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 Dialog-KB Arbitration Framework (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.
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 \textit{La Margherita} to the KB for $d_1$). $DKAF$ deletes, from the KB, rows that are misaligned with the dialog (e.g., removing \textit{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 \textit{inc-bAbI} and \textit{inc-TOD} respectively. Both \textit{inc-bAbI} and \textit{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)
We first describe the task of learning an end-to-end TOD system. We denote a dialog between user $u$ and agent $a$ as $d = [u^1_1, u^1_2, u^2_2, ..., u^m_m]$ 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 $\hat{a}^j$ given dialog history $[u^1_1, u^1_2, u^2_2, ..., u^r_r]$ 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 $K_T$ 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 $\{d_j, K_j\}_{j=1}^N$. 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 $e_1, e_2 \in d^i$, 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 $K_T^{j^{\text{post insertion}}}$.

4.2 Row Deletion (RD)

RD model predicts whether a row $\rho$ from KB $K$ (mis)aligns with a given dialog $d$. Here, $\rho$ 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 Klapra, 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

\[ \text{can be identified by NER, though in this work, we assume this is known} \]
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 given a training dialog. Figure (bottom) shows DKAF in action with KB updates from row insertion, row deletion and row completion to training KB. (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 \((d, K, \rho)\) and takes an action \(a \in \{0, 1\}\) where \(a = 0\) means \(\rho\) is misaligned with \(d\). Given reward function \(R_a(d, K, \rho)\), RL objective for training RD is

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

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 \(i^{th}\) 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 \(<mask>\) token. Let \(E_a\) be the set of entities occurring in agent utterances in the dialog. MEM objective is then to maximize following likelihood

\[
\mathcal{L}(d, K) = \prod_{e \in E_a} P(e|H_e, K) \tag{1}
\]

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 C.4.

We train both MEM and RD models using \(\{d_j, K_{ri}^j\}_{j=1}^{N}\). 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 \((d_j, K_{ri}^j, \rho)\) states from each \(d_j\) to obtain snapshot \(K_{rdj}^j\) 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
daims 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 $d$ with position indicators for $e_s$ and encodes re-
sulting dialog using dialog encoder. Similar to 4.2,
it computes KB entity features using KB encoder
and reasons over them using memory network. Fi-
ally, 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

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

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

5 Experimental Setup

5.1 Datasets Construction

Existing TOD datasets make a simplistic assump-
tion 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 system-
atically induce dialog-KB inconsistencies in two
existing TOD datasets, namely bAbI dialog (Bor-
des 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 gener-
ate 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 enti-
ties 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 to-
gether 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 ran-
don 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 avail-
ability is subject to temporal factors like day of the
week and time of the day. Moreover, restaur-
ants 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

²https://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 \( \Ktilde_j \). We use the \( \{d_j, \Ktilde_j\}_{j=1}^N \) 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
500 We compare training KB and temporarily KB for each dialog.
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
Figure 3: Figure shows the simulation pipeline used for generating datasets.

### Table 1: DKAF main Results

<table>
<thead>
<tr>
<th>Models</th>
<th>inc-bAbI</th>
<th>inc-TOD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Response Acc.</td>
<td>Dialog Acc.</td>
</tr>
<tr>
<td>CDNet</td>
<td>96.42 ± 0.25</td>
<td>64.20 ± 2.55</td>
</tr>
<tr>
<td>CDNet + DKAF</td>
<td>98.78 ± 0.05</td>
<td>86.37 ± 0.30</td>
</tr>
<tr>
<td>CDNet + Oracle</td>
<td>99.34 ± 0.17</td>
<td>92.30 ± 3.25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relevance</th>
<th>Naturalness</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDNet</td>
<td>3.08</td>
</tr>
<tr>
<td>CDNet + DKAF</td>
<td>3.31</td>
</tr>
</tbody>
</table>

Table 2: Human Evaluation on inc-bAbI

For inc-TOD, 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 (0-5). 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 performance of DKAF. For both inc-bAbI and inc-TOD, excluding row insertion leads to signif-
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.

### 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.

### Table 3: Ablation Results

<table>
<thead>
<tr>
<th>Configurations</th>
<th>inc-bAbI Dialog Acc. ±</th>
<th>inc-bAbI KB Ent. F1 ±</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDNet + DKAF</td>
<td>86.37 ± 0.25</td>
<td>0.68 ± 0.02</td>
</tr>
<tr>
<td>CDNet + DKAF - RI</td>
<td>73.67 ± 1.08</td>
<td>0.64 ± 0.02</td>
</tr>
<tr>
<td>CDNet + DKAF - RD</td>
<td>75.40 ± 3.89</td>
<td>0.69 ± 0.00</td>
</tr>
<tr>
<td>CDNet + DKAF - RC</td>
<td>81.87 ± 5.69</td>
<td>0.68 ± 0.02</td>
</tr>
<tr>
<td>CDNet</td>
<td>64.20 ± 2.08</td>
<td>0.64 ± 0.02</td>
</tr>
</tbody>
</table>

| Table 3: Ablation Results |

---

### 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.
A Data statistics

<table>
<thead>
<tr>
<th></th>
<th>inc-bAbI</th>
<th>inc-TOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Dialogs</td>
<td>1000</td>
<td>1614</td>
</tr>
<tr>
<td>Val Dialogs</td>
<td>1000</td>
<td>169</td>
</tr>
<tr>
<td>Test Dialogs</td>
<td>1000</td>
<td>251</td>
</tr>
</tbody>
</table>

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 \( d = [u_1^a, u_1^b, u_2^b, ..., u_m^a, u_m^b] \) be a given dialog with \( m \) turns where \( u_i \) is \( i^{th} \) utterance in the dialog. Let \( u_i = [w_{i1}, w_{i2}, ..., w_{it}] \) where \( w_{it} \) is encoding for \( k^{th} \) token in \( u_i \) and \( li \) 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

\[
[h_{i1}, h_{i2}, ..., h_{it}] = BiGRU([w_{i1}, w_{i2}, ..., w_{it}])
\]

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

\[
h_i = \sum_{k=1}^{l_i} \alpha_k h_{ik}
\]

\[
\alpha_k = \text{softmax}(g_a(h_{ik}))
\]

where \( g_a(h_{ik}) \) is a feed-forward network. Dialog level encoder takes \([h_1, h_2, ..., h_{2m}]\) as input and computes dialog feature vector \( c \) using Luong attention as

\[
[H_1, H_2, ..., H_{2m}] = GRU([h_1, h_2, ..., h_{2m}])
\]

\[
c = \sum_{i=1}^{2m} \beta_i H_i
\]

\[
\beta_i = \text{softmax}(g_d(H_i))
\]

where \( g_d \) is another feed forward network. Note that 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 \( G = (V, E, R) \) where \( V \) and \( E \) are set entities and relationships in KB respectively. \( R \) 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 \( Z_0 = \{ z_e^0 \}_{e \in V} \) of entity embeddings as input to the first r-GCN layer. \( l^{th} \) GCN layer updates the features for entity \( e \in V \) as

\[
z_e^l = \sigma \left( \sum_{r \in R} \sum_{e' \in N_e^r} W^{(l)} W^{(l-1)} z_e^{l-1} + W^{(l)} z_e^{l-1} \right)
\]

where \( N_e^r \) 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 \( \sigma \) is ReLU activation function. We use \( Z = \{ z_e \}_{e \in V} \) 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 \( Z \) 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
\[
o(t) = \sum_{k=1}^{|||} \gamma_k z_k
\]
\[
\gamma_k = \text{softmax}(g^t(z_k||q^{t-1})))
\]
\[
q(t) = q^{t-1} + o(t)
\]
where \(g^t\) is a feed-forward network at \(t^{th}\) hop and \(|||\) is concatenation operator. Output of the memory network is final query vector \(q = q^{(b)}\).

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 \(u_t\) in the dialog, RI model appends \(h_u\) with position vectors \(\text{pos}_i\) and \(\text{pos}_j\) 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 \(c\).

RI model concatenates dialog features \(c\) 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 \((d, K, \rho)\), RD model computes dialog features and KB features using dialog encoder and KB encoder respectively. It computes encoding for the input \(\rho\) as \(z_\rho = \sum_{e\in\rho} z_e\). Finally, it sets initial query \(q^0 = c\) and reasons over KB entity encoding using memory network to get refined final query vector \(q\). Finally, it concatenates vectors \(q, z_\rho\) 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 \(q^0 = f(c|h_{e_s})\) where \(f\) is a feed-forward layer as input to memory network.

Finally, it combines memory network output \(q\) with entity features \(z_{e_s}\) 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. \(\lambda\) 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 \(c\) and <mask> 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.

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
Table 5: GLMP Hyperparameters

<table>
<thead>
<tr>
<th>Models</th>
<th>inc-bAbI</th>
<th>inc-TOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLMP + Oracle</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>GLMP</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>GLMP + DKAF</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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 hour (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
<table>
<thead>
<tr>
<th>Models</th>
<th>\textit{inc-bAbI}</th>
<th>\textit{inc-TOD}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Response Acc.</td>
<td>Dialog Acc.</td>
</tr>
<tr>
<td>GLMP + Oracle</td>
<td>98.51 ± 0.16</td>
<td>81.67 ± 1.92</td>
</tr>
<tr>
<td>GLMP</td>
<td>97.80 ± 0.06</td>
<td>72.03 ± 1.13</td>
</tr>
<tr>
<td>GLMP + \textit{DKAF}</td>
<td>98.25 ± 0.15</td>
<td>78.63 ± 1.72</td>
</tr>
</tbody>
</table>

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 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

<table>
<thead>
<tr>
<th>r_name</th>
<th>r_phone</th>
<th>r_cuisine</th>
<th>r_address</th>
<th>r_location</th>
<th>r_number</th>
<th>r_price</th>
<th>r_rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>resto_london Moderate british 6stars</td>
<td>resto_london Moderate british 6stars_phone</td>
<td>british</td>
<td>resto_london Moderate british 6stars_address</td>
<td>london</td>
<td>four</td>
<td>moderate</td>
<td>6</td>
</tr>
<tr>
<td>resto_london Moderate british 6stars</td>
<td>resto_london Moderate british 6stars_phone</td>
<td>british</td>
<td>resto_london Moderate british 6stars_address</td>
<td>london</td>
<td>four</td>
<td>moderate</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r_name</th>
<th>r_phone</th>
<th>r_cuisine</th>
<th>r_address</th>
<th>r_location</th>
<th>r_number</th>
<th>r_price</th>
<th>r_rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>resto_london Moderate british 4stars</td>
<td>resto_london Moderate british 4stars_phone</td>
<td>british</td>
<td>resto_london Moderate british 4stars_address</td>
<td>london</td>
<td>four</td>
<td>moderate</td>
<td>4</td>
</tr>
<tr>
<td>resto_london Moderate british 4stars</td>
<td>resto_london Moderate british 4stars_phone</td>
<td>british</td>
<td>resto_london Moderate british 4stars_address</td>
<td>london</td>
<td>four</td>
<td>moderate</td>
<td>6</td>
</tr>
<tr>
<td>resto_london Moderate british 8stars</td>
<td>resto_london Moderate british 8stars_phone</td>
<td>british</td>
<td>resto_london Moderate british 8stars_address</td>
<td>london</td>
<td>four</td>
<td>moderate</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r_name</th>
<th>r_phone</th>
<th>r_cuisine</th>
<th>r_address</th>
<th>r_location</th>
<th>r_number</th>
<th>r_price</th>
<th>r_rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>resto_london Moderate british 4stars</td>
<td>resto_london Moderate british 4stars_phone</td>
<td>british</td>
<td>resto_london Moderate british 4stars_address</td>
<td>london</td>
<td>four</td>
<td>moderate</td>
<td>4</td>
</tr>
<tr>
<td>resto_london Moderate british 4stars</td>
<td>resto_london Moderate british 4stars_phone</td>
<td>british</td>
<td>resto_london Moderate british 4stars_address</td>
<td>london</td>
<td>four</td>
<td>moderate</td>
<td>6</td>
</tr>
<tr>
<td>resto_london Moderate british 1stars</td>
<td>resto_london Moderate british 1stars_phone</td>
<td>british</td>
<td>resto_london Moderate british 1stars_address</td>
<td>london</td>
<td>four</td>
<td>moderate</td>
<td>2</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>location</th>
<th>name</th>
<th>number_of_rooms</th>
<th>price_per_night</th>
<th>rating</th>
<th>ref_number</th>
<th>stars</th>
</tr>
</thead>
<tbody>
<tr>
<td>central_district</td>
<td>the_landmark_mandarin_oriental_hong_kong</td>
<td>5</td>
<td>1670</td>
<td>10</td>
<td>300pwrthi</td>
<td>5</td>
</tr>
<tr>
<td>hong_kong_island</td>
<td>the_upper_house</td>
<td>10</td>
<td>1381</td>
<td>10</td>
<td>lq9ic93b</td>
<td>5</td>
</tr>
<tr>
<td>tsim_sha_tsui</td>
<td>the_peninsula_hong_kong</td>
<td>7</td>
<td>1222</td>
<td>9</td>
<td>a4k516b</td>
<td>5</td>
</tr>
<tr>
<td>new_territories</td>
<td>tai_o_heritage_hotel</td>
<td>9</td>
<td>1921</td>
<td>8</td>
<td>ghkpm465</td>
<td>4</td>
</tr>
<tr>
<td>tsim_sha_tsui</td>
<td>soravit_on_granville</td>
<td>5</td>
<td>1514</td>
<td>8</td>
<td>tx66hpkf</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>location</th>
<th>name</th>
<th>number_of_rooms</th>
<th>price_per_night</th>
<th>rating</th>
<th>ref_number</th>
<th>stars</th>
</tr>
</thead>
<tbody>
<tr>
<td>central_district</td>
<td>four_seasons_hotel_hong_kong</td>
<td>5</td>
<td>1305</td>
<td>10</td>
<td>amoxngso</td>
<td></td>
</tr>
<tr>
<td>hong_kong_island</td>
<td>the_upper_house</td>
<td>10</td>
<td>1670</td>
<td>10</td>
<td>300pwrthi</td>
<td>5</td>
</tr>
<tr>
<td>tsim_sha_tsui</td>
<td>the_peninsula_hong_kong</td>
<td>7</td>
<td>1381</td>
<td>10</td>
<td>lq9ic93b</td>
<td>5</td>
</tr>
<tr>
<td>new_territories</td>
<td>tai_o_heritage_hotel</td>
<td>9</td>
<td>1222</td>
<td>9</td>
<td>a4k516b</td>
<td>5</td>
</tr>
<tr>
<td>tsim_sha_tsui</td>
<td>soravit_on_granville</td>
<td>5</td>
<td>1921</td>
<td>8</td>
<td>ghkpm465</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 8: An Example to demonstrate the correction made by **DKAF** to an inconsistent sample of *inc-TOD* dataset
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 performance of CDNet trained on inc-bAbI with and without DKAF
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

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

<table>
<thead>
<tr>
<th>location</th>
<th>name</th>
<th>number_of_rooms</th>
<th>phone_number</th>
<th>price_level</th>
<th>price_per_night</th>
<th>rating</th>
<th>ref_number</th>
<th>stars</th>
</tr>
</thead>
<tbody>
<tr>
<td>central_district</td>
<td>four_seasons_hotel_hong_kong</td>
<td>10</td>
<td>852_5706_6379</td>
<td>expensive</td>
<td>1305</td>
<td>10</td>
<td>amoNSagoo</td>
<td>5</td>
</tr>
<tr>
<td>central_district</td>
<td>the_landmark_mandarin_oriental_hong_kong</td>
<td>5</td>
<td>852_6550_4214</td>
<td>expensive</td>
<td>1670</td>
<td>10</td>
<td>300pwfBt</td>
<td>5</td>
</tr>
<tr>
<td>hong_kong_island</td>
<td>jw_marriott_hotel_hong_kong</td>
<td>10</td>
<td>852_7885_6633</td>
<td>expensive</td>
<td>2210</td>
<td>9</td>
<td>sY9h2x3</td>
<td>5</td>
</tr>
<tr>
<td>tsim_sha_tsui</td>
<td>the_peninsula_hong_kong</td>
<td>7</td>
<td>852_0352_0315</td>
<td>expensive</td>
<td>1222</td>
<td>9</td>
<td>a4k516l</td>
<td>5</td>
</tr>
<tr>
<td>tsim_sha_tsui</td>
<td>house_1881</td>
<td>2</td>
<td>852_0071_5333</td>
<td>expensive</td>
<td>1895</td>
<td>8</td>
<td>swm2n2uu</td>
<td>5</td>
</tr>
<tr>
<td>new_territories</td>
<td>horizon_suite_hotel</td>
<td>4</td>
<td>852_6964_6875</td>
<td>expensive</td>
<td>1594</td>
<td>8</td>
<td>7bdkw3y</td>
<td>0</td>
</tr>
<tr>
<td>tsim_sha_tsui</td>
<td>maharaja_guesthouse</td>
<td>10</td>
<td>852_0723_8650</td>
<td>expensive</td>
<td>1286</td>
<td>5</td>
<td>5wdd2n3l</td>
<td>1</td>
</tr>
</tbody>
</table>