Autoregressive Entity Generation for End-to-End Task-Oriented Dialog

Guanhuan Huang, Xiaojun Quan
School of Computer Science and Engineering
Sun Yat-sen University, Guangzhou, China
huanggh25@mail2.sysu.edu.cn
quanxj3@mail.sysu.edu.cn

Qifan Wang
Meta AI
Menlo Park, CA, USA
wqfcr@fb.com

Abstract

Task-oriented dialog (TOD) systems often require interaction with an external knowledge base to retrieve necessary entity (e.g., restaurant) information to support the response generation. Most current end-to-end TOD systems either retrieve the KB information explicitly or embed it into model parameters for implicit access. While the former approach demands scanning the KB at each turn of response generation, which is inefficient when the KB scales up, the latter approach shows higher flexibility and efficiency. In either approach, the systems may generate a response with conflicting entity information. To address this issue, we propose to generate the entity autoregressively first and leverage it to guide the response generation in an end-to-end system. To ensure entity consistency, we impose a trie constraint on entity generation. We also introduce a logit concatenation strategy to facilitate gradient backpropagation for end-to-end training. Experiments on MultiWOZ 2.1 single and CAMREST show that our system can generate more high-quality and entity-consistent responses.

1 Introduction

Task-oriented dialog (TOD) systems (Young et al., 2013; Budzianowski et al., 2018) have become prominent and drawn much attention from both academia and industries. Their mission is to help users accomplish specific tasks such as booking restaurants and reserving hotels through natural language conversations, where an external knowledge base (KB) is usually needed to support the generation of a system response. For example, when trying to recommend a restaurant, they will retrieve its address from the KB and generate a response.

Many recent state-of-the-art TOD systems (Mehri et al., 2019; Hosseini-Asl et al., 2020; Li et al., 2021) take a pipeline route that decomposes the task into modules that rely on intermediate annotations such as belief state and dialog act for

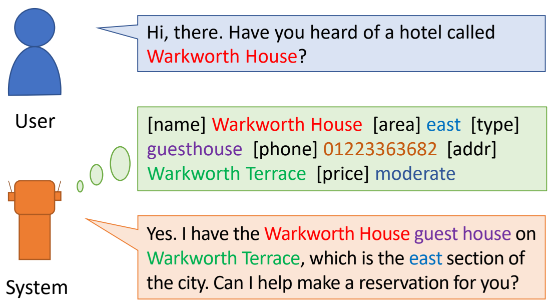
Hi, there. Have you heard of a hotel called Warkworth House?
User
[name] Warkworth House [area] east [type] guesthouse [phone] 01223363682 [addr] Warkworth Terrace [price] moderate
System
Yes. I have the Warkworth House guest house on Warkworth Terrace, which is the east section of the city. Can I help make a reservation for you?

Figure 1: An example to show that task-oriented dialog systems need to retrieve information (middle) from a knowledge base (KB) to generate a qualified system response. Entity values in the KB are color-highlighted.

supervision. These modules can be trained individually and assembled into a dialog system, mitigating the difficulty of generating a desired response directly from the dialog context and user utterance. Another motivation for the pipeline architecture is the necessity of querying KB with belief state, as shown in Figure 1, which would otherwise be non-trivial to realize in an end-to-end system. However, these annotations have to be crafted by human annotators, which is hardly realistic in practical scenes such as intelligent customer services where huge amounts of unannotated natural language conversations are accumulated. Besides, errors made in upstream modules may be propagated to downstream modules if they are not trained jointly.

There are mainly two approaches to eliminating the reliance on intermediate annotations and generating system response in an end-to-end manner. First, entity information in the KB can be accessed by soft attention (Madotto et al., 2018; Reddy et al., 2019; Qin et al., 2020). To this end, a memory network is usually used to encode the KB, and attention and pointer are then utilized to retrieve entity information from the memory. These attention-based methods tend to become cumbersome when the KB scales up. Second, the KB information can be stored in model parameters to avoid direct interaction with the KB during response generation (Madotto et al., 2020). This is partly motivated by the observation that pre-trained models such as BERT (Devlin et al., 2019) can carry certain relational and factual knowledge (Petroni et al., 2019). To embed the KB into model parameters, this approach first augments the original training set with KB entries and then encodes the training samples with a powerful encoder.

Despite the success in end-to-end TOD systems, one of the remaining problems is entity inconsistency during response generation (Qin et al., 2019), which means that the systems usually generate conflicting entity information in system responses. For example, they may generate a response "`Gourmet Kitchen is an Italian restaurant`" while `Gourmet Kitchen` is actually a `North American` restaurant. In this work, we aim to address this issue more scalably in our end-to-end TOD system. Following GPT-KE (Madotto et al., 2020), we first insert the KB into natural language dialogs by data augmentation, so that the KB can be embedded into model parameters whose size does not scale with the KB. Then, we predict the entity that will appear in the response autoregressively. To avoid generating an inconsistent entity, we impose a trie constraint on the decoding to ensure that the generated entity truly belongs to the KB. The generated entity is taken as an extra input to generate an entity-consistent system response. Besides, since tokens in the entity are integers, which hinders gradient backpropagation, we propose logit concatenation to allow for end-to-end training.

We evaluate our system on MultiWOZ 2.1 single (Budzianowski et al., 2018) and CAMREST (Wen et al., 2017), which are two task-oriented dialog benchmarks widely used in the literature. Experimental results show that it compares favorably with all the baselines. Particularly, it outperforms GPT-KE, a strong end-to-end TOD system that we follow, by a large margin. By ablation studies, we demonstrate that autoregressive entity generation assists in producing entity-consistent system responses in an end-to-end manner.

To our best knowledge, this is the first work that attempts to alleviate the entity inconsistency problem in TOD systems by generating the entity first and taking it as an input for response generation. The system can be trained end-to-end without accessing external KBs during response generation.

2 Related Work

End-to-end task-oriented dialog systems have drawn increasing attention in recent years. In one line of work, researchers propose to train the modules of a pipeline system jointly in an end-to-end framework, though they still require intermediate annotations for supervision. Among these works, SimpleTOD (Hosseini-Asl et al., 2020), SOLOIST (Peng et al., 2020), and UBAR (Yang et al., 2021) attempt to concatenate the dialog history, user utterance, belief state, dialog act, and system response into a long sequence, which is then modeled by a sequence-to-sequence generation model. HyKnow (Gao et al., 2021) extends the belief state to handle both structured and unstructured knowledge and trains the dialog state tracking and response generation modules jointly. Nevertheless, these systems are not the end-to-end solutions we pursue in this work since they still need intermediate annotations.

There are mainly two approaches to implementing intermediate annotations free end-to-end TOD systems. First, entity information in the KB can be accessed by soft attention. Mem2Seq (Madotto et al., 2018) combines the ideas of multi-hop attention over memory and a pointer network to incorporate KB information. Wen et al. (2018) proposed to compute a dialogue state representation from the dialog history and use it to interact with KB representations to retrieve entity information for response generation. GLMP (Wu et al., 2019) encodes the representations of dialog history and structural KB with a memory network and then passes the result to a decoder for response generation. DF-Net (Qin et al., 2020) includes a dynamic fusion module to generate a fused representation that explicitly captures the correlation between domains and uses it to query the KB. When the KB scales up, however, attention-based methods become less efficient.

Second, the KB information can be stored in model parameters to avoid further interaction with the KB during response generation. The motivation comes from the observation that pre-trained language models such as BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020) can already carry certain relational and factual knowledge (Petroni et al., 2019). GPT-KE (Madotto et al., 2020) is the seminal dialog system towards this goal. It first augments the training set with KB entries and then learns a response generation model from the augmented set in an end-to-end fashion, thus abandoning the KB during response generation.

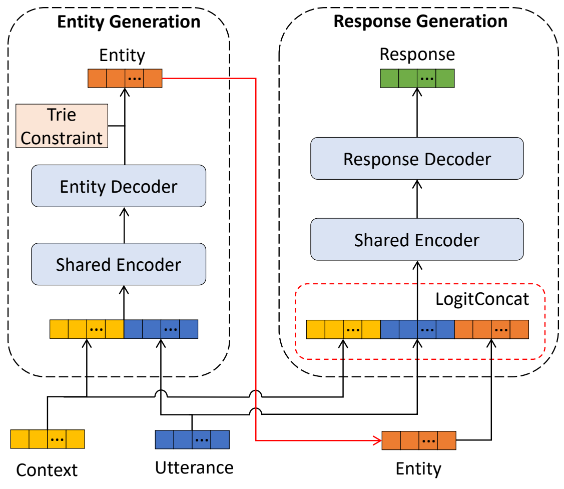

Figure 2: The architecture of our ECO system. The entity generation module takes the context and user utterance as input and generates a relevant entity. The response generation module takes the context, user utterance, and the generated entity as input to generate a system response. The two modules share the same encoder but have separate decoders. A trie constraint is imposed when generating the entity, and LogitConcat is proposed to facilitate end-to-end optimization.

Original dialog:	i am sorry but gourmet burger kitchen was the only north american restaurant in the centre area .
Template:	i am sorry but [name] was the only [food] restaurant in the [area] area .
New dialog:	i am sorry but da vinci pizzeria was the only italian restaurant in the north area .

Figure 3: An example to show how to construct a template from the training sample and generate a new sample from the template. Attributes are color-highlighted.

3 Methodology

As shown in Figure 2, our Entity-COnsistent end-to-end (ECO) task-oriented dialog system begins by embedding the KB into training dialogs (§3.2). Following GPT-KE (Madotto et al., 2020), we augment the original training set with KB entries and abandon the KB afterward. Unlike GPT-KE, which conducts data augmentation in data pre-processing before training, ECO conducts augmentation for each batch of training samples, which reduces the size of augmented training samples while maintaining high coverage of the KB. We then predict the entity (§3.3) that will appear in the response and incorporate it into response generation (§3.4) to ensure entity consistency, where LogitConcat is proposed to facilitate end-to-end optimization.

3.1 Notations

Given a training set $\mathcal{D}_{tr} = \{D_1, D_2, \dots, D_N\}$ of dialogs, where $D_i = \{U_{i,1}, R_{i,1}, \dots, U_{i,T}, R_{i,T}\}$ contains T turns of user utterance and system response, we denote the conversational context of the t-th turn in dialog D_i as $C_{i,t} = \{U_{i,1}, R_{i,1}, \dots, U_{i,t-1}, R_{i,t-1}\}$. A structured knowledge base is given in the form of a set of entities $KB = \{E_1, E_2, \dots, E_M\}$, each of which is represented as a sequence $E_i = \{a_1, v_{i,1}, a_2, v_{i,2}, \dots, a_K, v_{i,K}\}$ in which a_j and $v_{i,j}$ denote the jth attribute and its value for entity E_i, respectively. For simplicity, we assume each turn of dialog only relates to one entity and reformulate it as $\{U_{i,t}, R_{i,t}, E_{i,t}\}$. A user goal (Schatzmann et al., 2007) is defined for each dialog as $G_i = (G_{i,c}, G_{i,r})$, where $G_{i,c}$ specifies the constrained information (e.g., {location=`center`, price=`cheap`}) and $G_{i,r}$ denotes the required information (e.g., `address`, `name`).

3.2 Knowledge Base Embedding

To embed the KB into the training set, we first extract all mentioned entity values in both user utterances and ground truth responses based on given span annotations in the original training set. Then, we match entity values with the KB to identify which entity is mentioned in the current turn of conversation. Templates are then constructed by replacing entity-related tokens in the conversation with special attribute placeholders. For example, `north american` in Figure 3 is replaced with the corresponding attribute placeholder `[food]`. This template generation function is denoted as DELEX$(\cdot)$, which is used to generate a set $\mathcal{D}_{tm}$ of templates from the original training set $\mathcal{D}_{tr}$:

$$\mathcal{D}_{tm} = \text{DELEX}(\mathcal{D}_{tr}). \tag{1}$$

Next, we generate new dialog samples with the templates in $\mathcal{D}_{tm}$. We refer to this process as data augmentation. To begin with, we obtain a set of KB entities, $\mathcal{G}_{mt} = \{E_1, E_2, \dots, E_G\}$, that match the predefined user goals. Then, we randomly select an entity E_i from $\mathcal{G}_{mt}$ and replace the placeholders with the corresponding values of E_i. For instance, we replace `[food]` and `[area]` in Figure 3 with `italian` and `north`, respectively. The function of generating samples from templates is defined as RELEX$(\cdot)$, which is executed P times to insert P entities, producing a new set $\mathcal{D}_{au}$:

$$\mathcal{D}_{au} = \bigcup_{p=1}^{P} \text{RELEX}(\mathcal{D}_{tm}). \tag{2}$$

Note that usually only a subset of samples in $\mathcal{D}_{tr}$ can successfully match with entities in KB during data augmentation, making $\mathcal{D}_{au}$ not cover all the samples of $\mathcal{D}_{tr}$. For this reason, we join $\mathcal{D}_{tr}$ and $\mathcal{D}_{au}$ to get our final training set $\mathcal{D}_{fn}$.

$$\mathcal{D}_{fn} = \mathcal{D}_{tr} \bigcup \mathcal{D}_{au} \tag{3}$$

The selected entities during the above augmentation process are treated as ground truth entities for the corresponding dialog samples. This means that only the samples in $\mathcal{D}_{au}$ have entity labels while the samples in $\mathcal{D}_{tr}$ do not. Since all the placeholders in the templates are replaced with values from the same entity, this data augmentation process ensures that the augmented training samples contain consistent entity information.

3.3 Autoregressive Entity Generation

To predict the entity that will appear in the response, we propose to generate it autoregressively. For the sake of brevity, we use C_t and U_t to represent the current dialog context and user utterance, respectively. Then, we concatenate C_t and U_t, encode them into a vector representation, and take it as an input for entity generation:

$$\mathbf{g}_t = \mathrm{Enc}(\mathrm{Emb}([C_t; U_t])), \tag{4}$$

where $\mathrm{Emb}(\cdot)$ is the embedding function implemented by a global embedding matrix $\mathbf{W}_e$. $\mathrm{Enc}(\cdot)$ denotes the encoder which is shared with the response generation module (§3.4).

To generate an entity $\hat{E}_t$ autoregressively, the decoder iteratively predicts a token $\hat{e}_{t,k}$ based on the already generated sequence $\hat{E}_{t,<k}$ and vector representation $\mathbf{g}_t$:

$$\hat{P}(\hat{e}_{t,k}) = \mathrm{Dec}_e(\hat{e}_{t,k} | \hat{E}_{t,<k}, \mathbf{g}_t). \tag{5}$$

Since the gold entities of the samples in $\mathcal{D}_{au}$ are known, the cross-entropy loss of entity generation on $\mathcal{D}_{au}$ is defined as:

$$\mathcal{L}_{en} = \sum_{D \in \mathcal{D}_{au}} \sum_{E_t \in D} \mathrm{CELoss}(\hat{E}_t, E_t), \tag{6}$$

where E_t denotes the ground truth entity for the t-th turn of conversation.

For the samples in $\mathcal{D}_{tr}$, which have no entity labels, we do not calculate their loss during entity generation, but instead calculate their loss in response generation (§3.4) to realize end-to-end optimization like DualTKB (Dognin et al., 2020).

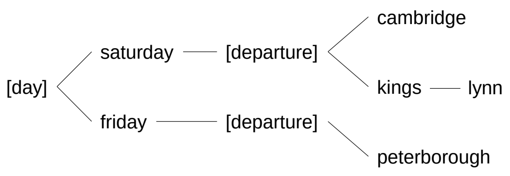

Figure 4: A trie with three entity sequences: `[day] saturday [departure] cambridge`, `[day] saturday [departure] kings lynn`, and `[day] friday [departure] peterborough`.

3.3.1 Trie Constraint

Inspired by GENRE (Cao et al., 2021), we construct a trie structure (a prefix tree) to ensure the generated entity truly belongs to the KB. For each entity in the KB, we construct a sequence as follows. For each value in an entity, we put its attribute placeholder to precede it and concatenate all pairs of attribute and value to form a sequence such as `[name] cityroomz [area] centre [type] hotel`. As depicted in Figure 4, a node in the trie denotes a token, and its child nodes denote all the succeeding tokens.

When decoding the k-th token $\hat{e}_{t,k}$ during the generation of entity $\hat{E}_t$, we have the decoded sequence $\hat{E}_{t,<k} = \{\hat{e}_{t,1}, \hat{e}_{t,2}, \ldots, \hat{e}_{t,k-1}\}$ in hand and walk through the trie along the path of $\hat{E}_{t,<k}$ to generate the next token. We use $\mathcal{E}_{t,k}$ to represent the set of possible tokens at this time step and re-compute $\hat{P}(\hat{e}_{t,k})$ as:

$$P(\hat{e}_{t,k}) = \begin{cases} \frac{\hat{P}(\hat{e}_{t,k})}{Z}, & \hat{e}_{t,k} \in \mathcal{E}_{t,k} \\ 0, & \text{else} \end{cases} \tag{7}$$

where

$$Z = \sum_{\hat{e}_{t,k} \in \mathcal{E}_{t,k}} \hat{P}(\hat{e}_{t,k}). \tag{8}$$

Since only tokens from $\mathcal{E}_{t,k}$ have non-zero probabilities in $P(\hat{e}_{t,k})$, the model always samples a token from $\mathcal{E}_{t,k}$. Therefore, the generated entity is guaranteed to be valid constantly.

3.4 Response Generation

During training, for each sample in $\mathcal{D}_{au}$, the model generates a response based on the context, user utterance, and the corresponding ground truth entity by concatenating and encoding them with the shared encoder defined in Eq. (4):

$$\mathbf{h}_t = \mathrm{Enc}(\mathrm{Emb}([C_t; U_t; E_t])). \tag{9}$$

For each sample in $\mathcal{D}_{tr}$ which has no ground truth entity label, the generated entity is used:

$$\mathbf{h}_t = \mathrm{Enc}(\mathrm{Emb}([C_t; U_t; \hat{E}_t])). \tag{10}$$

The response decoder then takes $\mathbf{h}_t$ as input and generates the response $\hat{R}_t$ token by token as:

$$P(\hat{r}_{t,k}) = \mathrm{Dec}_r(\hat{r}_{t,k}|\hat{R}_{t,<k}, \mathbf{h}_t). \quad (11)$$

The cross-entropy loss is calculated between the generated response $\hat{R}_t$ and the ground truth response R_t:

$$\mathcal{L}_{re} = \sum_{D \in \mathcal{D}_{fn}} \sum_{R_t \in D} \mathrm{CELoss}(\hat{R}_t, R_t). \quad (12)$$

3.4.1 Logit Concatenation

For those samples in $\mathcal{D}_{tr}$, since the tokens in each generated entity are integers, the gradients of response generation cannot be directly passed to the encoder during training. To address this, we modify Eq. (10) and input the distributions of generated entity tokens to the encoder. Specifically, for the k-th token $\hat{e}_{t,k}$ of a generated entity $\hat{E}_t$, its output distribution $P(\hat{e}_{t,k})$ over vocabulary from the entity decoder is first computed using Eq. (7) and then used to approximate $\hat{e}_{t,k}$ for gradient propagation. $P(\hat{e}_{t,k})$ can be encoded as:

$$\mathbf{h}_{t,k} = P(\hat{e}_{t,k})\mathbf{W}_e^T, \quad (13)$$

where $\mathbf{W}_e$ is the global embedding matrix introduced above.

If both $P(\hat{e}_{t,k})$ and $\mathbf{W}_e$ receive gradients during propagation, the training may collapse since it is much easier to update $\mathbf{W}_e$ than $P(\hat{e}_{t,k})$, which requires understanding the context and utterance to obtain relevant information. Therefore, we alter the equation by stopping gradients on $\mathbf{W}_e$:

$$\hat{\mathbf{h}}_{t,k} = P(\hat{e}_{t,k}) \cdot \mathrm{StopGrad}(\mathbf{W}_e^T), \quad (14)$$

$$\hat{\mathbf{h}}_t = \{\hat{\mathbf{h}}_{t,1}, \ldots, \hat{\mathbf{h}}_{t,|\hat{E}_t|}\}. \quad (15)$$

We use $\hat{\mathbf{h}}_t$ as the representation of entity $\hat{E}_t$ and concatenate it with the embeded context C_t and user utterance U_t, which is then encoded to replace Eq. (10) during training:

$$\mathbf{h}_t = \mathrm{Enc}(\mathrm{Emb}([C_t; U_t]); \hat{\mathbf{h}}_t]). \quad (16)$$

Since $P(\hat{e}_{t,k})$ is a distribution vector rather than an integer, gradients can be backpropagated to the encoder during training. At inference time, we take the generated entity tokens rather than $P(\hat{e}_{t,k})$ as input for response generation, as described in Eq. (10). This brings a gap between training and inference, which will be studied in Section 4.5.

3.5 Joint Training

The final system is trained by minimizing the sum of entity loss $\mathcal{L}_{en}$ and response loss $\mathcal{L}_{re}$:

$$\mathcal{L} = \mathcal{L}_{en} + \mathcal{L}_{re}. \quad (17)$$

4 Experiments

4.1 Dataset

We conduct experiments on MultiWOZ 2.1 single (Budzianowski et al., 2018) and CAMREST (Wen et al., 2017). CAMREST consists of one domain of Cambridge restaurant booking while MultiWOZ 2.1 single consists of five domains: `Attraction`, `Hotel`, `Restaurant`, `Taxi`, and `Train`. Following previous work (Qin et al., 2020; Madotto et al., 2020), we select only the dialogues which involves a single domain from MultiWOZ 2.1 to form the MultiWOZ 2.1 single dataset. We follow the same pre-processing and augmentation procedures as GPT-KE (Madotto et al., 2020). Note that not all dialogs in the original training set can be successfully used to generate templates due to the diversity of entity values. On MultiWOZ 2.1 single, 63/116/289/59 templates are respectively generated for domains `Attraction`/`Hotel`/`Restaurant`/`Train`, and no template is generated for the `Taxi` domain since MultiWOZ 2.1 single does not provide KB for this domain. On CAMREST, 161 templates are constructed for data augmentation.

Following previous works (Qin et al., 2020; Madotto et al., 2020), we adopt BLEU, Inform, Success, and F1 as the metrics to evaluate model performance on MultiWOZ 2.1 single, and employ BLEU, F1, and Success on CAMREST. Inform and Success are calculated based on the given user goal of a dialog session, and inconsistent entity information will lower the two metrics. Meanwhile, an overall score is also calculated: Score = BLEU + (Inform + Success)/2.

4.2 Measuring Entity Consistency

Measuring the entity consistency of a given system response remains a problem in task-oriented dialog systems. Qin et al. (2021) annotated three kinds of inconsistency by human experts on the KVRET (Eric et al., 2017) dataset, i.e., user query inconsistency, dialog history inconsistency, and knowledge base inconsistency. They then trained models as a form of automatic metrics to identify which kind of inconsistency appears in system responses.

	BLEU	**Inform**	**Success**	**Score**	**F1**	**Consistency**
Mem2Seq (Madotto et al., 2018)	6.60	-	-	-	21.62	-
DSR (Wen et al., 2018)	9.10	-	-	-	30.00	-
GLMP (Wu et al., 2019)	6.90	-	-	-	32.40	-
DF-Net (Qin et al., 2020)	9.40	-	-	-	35.10	-
GPT2 (Radford et al., 2019)	14.33	64.60	51.77	72.52	30.38	-
GPT-KE (Madotto et al., 2020)	**15.05**	72.57	64.16	83.42	39.58	54.46
BART-KE	12.80±0.22	70.94±2.05	61.36±2.12	78.95±2.05	39.31±0.22	52.96±0.48
ECO (ours)	12.61±0.20	**83.63±0.63**	**75.37±0.21**	**92.11±0.20**	**40.87±0.24**	**56.84±0.36**

Table 1: Main results on MultiWOZ. Scores of baselines except BART-KE are from original papers, and "-" denotes scores originally unavailable. BART-KE is the baseline implemented by replacing GPT-2 in GPT-KE with BART.

However, their method is trained on KVRET and may not be suitable to measure inconsistency on other datasets. Furthermore, the first few turns may include irrelevant entity information, such as providing several hotels for the user to choose, which makes it difficult to identify whether the generated response is dialog history consistent or not.

The above analysis motivates us to propose a new consistency metric that focuses on user query consistency and knowledge base consistency. It is a turn-level metric that requires all entity information in the user utterance and the system response to belong to the same entity in KB. To be specific, we first extract all entity information in the user utterance and the system response, and then search the knowledge base. If there is an entity that contains all the extracted information, this turn of conversation scores 1, and 0 otherwise. The final Consistency metric is calculated as the average score over all conversation turns. The method of extracting entity information from utterances and responses is the same as in calculating F1.

4.3 Experiment Settings

Different from GPT-KE, we use BART (Lewis et al., 2020) as our backbone model due to the limitation of computation power. We also replace GPT-2 (Radford et al., 2019) in GPT-KE with BART to form a new baseline, BART-KE. We set the max input sequence length to 256, the repeat times P in RELEX to 12, and the batch size to 12. Experiments are conducted on a single NVIDIA 2080ti and cost about 11G GPU RAM. We conduct ablation studies on MultiWOZ 2.1 single as it is a more challenging dataset with multiple domains of dialogs. For most variants of our method, we run 30 epochs and evaluate them per 5 epochs, saving a model checkpoint after each evaluation. We then select the best checkpoint based on model performance on the development set and finally report the test results. For the ablation setting of *w/ tr*, which lacks supervision from gold entity labels for training, we run 50 epochs to select the best.

	BLEU	**F1**	**Success**
KB-Trs	14.80	45.30	-
MLMN	13.61	54.85	-
BoSsNet	15.20	43.10	-
KBRet	**18.64**	55.76	62.03
GPT-KE	18.00	54.85	74.68
BART-KE	17.84±0.28	70.42±0.37	75.06±1.52
ECO (ours)	18.42±0.27	**71.56±0.39**	**78.77±1.85**

Table 2: Main results on CAMREST. KB-Trs (E. et al., 2019), MLMN (Reddy et al., 2019), BoSsNet (Raghu et al., 2019), KBRet (Qin et al., 2019), and GPT-KE (Madotto et al., 2020) are baselines for comparison.

4.4 Main Results

The overall results are shown in Table 1 and Table 2. We observe that ECO outperforms GPT-KE and other baselines by a large margin in all metrics except BLEU, showing that ECO can reach the user goals of this dialog dataset more effectively. The improvement of ECO over BART-KE suggests that ECO's success mainly comes from the model design rather than BART itself. Specifically, by generating an entity with trie constraint to help response generation, ECO obtains consistent entity information and improves entity consistency of generated response. On the other hand, we note that BART-based methods (BART-KE and ECO) achieve relatively lower BLEU scores than the GPT-2 family baselines (GPT-2 and GPT-KE) on MultiWOZ. The main reason should be that we do not post-train BART with language modeling objectives on the training set, which affects the fluency of generated responses, while responses in MultiWOZ are more diverse across domains.

	Percentage (%)	Inform	Success	F1
single inform	46.0	91.67±1.63	83.01±1.98	61.09±0.81
multi inform	54.0	76.78±1.55	68.85±1.77	31.03±0.73
single success	84.5	83.60±1.23	77.49±0.43	42.74±0.10
multi success	15.5	83.81±2.69	63.81±1.35	33.62±1.28
total	100.0	83.63±0.63	75.37±0.21	40.87±0.24

Table 3: Results of study on how multiple matched entities affect evaluation metrics, where single/multi inform/success refer to the situation with single/multiple matched entities when calculating Inform/Success, and Percentage (%) means the proportion of samples in the test set.

We also analyze why the improvement of F1 is much smaller than that of Inform and Success on MultiWOZ. Inform and Success are calculated based on user goals, and in some circumstances, there are multiple entities that match a user goal. However, only the one that is mentioned in the ground truth response is counted as correct in F1. Therefore, a large improvement on Inform and Success means ECO achieves user goals better, but the entity mentioned in the generated response may be different from the one in the ground truth. As shown in Table 3, over 50% of test samples have multiple matched entities when calculating Inform, and the percentage is 15.5% when calculating Success. Multiple matched entities reduce model performance on all metrics, especially on F1.

4.5 Ablation Studies

4.5.1 Training Datasets

Unlike samples in $\mathcal{D}_{tr}$, samples in $\mathcal{D}_{au}$ have ground truth entity labels, so their training objectives are different. We use ECO *w/ tr* to denote the training set that only contains samples from $\mathcal{D}_{tr}$ for end-to-end training, and use ECO *w/ au* to denote the training set that only contains samples from $\mathcal{D}_{au}$. As the results show in Table 4, ECO *w/ tr* has drops of 16.08 on Inform, 20.5 on Success, 4.42 on F1, and 4.41 on Consistency compared to ECO. Actually, without entity labels, the entity generation process is hard to converge, making response generation lack entity information as input and lowering model performance as a result.

On the other hand, ECO *w/ au* has less drops than ECO *w/ tr* compared to ECO on Inform, Success, and Consistency. However, the drops of ECO *w/ au* is more obvious on F1, which is caused by the fact that $\mathcal{D}_{au}$ does not includes the `Taxi` domain. These results demonstrate that the augmentation introduces more important KB information for response generation than the original training set.

4.5.2 Trie Constraint

In this work, the trie constraint is the key to guaranteeing entity consistency in a system response. Figure 5 presents an example of decoding an entity on the trie. Through filtering out non-kid nodes, the decoding path is restricted to a path on the trie. From Table 4, we note that ECO outperforms ECO *w/o trie* by 2.95 on Inform, 3.25 on Success, 1.06 on F1, and 0.53 on Consistency. Thus, we can conclude that the model generates more informative responses with the help of consistently generated entities, which accounts for the improvement.

4.5.3 Logit Concatenation

LogitConcat is proposed to enable backpropagation after concatenating the dialogue context, user utterance, and the generated entity when training on $\mathcal{D}_{tr}$. Without this component, the model parameters of the entity generator will not be updated. In Table 4, ECO shows a promising improvement of 5.31 on Inform, 4.87 on Success, 0.99 on F1, and 1.69 on Consistency over ECO *w/o LogitConcat*.

4.5.4 Gap between Training and Evaluation

During the evaluation, ECO uses the generated entity sequence as an input for response generation, which is different from the training phase that uses LogitConcat. To study the impact, we conduct an experiment that also applies LogitConcat during the evaluation. From the results (ECO *w/ LogitEval*) in Table 4, we observe obvious drops of ECO on Inform, Success, and Consistency. This phenomenon shows that applying LogitConcat during the evaluation weakens the consistency of the generated entities since the probability distribution in LogitConcat is not a valid entity from the KB.

4.5.5 The Number of Templates

To study how the number of templates affects model performance, we randomly select several subsets of templates from the whole template set.

	BLEU	Inform	Success	Score	F1	Consistency
GPT-KE	**15.05**	72.57	64.16	83.42	39.58	54.46
BART-KE	12.80±0.22	70.94±2.05	61.36±2.12	78.95±2.05	39.31±0.22	52.96±0.48
ECO	12.61±0.20	**83.63±0.63**	**75.37±0.21**	**92.11±0.20**	**40.87±0.24**	**56.84±0.36**
w/ au	8.94±0.06	79.20±2.26	56.34±0.55	76.71±0.94	30.38±1.67	55.49±0.33
w/ tr	11.21±0.37	67.55±4.41	54.87±4.38	72.42±4.07	36.45±1.17	52.43±1.89
w/o trie	12.40±0.36	80.68±0.91	72.12±1.25	88.80±0.81	39.81±0.25	56.31±0.62
w/o LogitConcat	12.52±0.11	78.32±0.96	70.50±2.46	86.93±1.64	39.88±0.40	55.15±0.84
w/ LogitEval	12.85±0.28	71.98±0.55	65.04±0.96	81.36±1.00	40.58±0.46	53.62±0.81

Table 4: Results of ablation studies. ECO *w/ au* represents the ECO variant trained on samples of $\mathcal{D}_{au}$, and ECO *w/ tr* represents the ECO variant trained on samples of $\mathcal{D}_{tr}$. ECO *w/o trie* means that ECO does not apply the trie constraint during entity generation, while ECO *w/o LogitConcat* means it does not apply LogitConcat during training. ECO *w/o StopGrad* represents the ECO variant that drops $\mathrm{StopGrad}$ in LogitConcat. ECO *w/ LogitEval* represents the ECO variant that applies LogitConcat during inference.

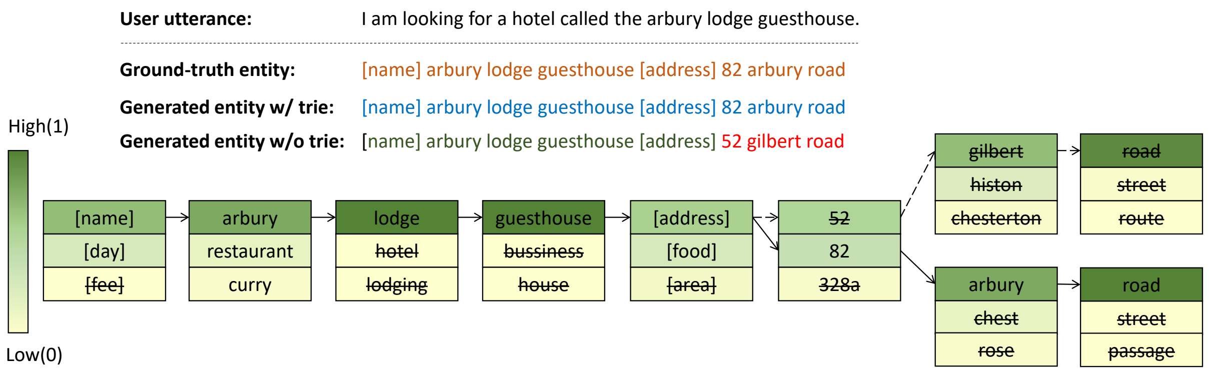
User utterance: I am looking for a hotel called the arbury lodge guesthouse.
Ground-truth entity: [name] arbury lodge guesthouse [address] 82 arbury road
Generated entity w/ trie: [name] arbury lodge guesthouse [address] 82 arbury road
Generated entity w/o trie: [name] arbury lodge guesthouse [address] 52 gilbert road
High(1)
Low(0)
[name]
[day]
~~[fee]~~
arbury
restaurant
curry
lodge
~~hotel~~
~~lodging~~
guesthouse
~~bussiness~~
~~house~~
[address]
[food]
~~[area]~~
~~52~~
82
~~328a~~
~~gilbert~~
~~histon~~
~~chesterton~~
~~road~~
~~street~~
~~route~~
arbury
~~chest~~
~~rose~~
road
~~street~~
~~passage~~

Figure 5: An example of decoding entity with the trie constraint. Darker backgrounds represent high decoding probabilities. The trie constraint filters out some tokens during decoding and results in a different decoding path, avoiding the red path which has a higher probability but generates inconsistent entity information.

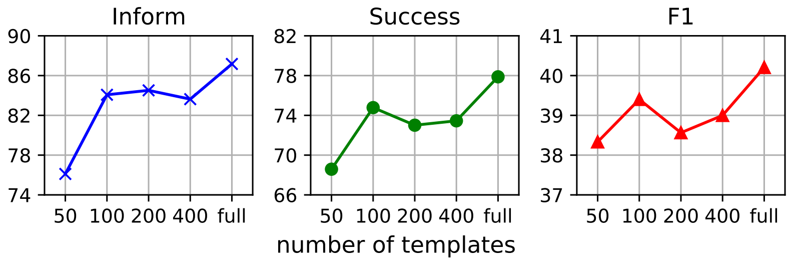

Figure 6: Ablation study of how the number of templates affects model performance on MuitiWOZ, where *full* means all the templates are used for knowledge base embedding.

As shown in Figure 6, the performance on all the metrics generally grows when the number of templates increases, but there are fluctuations when the number changes from 100 to 400.

5 Conclusion

We proposed an end-to-end task-oriented dialog system by encoding external knowledge into model parameters. To address entity inconsistency in system responses, we proposed to generate the entities first and took them as input to response generation. To ensure the generated entities are valid, a trie constraint was imposed on the generation, and a logit concatenation strategy was introduced to facilitate backpropagation for end-to-end training. Experiments demonstrate that this system can produce more high-quality and entity-consistent responses in an end-to-end manner. For future work, we will extend this system to handle multiple entities that are involved in each turn of conversation.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62176270) and the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (No. 2017ZT07X355).

References

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of EMNLP 2018*, pages 5016–5026. ACL.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *ICLR 2021*. OpenReview.net.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186. ACL.

Pierre L. Dognin, Igor Melnyk, Inkit Padhi, Cícero Nogueira dos Santos, and Payel Das. 2020. Dualtkb: A dual learning bridge between text and knowledge base. In *Proceedings of EMNLP 2020*, pages 8605–8616. ACL.

Haihong E., Wenjing Zhang, and Meina Song. 2019. Kb-transformer: Incorporating knowledge into end-to-end task-oriented dialog systems. In *2019 15th International Conference on Semantics, Knowledge and Grids (SKG)*, pages 44–48.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49, Saarbrücken, Germany. Association for Computational Linguistics.

Silin Gao, Ryuichi Takanobu, Wei Peng, Qun Liu, and Minlie Huang. 2021. Hyknow: End-to-end task-oriented dialog modeling with hybrid knowledge management. In *Findings of ACL/IJCNLP 2021*, pages 1591–1602. ACL.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *NeurIPS 2020*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of ACL 2020*, pages 7871–7880. ACL.

Yunhao Li, Yunyi Yang, Xiaojun Quan, and Jianxing Yu. 2021. Retrieve & memorize: Dialog policy learning with multi-action memory. In *Findings of ACL/IJCNLP 2021*, pages 447–459. ACL.

Andrea Madotto, Samuel Cahyawijaya, Genta Indra Winata, Yan Xu, Zihan Liu, Zhaojiang Lin, and Pascale Fung. 2020. Learning knowledge bases with parameters for task-oriented dialogue systems. In *Findings of EMNLP 2020*, pages 2372–2394. ACL.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of ACL 2018*, pages 1468–1478. ACL.

Shikib Mehri, Tejas Srinivasan, and Maxine Eskénazi. 2019. Structured fusion networks for dialog. In *Proceedings of SIGdial 2019*, pages 165–177. ACL.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020. SOLOIST: few-shot task-oriented dialog with A single pre-trained auto-regressive model. *CoRR*, abs/2005.05298.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *Proceedings of EMNLP-IJCNLP 2019*, pages 2463–2473. ACL.

Libo Qin, Yijia Liu, Wanxiang Che, Haoyang Wen, Yangming Li, and Ting Liu. 2019. Entity-consistent end-to-end task-oriented dialogue system with KB retriever. In *Proceedings of EMNLP-IJCNLP 2019*, pages 133–142. ACL.

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. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 2357–2367. Association for Computational Linguistics.

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 *Proceedings of ACL 2020*, pages 6344–6354. ACL.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Dinesh Raghu, Nikhil Gupta, and Mausam. 2019. Disentangling Language and Knowledge in Task-Oriented Dialogs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1239–1255, Minneapolis, Minnesota. Association for Computational Linguistics.

Revanth Reddy, Danish Contractor, Dinesh Raghu, and Sachindra Joshi. 2019. Multi-level memory for task oriented dialogs. In *Proceedings of NAACL-HLT 2019*, pages 3744–3754. ACL.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve J. Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Proceedings of NAACL-HLT 2007*, pages 149–152. ACL.

Haoyang Wen, Yijia Liu, Wanxiang Che, Libo Qin, and Ting Liu. 2018. Sequence-to-sequence learning for task-oriented dialogue with dialogue state representation. In *Proceedings of COLING 2018*, pages 3781–3792. ACL.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina Maria Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve J. Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL 2017*, pages 438–449. ACL.

Chien-Sheng Wu, Richard Socher, and Caiming Xiong. 2019. Global-to-local memory pointer networks for task-oriented dialogue. In *Proceedings of ICLR 2019*. OpenReview.net.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2021. Ubar: Towards fully end-to-end task-oriented dialog system with gpt-2. *Proceedings of AAAI 2021*, 35(16):14230–14238.

Steve J. Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proc. IEEE*, 101(5):1160–1179.