests *resto_rome_moderate_spanish_6stars* which has lower rating than *resto_rome_moderate_spanish_7stars*. SimpleTOD + DKAF suggests correct restaurants.

Utterances	
User	i am interested in finding a hotel .
Agent	how many stars should the hotel you are looking for have , and do you prefer a specific location ?
User	the hotel i am looking for should have at least 4 stars and i don't mind where it is located but it's rating should be at least 6 .
Agent	alright , do you have any other special requests ?
User	that would be all .
Agent	i found #3 hotels from which i would recommend mira_moon_hotel which has a rating of 9 .
User	i want to make a reservation at this hotel .
Agent	can you tell me your name please and how many nights you are going to stay at the hotel ?
User	my name is jean and i will stay there for 1 night . i am in need of having 6 rooms .
Agent	can you tell me when you will go there ?
User	i will go at the hotel on july 5 .
Agent	can you please confirm that you want to book 6 rooms for 1 night from july 5 at the mira_moon_hotel hotel ?
User	yes , that is what i want .
Agent	alright , your reservation has been made and the reference number for it is 3pw8nsim .

Training KB

name	rating	location	stars	price_level	price_per_night	phone_number	ref_number	number_of_rooms
the_t_hotel	10	pok_fu_lam	4	moderate	806	852_0586_7553	3v65a5oe	8
the_ritzcarlton_hong_kong	10	kowloon	5	expensive	2134	852_6768_3145	joaf239b	4
mier_serviced_apartments	7	central_district	4	moderate	885	852_0335_4038	rmrateru	2

DKAF updated KB

name	rating	location	stars	price_level	price_per_night	phone_number	ref_number	number_of_rooms
mira_moon_hotel	9		4				3pw8nsim	
mier_serviced_apartments	7	central_district	4	moderate	885	852_0335_4038	rmrateru	2

Table 17: DKAF updates to training KB given a dialog context. DKAF adds a new row for [mira_moon_hotel](#) which appears in the dialog. DKAF deletes two rows ~~the_t_hotel~~ and ~~the_ritzcarlton_hong_kong~~ that have rating higher than [mira_moon_hotel](#) upsetting dialog reasoning. Insertion and deletions are marked in green and red respectively. Entities in dialog are marked in blue.

name	rating	location	stars	price_level	price_per_night	phone_number	ref_number	number_of_rooms
alohas_hostel	9	kowloon	2	cheap	614	852_5153_2856	tt4jgzs2	9
ocean_inn	9	kowloon	2	cheap	632	852_7571_3796	8d2w237w	10
just_inn	8	tsim_sha_tsui	2	cheap	481	852_4281_9162	j3i1qob0	10
dragon_hostel	8	mong_kok	1	cheap	761	852_8389_5138	pej3y5ma	1
evergreen_hotel	7	kowloon	3	cheap	699	852_2759_1053	hmapwuwmm	2

Utterances	
User	hello , i would need a good hotel to stay .
Agent	are you looking for cheap or expensive hotel ?
User	i would need a hotel of cheap cost of living .
Agent	how about your preferences for the location and the star of the hotel ?
User	i am fine with any locations . the hotel would be minimum 1 stars rating .

Model	Response
SimpleTOD	there are #2 hotels available according to your preferences . i would recommend dragon_hostel with rating of 8 .
SimpleTOD + DKAF	i found #3 hotels from which i would recommend the alohas_hostel which has a rating of 9 .
Gold	ok , there are #5 available hotels that match your requirements . i would recommend alohas_hostel with 9 stars rating .

Table 18: SimpleTOD incorrectly suggests [dragon_hostel](#) which has lower rating than [alohas_hostel](#). SimpleTOD + DKAF suggests correct hotels.

name	rating	location	address	cuisine	dietary_restrictions	price_level	phone_number	ref_number
gaia_ristorante	8	sheung_wan	gf_grand_millennium_plaza_181_queens_road_central_sheung_wan	italian	gluten_free_options	expensive	852_2167_8200	gpwwc87k

Utterances	
User	hello , assistant ?
Agent	hello , how may i help you today ?
User	i would like a restaurant .
Agent	what sort of price and rating would you like ?
User	i would like an expensive restaurant with a rating of at least 3 . i don't want any russian cuisine .
Agent	what sort of location would you like ? do you have any dietary restrictions ?
User	i would like gluten_free_options near sheung_wan .

Model	Response
SimpleTOD	there are #2 restaurants that match your requirements . i would recommend falafel_tlv . it serves middle_eastern food and has a rating of 9 .
SimpleTOD + DKAF	i found a restaurant called gaia_ristorante . it serves italian food and has a rating of 8 . it is located at gf_grand_millennium_plaza_181_queens_road_central_sheung_wan and the phone number is 852_2167_8200 .
Gold	i recommend gaia_ristorante , which offers italian food and has a rating of 8 . it is located at gf_grand_millennium_plaza_181_queens_road_central_sheung_wan . you can call them at 852_2167_8200 .

Table 19: SimpleTOD hallucinates *falafel_tlv* hotels which does not exist in the KB.

Context	[context] [usr] good morning [sys] hello what can i help you with today ... [usr] do you have something else [endofcontext]
KB	[db][row] 60_west_hotel location sheung_wan, phone_number 852_9359_3222, ..., ref_number 9xkg8w7n [endofrow] [row] akvo_hotel location sheung_wan, phone_number 852_6317_4051, ..., ref_number ud0jiny2 [endofrow][endofdb]
Response	[response] i found #3 hotels from which i would recommend 60_west_hotel which has a rating of 8 . [endofresponse]

Table 20: SimpleTOD input representation for end-to-end TOD task