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

Anonymous ACL submission

Abstract

A task-oriented dialog (TOD) agent often grounds its responses in an external knowledge base (KB), which can be dynamic and may undergo frequent updates. Learning a TOD agent thus necessitates saving the KB snapshot contemporary to each individual training dialog. However, only the latest KB snapshot is often available during training. As a result, inconsistencies can arise in training data where dialogs and KB deliver diverging facts, potentially confusing the TOD learner.

In this work, we propose the novel problem of learning a TOD system with training data that has dialog-KB inconsistencies. We introduce two datasets for the task, created by systematically modifying two publicly available dialog datasets. We show that existing end-to-end TOD architectures suffer loss in performance due to these inconsistencies. In response, we propose a **D**ialog-**K**B **A**rbitration **F**ramework (*DKAF*) that reduces the inconsistencies – based on the dialog, *DKAF* introduces new rows to the KB and removes contradictory ones. The resulting KB is then used for training downstream TOD agents. We show that TOD agents trained with *DKAF* recover well from performance loss due to inconsistencies.

1 Introduction

A task-oriented dialog (TOD) system often requires information from a knowledge base (KB), to complete user goals like restaurant reservation, flight booking and managing personal calendars. This paper follows the recent line of research in *end-to-end* TOD, where dialog agents are trained based only on a set of training dialogs and an associated KB, without any expensive manual annotation (Wu et al., 2019; Qin et al., 2020; Raghu et al., 2021b).

KBs generally evolve over time – new rows (e.g., for new restaurants) may get added, and older ones removed. Figure 1 illustrates this, where K_1 and K_2 are KB snapshots at times t_0 and $t_0 + \Delta t$. K_1 transforms into K_2 as restaurant *Bangkok City* becomes available and *La Margherita* and *Prezzo* become unavailable for reservation. Dialogs d_1 and d_2, grounded into contemporary KB snapshots K_1 and K_2 respectively, produce different recommendations for the user goal of reserving an Italian restaurant. In d_1, agent makes two recommendations from K_1, whereas in d_2, no recommendation is feasible.

Effective learning of TOD on such dialogs necessitates saving KB snapshots for each training dialog. However, at training time, only a single KB snapshot (generally, the latest) may be available, which will get associated with all the training dialogs. This can cause KB and dialogs to portray diverging information resulting in *dialog-KB inconsistencies*. In the running example, K_T denotes the training 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 in training data can hinder learning of a TOD model. During training, inconsistencies can force the model to produce entities in its responses that are un-grounded (*La Margherita* in d_1). Furthermore, many agent responses depend upon reasoning over the KB – inconsistencies can upset these reasoning patterns by causing misalignment between dialog and KB (e.g., d_2). In either case, model either ends-up learning incorrect patterns or memorizes responses, leading to poor generalization. In this work, we propose the novel problem of end-to-end learning of TOD systems where training data has dialog-KB inconsistencies. We also present *DKAF* which predicts modifications to training KB, given a dialog context, to create a KB snapshot that would likely resemble the contemporary KB snapshot for the dialog. These generated KB snapshots along with the associated dialogs are then used to train any end-to-end TOD system.

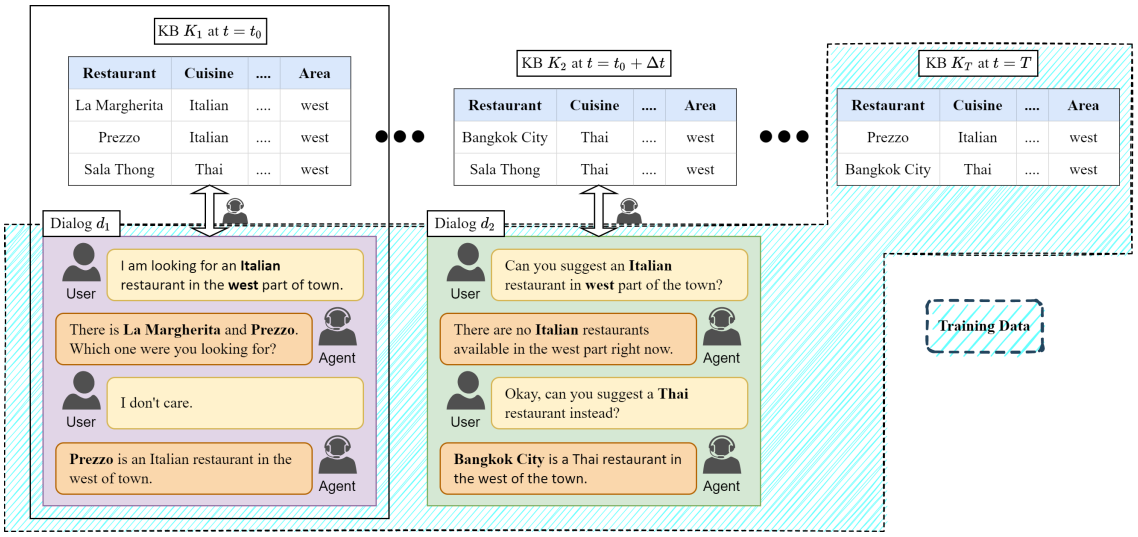

Figure 1: Figure shows snapshots of an evolving KB at times t_0, $t_0 + \Delta t$ and T. Over time, restaurants in the KB is changing, which is reflect in the KB snapshots K_1 and K_2 at time t_0 and $t_0 + \Delta t$ respectively. Dialogs d_1 and d_2 are consistent with KB snapshots K_1 and K_2. During 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.

Given a dialog, *DKAF* performs two kinds of updates on the KB. *DKAF* inserts to the KB new rows reflecting new entities and entity relations extracted from the dialog (e.g., inserting *La Margherita* to the KB for d_1). *DKAF* deletes, from the KB, rows that are misaligned with the dialog (e.g., removing *Prezzo* from the KB for d_2). Predicting these updates necessitates understanding 1) relationships among the entities occurring in the given dialog and 2) how the agent responses are grounded in the KB. *DKAF* incorporates these insights in three stages – row insertion, row deletion and row completion. *DKAF* is trained using a combination of weak supervision and reinforcement learning with reward depending upon the likelihood of generating gold agent utterance by the TOD model.

We construct two datasets by systematically infusing dialog-KB inconsistencies on bAbI (Bordes and Weston, 2017) and BiTOD (English) (Lin et al., 2021) datasets which we name *inc*-bAbI and *inc*-TOD respectively. Both *inc*-bAbI and *inc*-TOD have a subset of training dialogs with inconsistent information with respect to the associated KB – inconsistencies are randomly introduced by our dataset simulation process. Existing state-of-the-art models like CDNet (Raghu et al., 2021b) suffer losses when trained over these datasets. We show that *DKAF* trains effectively on these datasets and helps TOD models recover from the losses. In summary,

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

2. We present *DKAF* that alleviates dialog-KB inconsistencies by predicting KB updates based on a given training dialog, thus generating a KB snapshot likely to resemble the (latent) contemporary KB snapshot for the dialog.

3. We create two datasets for our task by systematic modification of publicly available bAbI and BiTOD datasets. We show that existing TOD models, trained in our setting, can perform poorly. Our experiments demonstrate that *DKAF* improves TOD performance.

We will release all resources for future research.

2 Related Work

End-to-end TOD models (Eric et al., 2017; Madotto et al., 2018; Lin et al., 2019; Raghu et al., 2021b, 2019; Wu et al., 2019; Madotto et al., 2018; He et al., 2020b; Yang et al., 2020; He et al., 2020a; Gou et al., 2021; Rony et al., 2022), 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)

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 $d = [u^u_1, u^a_1, u^u_2, u^a_2, ..., u^u_m, u^a_m]$ 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 $\hat{u}^a_i$ given dialog history $[u^u_1, u^a_1, u^u_2, u^a_2, ...u^u_i]$ 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. However, at training time a single KB snapshot K_T is available which gets associated with all training dialogs. This results in inconsistencies between dialogs and the KB. Accordingly, we propose task of learning end-to-end TOD system using $\{d_j, K_T\}_{j=1}^N$ where data has dialog-KB inconsistencies.

4 *DKAF*

Dialog-KB inconsistencies arise in a training dialog d_j when K_j, dialog's contemporary KB snapshot, differs from K_T. 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 (Sordoni et al., 2015). Encoder feature vectors for the dialog and entities are then passed through classifier network for relation r. Thus, RI model uses training dialog to identify relationships (e_1, r, e_2) missing from the KB. Figure 2 showcases this where *(Bangkok City, cuisine, Thai)* and *(Bangkok City, area, west)* get added to the KB.

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 $e1, e2 \in d$[1], 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 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^{ri}_j post insertion.

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 TOD system to generate 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 dialog encoder given in Section 4.1. Recent works (Banerjee and Khapra, 2019; Yang et al., 2020) showcase efficacy of GCNs in TOD modelling. Consequently, RD model includes an r-GCN (Schlichtkrull et al., 2018) KB encoder that computes KB entity features. Then, RD model reasons over KB entities using a memory network

[1] can be identified by NER, though in this work, we assume this is known

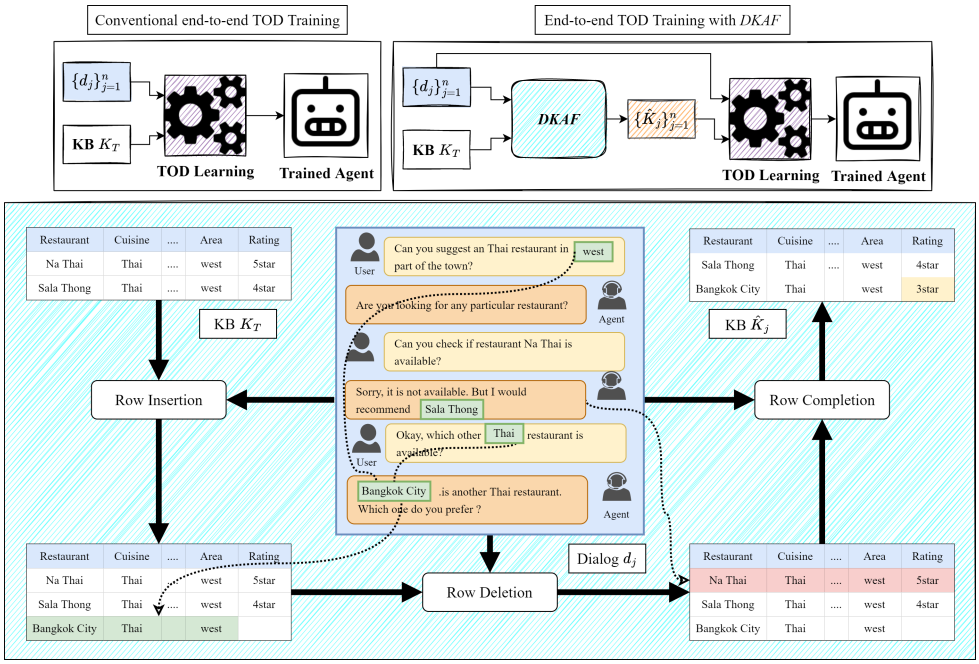

Figure 2: Comparison of conventional TOD learning (top-left) with TOD learning with *DKAF* (top-right). *DKAF* attempts to resolve dialog-KB inconsistencies by updating training KB K_T given a training dialog. Figure (bottom) shows *DKAF* in action with KB updates from row insertion, row deletion and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

(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.

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 an 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 task. We incorporate this insight using a novel masked entity modeling (MEM) task. Let e be an entity in the ith 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 <*mask*> token. Let E_a be the set of entities occurring in agent utterances in the dialog. 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

$$R_0(d, K, \rho) = sgn[\mathcal{L}(d, K \setminus \{\rho\}) - \mathcal{L}(d, K)]$$
$$R_1(d, K, \rho) = -R_0(d, K, \rho)$$

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 C.4.

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 it 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.

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

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

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, 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. So, all the 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*-TOD, respectively. bAbI dialog dataset consists of synthetically generated dialogs from restaurant reservation domain. BiTOD is a bilingual 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. Table 4 shows the train, validation and test splits of the *inc*-TOD and *inc*-bAbI datasets.

We follow a two-step procedure to simulate a dialog-KB inconsistent dataset. First, we generate an evolving KB by modifying its contents over time and maintaining a snapshot at each time step. Second, we assign a timestamp to each dialog and associate it with the 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., *cuisine=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 of the evolving KB that would be used for train. To mimic real-world settings, we only induce inconsistencies in the train dialogs. The test dialogs remain consistent.

To simulate the evolving KB, we add a binary random variable, named *available*, to each row in the KB and change its value over time as illustrated in Figure 3. We now describe how we simulate the KB evolution for the two datasets.

***inc*-bAbI:** In the real-world, a restaurant's availability is subject to temporal factors like day of the week and time of the day. Moreover, restaurants can have maintenance breaks and even go out of business. To mimic such a behaviour, we create a snapshot of the KB for every hour in a day by setting the number of restaurants available inversely proportional to the number of check-ins that occur during that hour of the day and day of the week. The check-in statistics are obtained from the Yelp dataset.[2] We 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. This simulation results in 20.7% of the train dialogs to be inconsistent with the last KB snapshot.

***inc*-TOD:** We set the availability of each KB entry following a Bernoulli distribution parameterized

[2] https://www.yelp.com/dataset

by a success probability p. We set p to 0.75 which results in 31% of the dialogs inconsistent with the last KB snapshot.

5.2 Algorithms

For all our experiments, we use CDNet (Raghu et al., 2021b), a state-of-the-art end-to-end TOD model, for learning TOD agents. We train CDNet using three settings:

CDNet: This is the vanilla CDNet trained with dialogs that are inconsistent with the KB. We use the $\{d_j, K_T\}_{j=1}^{N}$ pairs to train the CDNet model.

CDNet + *DKAF*: This is our proposed approach, which first performs KB arbitration for each dialog d_j with *DKAF* which results in $\hat{K}_j$. We use the $\{d_j, \hat{K}_j\}_{j=1}^{N}$ pairs to train the CDNet model.

CDNet + Oracle: During simulations, we save the KB snapshot K_j contemporary to each train dialog d_j. We use these $\{d_j, K_j\}_{j=1}^{N}$ pairs to train CDNet model. This setting provides an empirical upper bound for the performance of CDNet + *DKAF*.

We conduct similar experiments with another popular TOD model named GLMP (Wu et al., 2019) and report the results in 6.

5.3 Evaluation Metrics

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

As *inc*-TOD is human-generated, we follow Wu et al. (2019) and use BLEU Papineni et al. (2002) and Entity F1 Eric et al. (2017); Madotto et al. (2018) 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*-TOD along two dimensions: (i) *Relevance:* how useful are the responses given the dialog and KB, and (ii) *Naturalness:* how human-like are the predicted responses. Each dimension is annotated on a Likert scale of 0-4 (Likert, 1932).

5.4 Training Details

We fix the embedding size of CDNet to 200, its learning rate to 1e-4, batch size to 32 and dropout to 0.05, as these hyper-parameters give consistent performance across runs, and sample number of hops from $\{1, 3\}$. We observe that CDNet model achieves better validation performance when global row level attention is disabled. We select hyper-parameters that provide best response accuracy on *inc*-bAbI validation set and entity F1 on *inc*-TOD validation set. We repeat training with best hyper-parameters with three different initializations and report mean and standard deviation.

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 our main results. We first discuss the impact of dialog-KB inconsistencies on end-to-end TOD systems. We then discuss how well can *DKAF* mitigate the dialog-KB inconsistencies.

Impact of Dialog-KB Inconsistencies: To analyse the impact of dialog-KB inconsistencies on TOD agents, we compare the performance of CDNet with CDNet + Oracle. On *inc*-bAbI, CDNet has poor performance compared CDNet + Oracle with about 28 points loss in dialog accuracy. We analyse the predictions from the best performing models and find that CDNet+Oracle incorrectly predicts only 92 KB entities in all responses, whereas vanilla CDNet incorrectly predicts 596 KB entities. We observe a similar trend in *inc*-TOD with 12.85 point and 17 points drop in entity F1 and KB entity F1 respectively. We found that CDNet + Oracle incorrectly predicts 318 entities in all responses and CDNet incorrectly predicts 448 entities.

This drop in performance in vanilla CDNet trained with inconsistent dialogs is due to the memorization of KB entities rather than inferring them from the KB. When CDNet is trained with the Oracle KB snapshots, all entities in the dialogs are present in the KB. This enables the TOD agent in

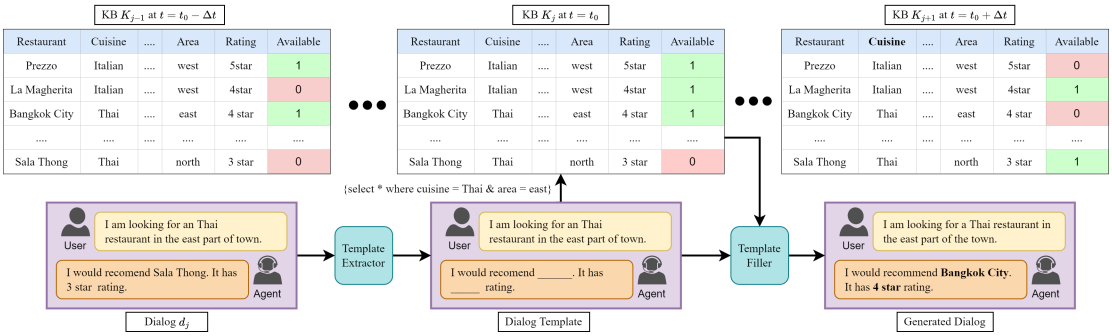

Figure 3: Figure shows the simulation pipeline used for generating datasets.

Models	*inc*-bAbI		*inc*-TOD		
	Response Acc.	Dialog Acc.	BLEU	Ent. F1	KB Ent. F1
CDNet	96.42 ± 0.25	64.20 ± 2.55	$\mathbf{18.00 \pm 0.97}$	0.68 ± 0.02	0.64 ± 0.02
CDNet + *DKAF*	$\mathbf{98.78 \pm 0.05}$	$\mathbf{86.37 \pm 0.30}$	17.77 ± 0.97	$\mathbf{0.70 \pm 0.02}$	$\mathbf{0.68 \pm 0.03}$
CDNet + Oracle	99.34 ± 0.17	92.30 ± 3.25	19.12 ± 0.39	0.82 ± 0.01	0.82 ± 0.01

Table 1: DKAF main Results

	Relevance	Naturalness
CDNet	3.08	3.44
CDNet + DKAF	3.31	3.42

Table 2: Human Evaluation on *inc*-TOD

learning to copy the KB entity necessary to generate the response. But when only the last snapshot of the KB is used, the KB entities in the dialogs used for training vanilla CDNet may not always be present in the KB, and hence the TOD agent is forced to memorize those KB entities rather than inferring them from the KB. Our analysis shows that the number of KB entities that are in the dialog but not in the KB for *inc*-bAbI and *inc*-TOD are 406 and 1125 respectively.

Efficacy of *DKAF*: We report performance of CDNet + *DKAF* model in Table 1. In *inc*-bAbI dataset, CDNet + *DKAF* exhibits substantial performance improvement over CDNet model with gains of 2.36 points and 22.17 points in response and dialog accuracies. To analyze the results of *DKAF* arbitration, we compare training KB K_T and arbitrated KB $\hat{K}_j$ and with contemporary KBs K_j for each dialog d_j in the training data. We treat the KBs as set of relationship triplets and compute average Jaccard similarity between K_j and $\hat{K}_j/K_T$ over all training dialogs. Similarity score between K_T and contemporary KB K_j is 0.78, while that between $\hat{K}_j$ and K_j improves to 0.87. This indicates that *DKAF* pushes K_T to be much closer to contemporary KB, which results in overall performance gains.

For *inc*-TOD dataset, CDNet + *DKAF* outperforms CDNet model in entity F1 and entity F1 KB metrics by a margin of 2 and 4.2 points. The gain in entity F1 KB is indicative of *DKAF*'s effectiveness in resolving inconsistencies. We analyse Jaccard similarity score between training KB K_T, arbitrated KB and contemporary KB. Similar to *inc*-bAbI, we observe that *DKAF* improves Jaccard similarity from 0.57 to 0.63. However, we observe quite a performance difference between CDNet + *DKAF* and CDNet + Oracle. We posit that this is partially because CDNet + Oracle has better coverage (0.89) over entities in test set compared to CDNet + *DKAF* model (0.79).

Human Evaluation: We summarize the human evaluation results on *inc*-TOD in Table 2. We randomly sample 55 (dialog-context,response) pairs from *inc*-TOD and three human judges labelled responses generated by CDNet and CDNet + *DKAF* on relevance and grammar on a Likert scale (0-4). We see that CDNet + *DKAF* outperforms the vanilla CDNet by 0.23 points on relevance.

6.2 Ablation Experiments

We perform ablation for each component in *DKAF* to measure how each stage contributes to overall *DKAF* performance. Table 3 reports our results. Row insertion is the major contributor to performkance of *DKAF*. For both *inc*-bAbI and *inc*-TOD, excluding row insertion leads to signif-

Configurations	inc-bAbI	inc-BiTOD
	Dialog Acc.	KB Ent. F1
CDNet + DKAF	86.37 ± 0.25	0.68 ± 0.02
CDNet + DKAF - RI	73.67 ± 1.08	0.64 ± 0.02
CDNet + DKAF - RD	75.40 ± 3.89	0.69 ± 0.00
CDNet + DKAF - RC	81.87 ± 5.69	0.68 ± 0.02
CDNet	64.20 ± 2.08	0.64 ± 0.02

Table 3: Ablation Results

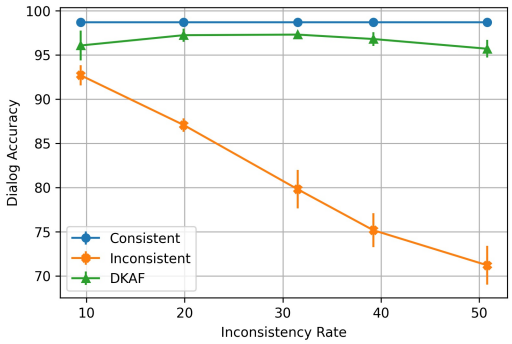

Figure 4: DKAF Incremental Analysis on *inc*-bAbI

icant performance drop. In case of *inc*-TOD, we observe that excluding row insertion also causes row deletion model to abstain from removing rows from the KB. Dropping row deletion results performance drop for *inc*-bAbI dataset while performance slightly improves form *inc*-TOD. Since in *inc*-bAbI, agent suggestions strictly follows restaurant ratings, inconsistencies upset the reasoning patterns much more severely requiring aggressive deletion. On the other hand, *inc*-TOD does not has such patterns resulting in a slight improvement in the performance. Finally, excluding row completion has a comparatively low-performance impact. The missing entities predicted by the RC module are those that are not mentioned in the dialog, but induced by using RL. There are only a few entities in the dialogs that gets affected by these missing latent entities. In *inc*-bAbI, restaurants are always recommended in descending order of ratings, guessing the rating is crucial for predicting the correct restaurant. Thus the major contribution of RC module is in guessing the appropriate rating for restaurants inserted by the RI module. On the other hand, *inc*-TOD 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 CDNet and CDNet + *DKAF* model. Figure 4 showcases the results. With an increasing number of dialog-KB inconsistencies, the performance of CDNet model decreases sharply. On the other hand, CDNet + *DKAF* is consistently able to recover from the performance drop with significant gains. Figure 4 also compares performance of CDNet + *DKAF* with CDNet + Oracle model which is an empirical upper bound in all the 5 cases.

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, losing over 25 accuracy points in one case. 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.

Limitations

DKAF model has only been tested on English data so far. Even within the current datasets, there is still some gap (as high as 10.8 accuracy points) between models trained on consistent data, versus those trained on inconsistent data, suggesting that more research is needed to bridge this gap further. 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 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.

Mihail Eric, Lakshmi. Krishnan, François Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. *ArXiv*, abs/1705.05414.

Yanjie Gou, Yinjie Lei, and Lingqiao Liu. 2021. Contextualize knowledge bases with transformer for end-to-end task-oriented dialogue systems. *ArXiv*, abs/2010.05740.

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*.

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. 1932. A technique for the measurement of attitude scales.

Zehao Lin, Xinjing Huang, Feng Ji, Haiqing Chen, and Ying Zhang. 2019. Task-oriented conversation generation using heterogeneous memory networks. *ArXiv*, abs/1909.11287.

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. Unsupervised 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*.

Md. Rashad Al Hasan Rony, Ricardo Usbeck, and Jens Lehmann. 2022. Dialokg: Knowledge-structure aware task-oriented dialogue generation. *ArXiv*, abs/2204.09149.

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 Sordoni, 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*.

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

A Data statistics

	inc-bAbI	*inc*-TOD
Train Dialogs	1000	1614
Val Dialogs	1000	169
Test Dialogs	1000	251

Table 4: No. of dialogs in train, validation and test sets.

B Model Components

B.1 Dialog Encoder

We use a hierarchical dialog encoder (Sordoni 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 $d = [u^u_1, u^a_1, u^u_2, u^a_2, ..., u^u_m, u^a_m] = [u_1, u_2, ..., u_{2m-1}, u_{2m}]$ be a given dialog with m turns where u_i is i^{th} utterance in the dialog. Let $u_i = [w_{i1}, w_{i2}, ..., w_{il_i}]$ 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

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

Encoding $\boldsymbol{h}_i$ for each utterance is then computed using Luong attention (Luong et al., 2015) as

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

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

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

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

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

$$\beta_i = softmax(g_d(H_i))$$

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

B.2 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 set of all relation types based on task ontology. KB encoder uses L-relation graph convolution (r-GCN) layers (Schlichtkrull et al., 2018) for computing KB entity feature. It forms set $Z_0 = \{z^0_e\}_{\forall e \in \mathcal{V}}$ 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

$$z^l_e = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{e' \in \mathcal{N}^r_e} W^{(l)}_r z^{(l-1)}_{e'} + W^{(l)}_0 z^{(l-1)}_e\right)$$

where $\mathcal{N}^r_e$ 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 $\boldsymbol{Z} = \{\boldsymbol{z}_e\}_{\forall e \in \mathcal{V}}$ to denote the output of the last (L^{th}) r-GCN layer.

B.3 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 $\boldsymbol{Z}$ forms the memory while query q^0 depends upon the model (RD, RC or MEM model). At l^{th} hop, memory network refines the query vector

using Luong attention as

$$o^{(l)} = \sum_{k=1}^{|Z|} \gamma_k \boldsymbol{z}_k$$
$$\gamma_k = softmax(g^l(\boldsymbol{z}_k || q^{(l-1)}))$$
$$q^{(l)} = q^{(l-1)} + o^{(l)}$$

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

C Model Architectures

C.1 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 following (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 $\boldsymbol{h_i}$ 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 $\boldsymbol{c}$.

RI model concatenates dialog features $\boldsymbol{c}$ 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.

C.2 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 $\boldsymbol{z}_\rho = \sum_{e \in \rho} \boldsymbol{z}_e$. Finally, it sets initial query $q^0 = \boldsymbol{c}$ and reasons over KB entity encoding using memory network to get refined final query vector $\boldsymbol{q}$. Finally, it concatenates vectors $\boldsymbol{q}, \boldsymbol{z}_\rho$ and passes the resulting vector through a binary classifier layer.

C.3 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 $q^0 = f(\boldsymbol{c}||h_{e_s})$ where f is a feed-forward layer as input to memory network. Finally, it combines memory network output $\boldsymbol{q}$ with entity features $\boldsymbol{z}_{e_s}$ and feeds resulting vector to a feed-forward layer that performs predictions over E_r of possible target entities.

C.4 Masked Entity Modelling

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 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), our MEM model uses position indicators and features with respect to <*mask*> 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 $\boldsymbol{c}$ and <*mask*> token features h_m. It receives $\boldsymbol{q}$ as output of the memory network.

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

D Hyper-parameters

We conducted experiments on two TOD models: CDNet and GLMP. For both the models we fixed the hyper-parameters after performing grid search over the validation set. For CDNet, we fix the input embedding size to 200, dropout to 0.05, learning rate to $1 \times e^{-4}$, batch size to 32 and the number of hops to 3 to obtain best performance for all the three model training configurations. For GLMP, we obtain best performance for the three training configuration at different settings. For this model too, we fixed the embedding size to 200, learning rate to $1 \times e^{-4}$, batch size to 32. But we have sampled the number of hops from $\{1, 3\}$ and dropout from $\{0.1, 0.3\}$. Table 5 shows the best parameter for number of hops and dropouts for GLMP.

glmp 3mins bitod (1 hr) cdnet

All experiments were run on a single Nvidia V100 GPU with 32GB of memory. DKAF has a

Models	*inc*-bAbI		*inc*-TOD	
	Hops	**Dropout**	**Hops**	**Dropout**
GLMP + Oracle	3	0.1	1	0.1
GLMP	1	0.1	3	0.1
GLMP + *DKAF*	1	0.1	1	0.1

Table 5: GLMP Hyperparameters

runtime of 4 hours on average on both *inc*-bAbI and *inc*-TOD. CDNet has an average runtime of 9 hours (17 min per epoch) ,and 7 hours (12 min per epoch) on *inc*-bAbI and *inc*-TOD respectively for all three model configurations. GLMP has an average runtime of 1.5 hour (2 mins per epoch) ,and 1 hours (3 min per epoch) on *inc*-bAbI and *inc*-TOD respectively for all three model configurations.

E GLMP Results

We provide GLMP results in table 6

F *DKAF* Example

Table 7 and 8 shows corrections by *DKAF* on *inc*-bAbI and *inc*-TOD samples.

G Dialog Example

Models	*inc*-bAbI		*inc*-TOD		
	Response Acc.	**Dialog Acc.**	**BLEU**	**Ent. F1**	**KB Ent. F1**
GLMP + Oracle	98.51 ± 0.16	81.67 ± 1.92	18.84 ± 1.04	0.73 ± 0.01	0.70 ± 0.02
GLMP	97.80 ± 0.06	72.03 ± 1.13	$\mathbf{14.10 \pm 1.52}$	0.60 ± 0.01	0.54 ± 0.02
GLMP + *DKAF*	$\mathbf{98.25 \pm 0.15}$	$\mathbf{78.63 \pm 1.72}$	13.66 ± 2.90	$\mathbf{0.63 \pm 0.02}$	$\mathbf{0.57 \pm 0.03}$

Table 6: GLMP Results

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

Inconsistent 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* corrected 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_1stars	resto_london_moderate_british_1stars_phone	british	resto_london_moderate_british_1stars_address	london	four	moderate	2

Table 7: An Example to demonstrate the correction made by *DKAF* to an inconsistent sample of *inc*-bAbI dataset

	Utterances
User	i want to book a hotel which should be expensive
System	what's your requirement about the stars of the hotel ?
User	at least 3 stars .
System	what's your requirements for hotel rating and your preferences for location ?
User	at least 5 rating any location
System	ok . what's your other requirements ?
User	that's all
System	13 available hotels , i recommend **four_seasons_hotel_hong_kong** . its rating is **10**
User	what's the price ?
System	**1305** hkd per night
User	book it please
System	what's your name ?
User	albert
System	how many nights would you like to stay ? when is your check in date ?
User	11 nights . the check in date is december 20 . i need 2 rooms .
System	can you confirm : albert is booking **four_seasons_hotel_hong_kong** hotel for 11 nights starting from december 20 . and you want to book 2 rooms .
User	correct
System	done booking ! your reference number is **amoxngso** .

Inconsistent KB

location	name	number_of_rooms	phone_number	price_level	price_per_night	rating	ref_number	stars
central_district	the_landmark_mandarin_oriental_hong_kong	5	852_6550_4214	expensive	1670	10	30opwf8i	5
hong_kong_island	the_upper_house	10	852_1521_1909	expensive	1381	10	lg9ic93b	5
tsim_sha_tsui	the_peninsula_hong_kong	7	852_0352_0315	expensive	1222	9	a4sk516l	5
new_territories	tai_o_heritage_hotel	9	852_9764_4460	expensive	1921	8	ghqmt465	4
tsim_sha_tsui	soravit_on_granville	5	852_1166_3319	expensive	1514	8	lx66bpkf	3

***DKAF* corrected KB**

location	name	number_of_rooms	phone_number	price_level	price_per_night	rating	ref_number	stars
	four_seasons_hotel_hong_kong			expensive	1305	10	amoxngso	
central_district	the_landmark_mandarin_oriental_hong_kong	5	852_6550_4214	expensive	1670	10	30opwf8i	5
hong_kong_island	the_upper_house	10	852_1521_1909	expensive	1381	10	lg9ic93b	5
tsim_sha_tsui	the_peninsula_hong_kong	7	852_0352_0315	expensive	1222	9	a4sk516l	5
new_territories	tai_o_heritage_hotel	9	852_9764_4460	expensive	1921	8	ghqmt465	4
tsim_sha_tsui	soravit_on_granville	5	852_1166_3319	expensive	1514	8	lx66bpkf	3

Table 8: An Example to demonstrate the correction made by *DKAF* to an inconsistent sample of *inc*-TOD dataset

r_name	r_phone	r_cuisine	r_address	r_location	r_number	r_price	r_rating
resto_paris_moderate_british_3stars	resto_paris_moderate_british_3stars_phone	british	resto_paris_moderate_british_3stars_address	paris	four	moderate	3
resto_paris_moderate_british_2stars	resto_paris_moderate_british_2stars_phone	british	resto_paris_moderate_british_2stars_address	paris	four	moderate	2
resto_paris_moderate_british_1stars	resto_paris_moderate_british_1stars_phone	british	resto_paris_moderate_british_1stars_address	paris	four	moderate	1

	Utterances
User	good morning
System	hello what can i help you with today
User	can you book a table for four in a cheap price range
System	i'm on it
User	<silence>
System	any preference on a type of cuisine
User	i love italian food
System	where should it be
User	rome please
System	ok let me look into some options for you
User	<silence>
System	api_call italian rome four cheap
User	instead could it be with british food
System	sure is there anything else to update
User	actually i would prefer in paris
System	sure is there anything else to update
User	actually i would prefer in a moderate price range
System	sure is there anything else to update
User	no
System	ok let me look into some options for you
User	<silence>
System	api_call british paris four moderate
User	<silence>
System	what do you think of this option: **resto_paris_moderate_british_3stars**
User	it's perfect
System	great let me do the reservation
User	may i have the address of the restaurant

Configurations	**Predictions**
Gold	here it is resto_paris_moderate_british_3stars_address
CDNet trained on incTOD	here it is resto_paris_moderate_british_2stars_address
DKAF + CDNet trained on incTOD	here it is resto_paris_moderate_british_3stars_address

Table 9: Example to compare the response generation performace of CDNet trained on *inc*-bAbI with and without *DKAF*

location	name	number_of_rooms	phone_number	price_level	price_per_night	rating	ref_number	stars
central_district	four_seasons_hotel_hong_kong	10	852_5706_6379	expensive	1305	10	amoxngso	5
central_district	the_landmark_mandarin_oriental_hong_kong	5	852_6550_4214	expensive	1670	10	30opwf8i	5
hong_kong_island	jw_marriott_hotel_hong_kong	10	852_7885_6633	expensive	2210	9	s5y9h2s3	5
tsim_sha_tsui	the_peninsula_hong_kong	7	852_0352_0315	expensive	1222	9	a4sk516l	5
tsim_sha_tsui	house_1881	2	852_0071_5353	expensive	1895	8	swm2n2uu	5
tsim_sha_tsui	shama_tsim_sha_tsui_hong_kong	4	852_6964_6875	expensive	1594	8	7bdtkw3y	0
new_territories	horizon_suite_hotel	1	852_2004_6097	expensive	1290	5	jyfbz1tm	0
tsim_sha_tsui	maharaja_guesthouse	10	852_0723_8650	expensive	1286	5	5wdd2n3l	1

	Utterances
User	hi , i am looking for hotel , do you have any recommendation for hotels ? the rating of the hotel should be at least 4 please
System	okay please any requirement about the stars of the hotel and where do you want the hotel to locate ?
User	okay i am fine with any stars and i am good with all locations . the price level should be expensive
System	there are 21 available hotels , i recommend four_seasons_hotel_hong_kong which has a rating of 10 .
User	how much is the hotel per night please

Configurations	**Predictions**
Gold	okay the price of the hotel is 1305 hkd per night .
CDNet trained on incTOD	the hotel is four_seasons_hotel_hong_kong hkd per night .
DKAF + CDNet trained on incTOD	the hotel is 1305 hkd per night .

Table 10: Example to compare the response generation performance of CDNet trained on *inc*-TOD with and without *DKAF*