Decoupling Strategy and Surface Realization for Task-oriented Dialogues

Anonymous authors
Paper under double-blind review

Abstract

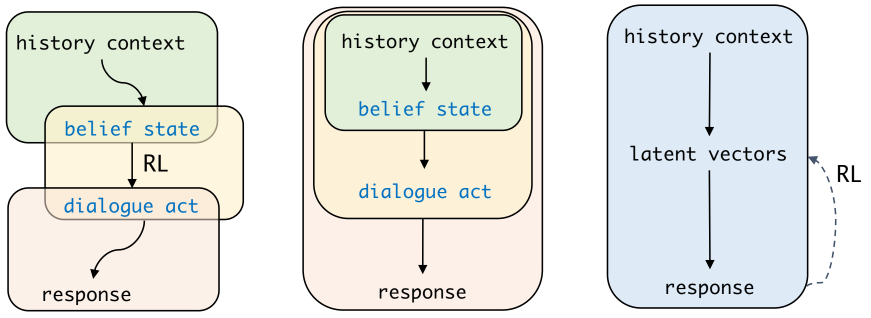
Task-oriented dialogue systems assist users in completing various tasks by generating appropriate responses. The key lies in effective strategy learning and surface realization, which are largely mixed together by the cutting-edge methods. They thus face two problems: a) the learning of high-level strategy could easily be misled by the detailed word sequence optimization, and b) directly emphasizing the agent's goal through reinforcement learning (RL) also leads to corrupted solutions like ungrammatical or repetitive responses. In this work, we propose to decouple the strategy learning and surface realization in a general framework, called DSSR. The core is to construct a latent content space for strategy optimization and disentangle the surface style from it. Specifically, we optimize the latent content distribution for strategy towards task completion, and assume that such distribution is shared across different surface style realizations. By further constructing an encoder-decoder scheme for the surface part, it not only facilitates decoupled optimization via RL for both strategy and surface asynchronously, but also supports controllable surface style transfer of responses. We test DSSR on the multi-domain dialogue datasets MultiWoz 2.0 and MultiWoz 2.1 in comparison with methods mixing strategy and surface realization in different levels, showing improvements in the performance evaluated by various evaluation metrics. Finally, we demonstrate the semantic meanings of latent content distributions to show the disentangling effect of DSSR, and show that it can do effective surface style transfer as by-products.

1 Introduction

With the rise of various personal assistants, task-oriented dialogue (ToD) systems have received a surge in popularity and attention. Different from open-domain dialogues, the ultimate goal of such ToD systems is to achieve satisfactory task completion, thus the generated responses are evaluated on both the strategy and surface realization. Traditionally, a ToD system is built using the divide and conquer approach (Mehri et al., 2019), resulting in multiple modules in a pipeline as shown in Figure 1 (a). The prediction of annotated dialogue act absorbs the strategy modeling, while the following response generation realizes the surface forms. However, such hard decoupling scheme suffers from error propagation and information loss between the modules, *e.g.* the context details such as response pattern in former turns are no longer accessible after dialogue act prediction.

To alleviate the limitations of such hard decoupling, another line of efforts uses a single language model that subsumes modules together (Hosseini-Asl et al., 2020), where all details are kept as shown in Figure 1 (b). Nonetheless, all labeled results in the middle are indispensable and such models focus on optimizing the likelihood of the data but fail to foresee the future for strategy optimization. To address this problem, we see a surge in end-to-end approaches that rely on latent vectors and apply RL techniques to learn good strategy (Zhao et al., 2019), as illustrated in Figure 1 (c). Although promising performance is achieved, such methods usually mix the strategy and surface realization together (He et al., 2018). As a consequence, the learning of high-level strategy would easily be misled by the detailed word sequence optimization, and directly optimizing the strategy via RL often leads to corrupted utterances that are ungrammatical or repetitive.

In this work, we aim to directly optimize strategy while maintaining good control over surface realization. To realize this aim, it requires the system to moderately disentangle the two aspects and



(a) Pipe-lined (b) Language modeling (c) End-to-end RL

Figure 1: The three popular modeling schemes for task-oriented response generation.

optimize them in a collaborative way. However, these aspects interact in subtle ways in natural language responses. Hence, instead of directly separating them into two-stages for sub-optimal results, we formalize a latent content space to represent the surface-independent content part of the agent response, and permit it to be shared across different surface realizations as shown in Figure 2. We further construct an encoder decoder close-loop to deliberately control the surface realization: the encoder takes each surface realization form and its style indicator as input to extract the surface-independent content representation. The learned representation is then carefully aligned and passed to the surface-dependent decoder for rendering with assigned surface styles. In general, the surface-independent content space is the crux of our proposed DSSR. It not only enables decoupled optimization via RL for both the strategy and surface asynchronously, but also supports controllable surface style transfer of responses.

To demonstrate the effectiveness of the proposed DSSR, we evaluate it on two public multi-domain dialogue datasets (MultiWoz 2.0 and 2.1). We compare it with the state-of-the-art methods that mixing strategy and surface realization in different levels. The model achieves strong performance improvements on automatic evaluation metrics, reaching 110.47 and 107.08 combined scores respectively, leading the board on the total performance in the official records [1]. Moreover, we demonstrate the disentangling effect of DSSR by illustrating the semantic meanings of latent distributions and show that it can perform effective style transfer between dialogue acts and responses as by-products.

2 Related Work

Separated Response Generation. ToD systems help users to complete tasks such as finding restaurants or booking flights. Building such systems typically requires separated modules to construct a pipeline: natural language understanding to extract user's intents (*e.g. inform*) and slot values (*e.g. area-center*), dialogue state tracking (DST) to update belief states, dialogue policy to decide the system's next action, and natural language generation (NLG) to generate real responses. Such separated modules are trained independently with different supervision, *e.g.* the dialogue policy module employs dialogue act labels, and the NLG module accesses templatized or natural responses.

- ***Strategy*** is mainly modeled in the dialogue policy module (Wen et al., 2017; Zhao & Eskenazi, 2016; Liu & Lane, 2017). It aims to solve real-world challenges more efficiently, *e.g.* to improve task completion (Li et al., 2016) or win negotiations (Lewis et al., 2017) *etc*. A classic solution employs RL to learn the optimal action distribution conditioned on the belief state. However, the action space is hand-crafted dialogue acts and slot values *e.g. inform(departure time, 17:00)*. This is limited because it can only handle simple domains whose entire action space can be enumerated. More importantly, it is hardly optimal for strategy modeling. Limited by the act and slot numbers, it not only fails to represent some complicated situations, but also loses many context details.
- ***Surface realization*** is the goal of the subsequent NLG module, where two broad categories of methods are popular. (1) *Template-based methods* require a minimal set of manually defined templates to generate simple utterances. Some efforts exploit a retrieval process to replace the handcrafting procedure (Wu et al., 2019). Thus, the produced responses are often fluent, but

[1]https://github.com/budzianowski/multiwoz

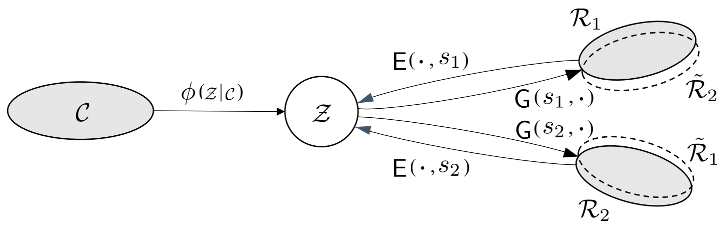


Figure 2: An overview of the proposed decoupling framework DSSR. The network ϕ maps the dialogue context $\mathcal{C}$ to latent content $\mathcal{Z}$ in agent action space. $\mathcal{Z}$ is shared in responses $\mathcal{R}_1$ and $\mathcal{R}_2$ in two styles s_1 and s_2. Encoder E extracts the content while stripping off the indicated surface styles, and generator G generates the response back when given the original style. When indicating with a different style, transferred $\tilde{\mathcal{R}}_1$ and $\tilde{\mathcal{R}}_2$ will be generated respectively.

not always natural as some required semantic information might be mismatched (Langkilde & Knight, 1998; Kale & Rastogi, 2020; Wang et al., 2021). (2) *Seq2seq-based methods* are also commonly leveraged to directly convert a sequential representation of system actions to a system response (Zhu et al., 2019; Wang et al., 2020b). Such methods can generate utterances containing novel patterns but heavily rely on a large amount of training data. Thus specific problems such as domain adaptation and transfer learning in low resource settings has been extensively studied (Chen et al., 2020; Peng et al., 2020b). However, all of these methods ignore the different styles in surface realization, let alone support flexible transfer between various realizations.

Entangled Response Generation. To overcome the information loss problem of separated response generation, numerous end-to-end entangled methods have been proposed, which fall in two lines. (1) *Language modeling methods* largely benefit from the large pre-trained transformer-based models such as BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019). By connecting dialogue context, intermediate results and response into a long sequence, such systems typically rely on language modeling techniques to directly optimize the data likelihood while neglecting the planning altogether (Hosseini-Asl et al., 2020; Peng et al., 2020a; Yang et al., 2021). It sets barriers for systems to anticipate the future for more intelligent responses. (2) *End-to-end RL methods* hence come with latent variables and optimize via RL. Initially, the action space for RL is defined as the entire vocabulary (Li et al., 2016). However, it blows up the size of action space and the trajectory length. Also, to simultaneously optimize the language coherence and decision making within one model can lead to divergence (Das et al., 2017; He et al., 2018). Therefore, Zhao et al. (2019) propose to construct a latent space between the context encoder and the response decoder as the action space. Better performance is reported due to the condensed representation and shorter trajectory. Lubis et al. (2020) further leverage auxiliary tasks to shape the latent variable distribution and Wang et al. (2020a) model the hierarchical structure between the dialogue policy and NLG with the option framework (Sutton et al., 1999). While improvements are reported on task completion, lack of interpretability and controllability remains a major challenge. In our work, we not only decode responses conditioned on the latent variable to decouple action selection and language generation, but also encourage the latent variable to truly interpret the content of agent actions and support flexible control of surface realizations.

3 FORMULATION

In this section, we formalize the response generation task for ToD and illustrate the feasibility of the decoupling scheme. We assume that the data are generated by the following process:

1. given dialogue context $\boldsymbol{c}$, a latent content variable $\boldsymbol{z}$ is generated from the distribution $p(\boldsymbol{z}|\boldsymbol{c})$;
2. a latent style variable $\boldsymbol{s}$ is generated from a prior distribution $p(\boldsymbol{s})$;
3. a detailed response $\boldsymbol{r}$ is generated from conditional distribution $p(\boldsymbol{r}|\boldsymbol{s}, \boldsymbol{z})$.

We observe a set of dialogue context $\boldsymbol{c}$ paired with responses with the same content distribution but in two different styles $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. Formally, we have a dataset $\mathcal{D} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_i, \cdots, \mathcal{T}_N\}$ with N data samples, where each tuple $\mathcal{T}_i = (\boldsymbol{c}^{(i)}, \boldsymbol{r}_1^{(i)}, \boldsymbol{s}_1^{(i)}, \boldsymbol{r}_2^{(i)}, \boldsymbol{s}_2^{(i)})$. We do not have the exact definitions of the styles but can differentiate between them, *i.e.* each response sample can be viewed as drawn from $p(\boldsymbol{r}_1|\boldsymbol{s}_1)$ or $p(\boldsymbol{r}_2|\boldsymbol{s}_2)$ respectively. Our goal is to estimate the response generation functions $p(\boldsymbol{r}_1|\boldsymbol{c}, \boldsymbol{s}_1)$ and $p(\boldsymbol{r}_2|\boldsymbol{c}, \boldsymbol{s}_2)$. Since the latent content part $\boldsymbol{z}$ is shared, we also learn the surface style transfer functions $p(\boldsymbol{r}_1|\boldsymbol{r}_2; \boldsymbol{s}_1, \boldsymbol{s}_2)$ and $p(\boldsymbol{r}_2|\boldsymbol{r}_1; \boldsymbol{s}_1, \boldsymbol{s}_2)$.

Intuitively, the latent content variable $\boldsymbol{z}$ should carry most of the complexity of response $\boldsymbol{r}$, while the surface style indicator $\boldsymbol{s}$ should have relatively simple effects. Still, responses generated from different surface styles should be "distinct" enough, otherwise, the controlled generation task is not well defined (Shen et al., 2017). Here, we assume that $\boldsymbol{z}$ is a Gaussian mixture and we demonstrate that it is probable to recover the generation functions and surface style transfer functions.

Lemma 1. *Let $\boldsymbol{z}$ be a mixture of Gaussians $p(\boldsymbol{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. Assume $K \geq 2$, and there are two different $\boldsymbol{\Sigma}_i \neq \boldsymbol{\Sigma}_j$. Let $\mathcal{S} = \{(\boldsymbol{A}, \boldsymbol{b}) || \boldsymbol{A}| \neq 0\}$ be all invertible affine transformations, and $p(\boldsymbol{r}|\boldsymbol{s}, \boldsymbol{z}) = \mathcal{N}(\boldsymbol{r}; \boldsymbol{A}\boldsymbol{z} + \boldsymbol{b}, \epsilon^2 \boldsymbol{I})$, in which ϵ is a noise. Then for all $\boldsymbol{s} \neq \boldsymbol{s}' \in \mathcal{S}$, $p(\boldsymbol{r}|\boldsymbol{s})$ and $p(\boldsymbol{r}|\boldsymbol{s}')$ are different distributions.*

The lemma shows that the controlled generation between $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, two affine transformations of content $\boldsymbol{z}$, can be recovered given the observed marginals $p(\boldsymbol{r}_1|\boldsymbol{s}_1)$ and $p(\boldsymbol{r}_2|\boldsymbol{s}_2)$ when surface style $\boldsymbol{s}$ is distinguishable and $\boldsymbol{z}$ is a mixture of Gaussian distributions with more than two components.

4 Method

As discussed, learning decoupled response generation for better interpretability and controllability is essentially learning the conditional probability $p(\boldsymbol{r}_1|\boldsymbol{c}, \boldsymbol{s}_1)$ and $p(\boldsymbol{r}_2|\boldsymbol{c}, \boldsymbol{s}_2)$ under our generative assumption. In the learning scheme, we also harvest surface style transfer functions $p(\boldsymbol{r}_1|\boldsymbol{r}_2; \boldsymbol{s}_1, \boldsymbol{s}_2)$ and $p(\boldsymbol{r}_2|\boldsymbol{r}_1; \boldsymbol{s}_1, \boldsymbol{s}_2)$ bridged by the latent $\boldsymbol{z}$. We have demonstrated the feasibility of the learning problem by assuming $\boldsymbol{z}$ as a Gaussian mixture with multiple components and requiring styles being distinguishable. In this section, we introduce the proposed DSSR framework in detail.

4.1 Preliminaries

In the current popular end-to-end RL framework (Zhao et al., 2019; Lubis et al., 2020), the response generation task is typically performed in two steps: supervised learning (SL) pretraining and RL finetuning. In the SL pretraining step, the model learns to generate a response $\boldsymbol{r}$ based on the dialogue context $\boldsymbol{c}$ in an end-to-end fashion. During SL, it updates the neural network parameters θ to maximize the log likelihood of the training data as:

$$L_{SL} = \mathbb{E}_{\boldsymbol{r},\boldsymbol{c}}[\log p_\theta(\boldsymbol{r}|\boldsymbol{c})]. \tag{1}$$

After achieving a good parameter setting $\hat{\theta}$ via SL, the RL step starts from it and further updates the model parameter *w.r.t.* the task-specific goal, reflected as a reward. Suppose a dialogue has T turns, for a specific time-step t, the discounted return is defined as $O_t = \sum_{i=t}^{T} \gamma^{i-t} o_i$, where o_t is the immediate reward for turn t and $\gamma \in [0, 1]$ is the discount factor. The model tries to maximize the expected return from the first time-step onwards, which is $J = \mathbb{E}[\sum_{t=0}^{T} \gamma^t o_t]$.

4.2 Decoupling Strategy and Surface Realization

To enable appropriate decoupling of strategy and surface realization, we leverage a latent variable $\boldsymbol{z}$ and introduce a surface style indicator $\boldsymbol{s}$. Then the SL part becomes $p(\boldsymbol{r}|\boldsymbol{c}, \boldsymbol{s})$, which can be further factorized into $p(\boldsymbol{r}|\boldsymbol{c}, \boldsymbol{s}) = p(\boldsymbol{z}|\boldsymbol{c}) p(\boldsymbol{r}|\boldsymbol{z}, \boldsymbol{s})$. By treating the latent space $\boldsymbol{z}$ as the content part of the response and adding surface style factor during generation, we not only manage to reduce the action space size and trajectory length for RL, but also support controllable response generation.

Strategy. Under this setting, the RL finetuning step aims to learn a good strategy towards task completion goals. We apply REINFORCE (Williams, 1992) to finetune ϕ which is responsible for choosing the best latent action $\boldsymbol{z}$ given dialogue context $\boldsymbol{c}$ (*c.f.*, Figure 2). The policy gradient is defined as:

$$\nabla J(\phi) = \mathbb{E}_\phi[\sum_{t=0}^{T} O_t \nabla \log \phi(\boldsymbol{z}_t|\boldsymbol{c}_t)]. \tag{2}$$

By introducing RL over strategy learning, the model manages to foresee the future for more intelligent response in order to achieve better task completion.

Surface Realization. As discussed in Section 3, we assume that the content variable $\boldsymbol{z}$ takes the form as a multivariate Gaussian mixture with a diagonal covariance matrix. It is learned during the SL pretraining step using stochastic variational inference by maximizing the evidence lowerbound (ELBO) on the data log likelihood. To further combat the exposure bias in the latent content space, we may come up with a new variational lower bound as:

$$L_{ELBO} = \mathbb{E}_{\boldsymbol{z}\sim\phi(\boldsymbol{z}|\boldsymbol{c})}[\log \mathsf{G}(\boldsymbol{r}|\boldsymbol{z},\boldsymbol{s})] - \eta \text{ KL } [\phi(\boldsymbol{z}|\boldsymbol{c})||p(\boldsymbol{z})], \tag{3}$$

where η is a weight to constrain $\phi(\boldsymbol{z}|\boldsymbol{c})$ to be similar to certain priors. Note that in Figure 2, we defined an encoder network E and a generator G. Formally, let $\mathsf{E} : \mathcal{R} \times \mathcal{S} \to \mathcal{Z}$ be an encoder that infers the content $\boldsymbol{z}$ from a given response $\boldsymbol{r}$ and a surface style $\boldsymbol{s}$, and $\mathsf{G} : \mathcal{Z} \times \mathcal{S} \to \mathcal{R}$ be a generator that generates a response $\boldsymbol{r}$ from a given content $\boldsymbol{z}$ and surface style $\boldsymbol{s}$. Naturally, the extracted $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ from $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ should be aligned to the $\boldsymbol{z}$ predicted by $\phi(\boldsymbol{z}|\boldsymbol{c})$. Hence the second term about KL divergence in Equation (3) can be replaced as:

$$\alpha \text{ KL } [\phi(\boldsymbol{z}|\boldsymbol{c})||\mathsf{E}(\boldsymbol{z}_1|\boldsymbol{r}_1,\boldsymbol{s}_1)] + \beta \text{ KL } [\phi(\boldsymbol{z}|\boldsymbol{c})||\mathsf{E}(\boldsymbol{z}_2|\boldsymbol{r}_2,\boldsymbol{s}_2)], \tag{4}$$

where α and β are hyper parameters for weights.

Meanwhile, the networks E and G form an auto-encoder model. When applying to the same surface style, the reconstruction loss for responses will be,

$$\begin{aligned} L_{RE}(\mathsf{E},\mathsf{G}) = \; & \mathbb{E}_{\boldsymbol{r}_1\sim\boldsymbol{R}_1}[-\log p_{\mathsf{G}}(\boldsymbol{r}_1|\boldsymbol{s}_1,\mathsf{E}(\boldsymbol{r}_1,\boldsymbol{s}_1))] + \\ & \mathbb{E}_{\boldsymbol{r}_2\sim\boldsymbol{R}_2}[-\log p_{\mathsf{G}}(\boldsymbol{r}_2|\boldsymbol{s}_2,\mathsf{E}(\boldsymbol{r}_2,\boldsymbol{s}_2))]. \end{aligned}$$

The overall training objective for SL is defined to update the parameters for the mapping network ϕ, encoder E and generator G. It is defined as:

$$L_{ELBO} + \lambda L_{RE},$$

where λ is the weight hyper parameter to adjust the effect of reconstruction. The KL divergence term in L_{ELBO} is defined in Equation (4).

Although the reconstruction loss is only calculated for the same surface style, the alignment of $\boldsymbol{z}$ by KL divergence in Equation (4) enables the style-controlled response generation, *i.e.* transferring styles between responses. For instance, when E and G are well-trained, given a response $\boldsymbol{r}_1$, its surface style $\boldsymbol{s}_1$ and a transfer target style $\boldsymbol{s}_2$, we can transfer it to $\boldsymbol{r}_2$. Formally,

$$\begin{aligned} p(\boldsymbol{r}_2|\boldsymbol{r}_1;\boldsymbol{s}_1,\boldsymbol{s}_2) &= \int_{\boldsymbol{z}} p(\boldsymbol{z}|\boldsymbol{r}_1,\boldsymbol{s}_1)p(\boldsymbol{r}_2|\boldsymbol{z},\boldsymbol{s}_2)d\boldsymbol{z} \\ &= \mathbb{E}_{\boldsymbol{z}\sim\mathsf{E}(\boldsymbol{r}_1,\boldsymbol{s}_1)}[\mathsf{G}(\boldsymbol{r}_2|\boldsymbol{z},\boldsymbol{s}_2)]. \end{aligned}$$

Asynchronous Optimization. After obtaining a good parameter setting for the surface realization network G via SL pretraining, we can also further finetune it via RL to obtain better results. Hence we apply REINFORCE again to update G which is responsible for transforming $\boldsymbol{z}$ into the indicated surface-form. By treating every output word as an action step, the policy gradient is defined as:

$$\nabla J(\mathsf{G}) = \mathbb{E}_{\mathsf{G}}[\sum_{t=0}^{T}\sum_{j=0}^{U_t} O_{tj}\nabla \log \mathsf{G}(\boldsymbol{w}_{tj}|\boldsymbol{w}_{<tj},\boldsymbol{z}_t,\boldsymbol{s}_t)], \tag{5}$$

where U_t is the number of tokens in the response at turn t and j is the token index in the response.

The goal of the whole DSSR learning process is to find the best maximizers that can maximize the reward value regarding both strategy and surface realization. The two policies are defined in Equation (2) and (5), respectively. If we synchronously update these two policies, the composite state will be inconsistent before and after the update each time. Consequently, the value does not always monotonically improve during the learning process. It will affect the convergence of both policies. Therefore, we update the two asynchronously during learning to guarantee the convergence of these policies to a local maximizer.

4.3 Implementation of DSSR

In implementation of DSSR, we represent the content part as latent vectors $\boldsymbol{z}$, which enables the flexible decoupling scheme. In detail, a content variable $\boldsymbol{z}$ is sampled from a dialogue strategy represented as a multivariate Gaussian mixture such that $\phi(\boldsymbol{z}|\boldsymbol{c}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. It is implemented with one-layer linear model and outputs the mean and variance. Inspired by (Chen et al., 2019), we further append the predicted graph act vector to enhance the model's generalization ability to sparse training instances. The input of $\phi(\boldsymbol{z}|\boldsymbol{c})$ is a context vector $\boldsymbol{c}$. Specifically, the last user utterances are firstly encoded with a bidirectional RNN with GRU cell and global type attention mechanism. Then, an oracle dialogue state and an oracle database search result are concatenated with it to form the $\boldsymbol{c}$. The utterance encoder is only trained during pretraining and fixed as a context extractor during asynchronous RL, so that the context space is reduced.

For the surface realization part, we implement the encoder E and the generator G by using single-layer RNNs with GRU cell. E takes an input sentence $\boldsymbol{r}$ with surface style $\boldsymbol{s}$. It strips off the indicated style information from the input sentence and outputs the latent content variable $\boldsymbol{z}$ as its content representation. G generates a response $\boldsymbol{r}$ from the content vector $\boldsymbol{z}$ with style indicator $\boldsymbol{s}$. For the generator G, $p(\boldsymbol{w}_t|\boldsymbol{z}, \boldsymbol{s})$ is represented as a categorical distribution over a word, conditioned on a content $\boldsymbol{z}$ and a style $\boldsymbol{s}$. The information in the initial state is assumed to be propagated to the hidden states at the future time steps, so we only feed it in the initial state in implementation. The surface style indicator $\boldsymbol{s}$ is implemented via one-layer linear model, only trained during pretraining and fixed during asynchronous RL.

5 Experiments

5.1 Settings

Datasets. We conduct experiments on the latest benchmark datasets MultiWoz 2.0 (Budzianowski et al., 2018) and MultiWoz 2.1 (Eric et al., 2019) to evaluate our proposed decoupling scheme. MultiWoz 2.0 is a large scale task-oriented dialogue dataset containing over ten thousand dialogues that spans over seven distinct domains. All the dialogues are collected by human-to-human conversations via the crowd sourcing WOZ setting. In which, every dialogue is generated where the user is given a pre-defined goal and the system attempts to fulfill the goal by interacting with the user. We follow the same delexicalized method provided by (Budzianowski et al., 2018) to pre-process the dataset, which is widely applied in other works (Zhao et al., 2019; Wang et al., 2020a). MultiWoz 2.1 is a modified version of MultiWoz 2.0 which mainly fixes the noisy dialogue state annotations and corrects a small fraction of dialogue utterances. Note that we follow the public divisions to split the datasets into training, validation and testing sets (Budzianowski et al., 2018; Eric et al., 2019).

Evaluation Metric. Following existing response generation works such as (Budzianowski et al., 2018), we adapt three automatic metrics measured in percentage to evaluate the generated utterances from a dialogue system such as *Inform* rate, *Success* rate and *BLEU* score. *Inform* rate measures whether the system has provided the correct entity (*e.g.*, the name of restaurant). *Success* rate shows the ratio of correct answers provided for request slots in the generated utterances. The fluency of the generated response is measured by *BLEU* (Papineni et al., 2002) score. The *Combined* score is computed as $(BLEU + 0.5 \times (Inform + Success))$ (Budzianowski et al., 2018) to fairly evaluate the performance of a dialogue system as a popular total score.

Task Setting. As we focus on decoupling the strategy and different surface realizations of the response, experiments are conducted on the dialogue-context-to-text generation task. As originally proposed in (Budzianowski et al., 2018), we assume that the model has access to the oracle belief state and database search result. Given the dialogue context, the model is trained to generate appropriate utterances as a response in each turn. Besides the natural language responses, we also view the structured dialogue acts in sequence as another style of surface realization. It carries the same content meaning as its natural language counterpart but in different surface style. Given the surface style indicators, our model manages to flexibly transfer between these two realizations. After supervised training of the model, to further fine-tune the model with asynchronous RL, each dialogue is only evaluated with the goal (*e.g.* calculating the *Success* rate) and the *BLEU* score at the end of dialogue. More training details can be found on Appendix B.

Baseline Models. The variation of DSSR without RL finetuning is denoted as DSSR$_{\text{-}SL}$. We compare DSSR with methods mixing strategy and surface realization at different levels. These baselines are organized into three groups, *i.e.*, pipe-lined, language modeling based and end-to-end RL based methods. All these models leverage oracle dialogue states and database search results.

- **Pipe-lined**. SFN (Mehri et al., 2019) learns neural dialogue modules corresponding to the structured components of traditional dialogue systems. It obtains strong results both with (denoted as SFN) and without reinforcement learning (SFN$_{\text{-}SL}$).
- **Language modeling**. All compared methods are based on fine-tuning the pretrained GPT-2. SimpleTOD (Hosseini-Asl et al., 2020) works on turn-level sequences, while UBAR (Yang et al., 2021) treats a whole dialogue session as a single training sequence. They both rely on labeled intermediate results, but DialogGPT (Zhang et al., 2020) totally ignores such labels.
- **End-to-end RL**. We also compare with the state-of-the-art end-to-end RL based methods. LaRL (Zhao et al., 2019) is the first to represent dialogue act as latent vectors in ToD. During RL training, it only updates dialogue policy. Based on that, LAVA (Lubis et al., 2020) further leverages three auxiliary tasks to shape the latent variable distribution. HNDO (Wang et al., 2020a) adopts the option framework (Sutton et al., 1999) to model the hierarchical relation between dialogue policy and NLG. We also report its SL only version as HNDO$_{\text{-}SL}$.

5.2 Main Results

How does decoupling work? The main results for response generation are shown in Table 1. The proposed DSSR method achieves the best performance regarding the overall performance reflected by the *Combined* scores. It shows balanced results over both strategy learning for task completion and surface realization, validating the effectiveness of the decoupling scheme. Specifically, we observe a general trend that RL applied methods can largely boost the strategy part as expected, because the task completion rates are directly considered as rewards for optimization. For example, in pipeline-based methods, the RL applied SFN outperforms its SL counterparts, especially in task completion metrics like *Inform* rate and *Success* rate. This is also true in end-to-end RL based methods such as HDNO. However, such methods tend to generate as many slots as possible to increase these rates, which leads to ungrammatical or repetitive responses. Hence, such destruction of surface realization is often the price to pay for the increasing strategy performance. It is evidenced by the generally lowest *BLEU* score reported in LaRL and LAVA, the decrease of *BLEU* in SFN from SFN$_{\text{-}SL}$ and HDNO from HDNO$_{\text{-}SL}$.

On the contrary, since the proposed DSSR deliberately decouples the strategy and surface realization and optimizes them via asynchronous RL separately, it not only enhances the strategy part to predict succinct slots, but also further helps the surface realization to generate more fluent responses (see examples in Appendix C.1). For language modeling based methods, the results show that how to model the task is the key to achieve good performance via the powerful large-scale pretraining models such as GPT-2. Although UBAR manages to achieve over 100.0 *Combined* scores, the lack of foreseeing the future is still a main shortcoming for such models.

Table 1: Main results on MultiWoz 2.0 and MultiWoz 2.1.

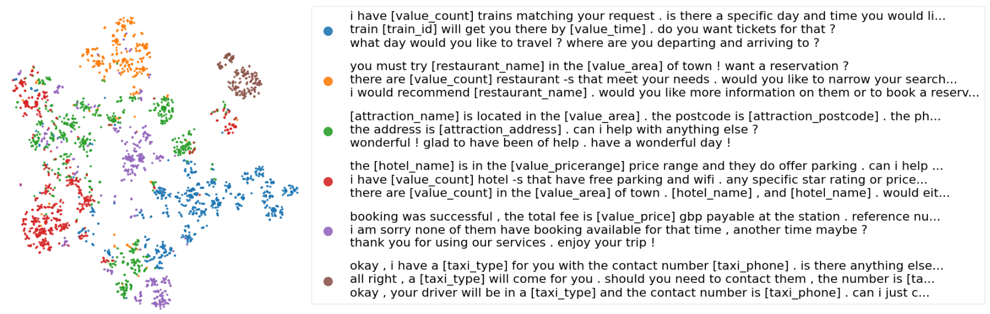
	MultiWoz 2.0				**MultiWoz 2.1**			
	Inform	**Success**	**BLEU**	**Combined**	**Inform**	**Success**	**BLEU**	**Combined**
SFN$_{\text{-}SL}$	90.00	74.20	18.35	100.45	63.10	53.10	17.56	75.66
SFN	94.40	83.10	16.34	105.09	87.80	76.20	10.57	92.57
DialogGPT	73.40	48.00	12.16	72.86	72.10	50.10	12.62	73.72
SimpleTOD	88.90	67.10	16.90	94.90	85.10	73.50	16.22	95.52
UBAR	94.00	83.60	17.22	106.02	89.6	78.6	17.34	101.44
LaRL	93.49	84.98	12.01	101.25	92.39	**85.29**	13.72	102.56
LAVA	**97.50**	**94.80**	12.02	108.17	**96.39**	83.57	14.02	104.00
HDNO$_{\text{-}SL}$	78.60	70.40	19.26	93.76	78.80	66.70	18.46	91.21
HDNO	95.80	84.50	18.61	108.76	93.20	81.90	18.35	105.90
DSSR$_{\text{-}SL}$	92.90	84.90	19.38	108.28	90.10	82.70	19.99	106.39
DSSR (ours)	94.60	87.20	**19.57**	**110.47**	90.40	81.90	**20.93**	**107.08**

On the effect of different surface realization. DSSR also manages to generate the dialogue act sequences, where each action is organized as a *(domain, action, slot, value)* tuple. We separately check the accuracy of domain, action and slot. We also evaluate the tuples via *F1* score. Since the end-to-end RL based methods do not have such outputs, we skip the comparison. Pipeline-based SFN works well on task completion metrics and predict vector based dialogue acts. However, its tuple *F1* score did not exceed 30 thus we did not report here. We suspect that its RL fine-tuning process makes the dialogue act vector diverge from its original semantic space where each dimension corresponds to a specific dialogue act. Instead, we compare with UBAR, which is the best performing method with such outputs. As shown in Table 2, DSSR performs better, which reflects that the learned latent vector indeed contains the right content that can be effectively realized in different surface forms.

Table 2: Dialogue act generation results on MultiWoz 2.0 and MultiWoz 2.1.

	MultiWoz 2.0				**MultiWoz 2.1**			
	domain	**action**	**slot**	**F1**	**domain**	**action**	**slot**	**F1**
UBAR	88.42	58.88	50.72	52.41	87.46	59.54	50.56	52.95
DSSR (ours)	**89.41**	**74.72**	**61.91**	**59.45**	**89.27**	**73.61**	**61.67**	**58.73**

On the semantic meanings of latent content space. As shown in Figure 3, we visually assess the latent content space by first clustering the latent content vector of each system response in the testing set into six clusters, and then projecting them with t-SNE (Van der Maaten & Hinton, 2008) to analyze the formed clusters. Through inspecting the randomly selected system utterances as shown in the right hand side, we find that the clusters of latent content vectors of both DSSR and HDNO possess some semantic meanings. For example, the cluster in blue dots in DSSR is related to train booking and the cluster in yellow dots is related to restaurant recommendation, while the cluster in brown dots in HDNO is related to the general phrases for goodbye at the end of service. However, it is also obvious that the clusters from DSSR as shown in Figure 3 (a) are relatively better separated, which demonstrates clearer semantic meanings expressed by these content vectors. This might be due to the successful decoupling of content and surface styles in these latent vectors.


● i have [value_count] trains matching your request . is there a specific day and time you would li...
train [train_id] will get you there by [value_time] . do you want tickets for that ?
what day would you like to travel ? where are you departing and arriving to ?

● you must try [restaurant_name] in the [value_area] of town ! want a reservation ?
there are [value_count] restaurant -s that meet your needs . would you like to narrow your search...
i would recommend [restaurant_name] . would you like more information on them or to book a reserv...

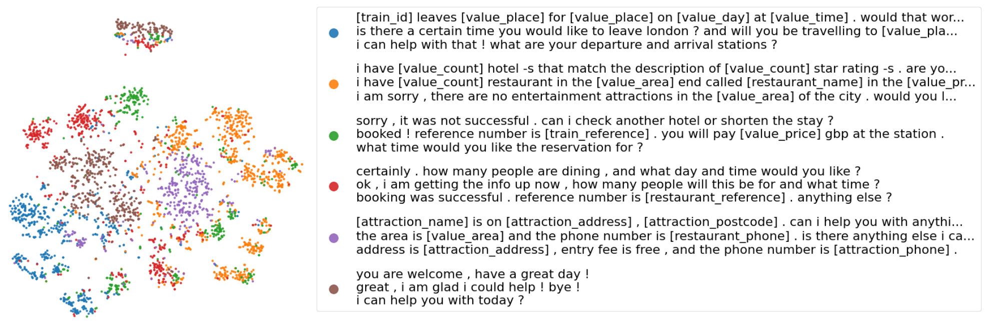
● [attraction_name] is located in the [value_area] . the postcode is [attraction_postcode] . the ph...
the address is [attraction_address] . can i help with anything else ?
wonderful ! glad to have been of help . have a wonderful day !

● the [hotel_name] is in the [value_pricerange] price range and they do offer parking . can i help ...
i have [value_count] hotel -s that have free parking and wifi . any specific star rating or price...
there are [value_count] in the [value_area] of town . [hotel_name] , and [hotel_name] . would eit...

● booking was successful , the total fee is [value_price] gbp payable at the station . reference nu...
i am sorry none of them have booking available for that time , another time maybe ?
thank you for using our services . enjoy your trip !

● okay , i have a [taxi_type] for you with the contact number [taxi_phone] . is there anything else...
all right , a [taxi_type] will come for you . should you need to contact them , the number is [ta...
okay , your driver will be in a [taxi_type] and the contact number is [taxi_phone] . can i just c...

(a) DSSR (ours)


● [train_id] leaves [value_place] for [value_place] on [value_day] at [value_time] . would that wor...
is there a certain time you would like to leave london ? and will you be travelling to [value_pla...
i can help with that ! what are your departure and arrival stations ?

● i have [value_count] hotel -s that match the description of [value_count] star rating -s . are yo...
i have [value_count] restaurant in the [value_area] end called [restaurant_name] in the [value_pr...
i am sorry , there are no entertainment attractions in the [value_area] of the city . would you l...

● sorry , it was not successful . can i check another hotel or shorten the stay ?
booked ! reference number is [train_reference] . you will pay [value_price] gbp at the station .
what time would you like the reservation for ?

● certainly . how many people are dining , and what day and time would you like ?
ok , i am getting the info up now , how many people will this be for and what time ?
booking was successful . reference number is [restaurant_reference] . anything else ?

● [attraction_name] is on [attraction_address] , [attraction_postcode] . can i help you with anythi...
the area is [value_area] and the phone number is [restaurant_phone] . is there anything else i ca...
address is [attraction_address] , entry fee is free , and the phone number is [attraction_phone] .

● you are welcome , have a great day !
great , i am glad i could help ! bye !
i can help you with today ?

(b) HDNO

Figure 3: The latent content vectors of DSSR and HDNO clustered in six categories visualized via the T-SNE algorithm. We randomly show three turns of system utterances for each cluster.

5.3 BY-PRODUCTS

Labeling dialogue acts for natural responses (NR to DA). Thanks to the encoder-decoder close-loop in the proposed DSSR, it is possible to freely transfer between different surface styles of the responses. Here, we evaluate its performance on transferring natural language response (NR) to dialogue act sequence (DA). We feed in a natural language response and input its style indicator to the encoder, then assign the dialogue act sequence style indicator to the decoder to generate a corresponding dialogue act sequence. In this way, we are sort of labeling dialogue acts for natural language responses. The results are reported in Table 3. Generally speaking, the performances are reasonably well. It's accuracy scores exceed 90% for domain prediction on both MultiWoz 2.0 and MultiWoz 2.1, and the *(domain-slot-value)* tuple *F1* scores surpass 70% on both datasets. Such performance signals a possibility of using our method to assist the tedious dialogue act labeling in ToD dataset construction, which might save much human labor. Also, the results demonstrate that the encoder-decoder close-loop is well-trained, and it indeed manages to discriminate the different surface styles and support style controlled generation. Detailed transfer examples can be found in Appendix C.2.

Table 3: Dialogue act labeling results for natural responses on MultiWoz 2.0 and MultiWoz 2.1.

MultiWoz 2.0				**MultiWoz 2.1**			
domain	**slot**	**value**	**F1**	**domain**	**slot**	**value**	**F1**
90.76	64.22	78.03	70.51	90.70	64.96	76.68	71.83

Natural response generation from dialogue acts (DA to NR). With well-trained encoder-decoder close-loop, we also evaluate its performance on transferring dialogue act sequence (DA) to natural language response (NR). We compare it with transformer based HDSA (Chen et al., 2019), LSTM based SC-LSTM (Wen et al., 2015) and GPT-2 based SC-GPT (Peng et al., 2020b). Results are reported in Table 4. Note that DSSR is based on single-layer RNNs with GRU cell which might have hindered its learning capability, hence results in relatively low *BLEU* score. However, it still exceeds that of SC-LSTM. More importantly, we observe that DSSR achieves similar *Entity F1* score with the best performing SC-GPT which is deliberately designed for this task. Following (Chen et al., 2019), *Entity F1* (Wen et al., 2017) is used to evaluate the entity coverage accuracy (including all slot values, days and references, *etc*.) Therefore, it indicates that the proposed DSSR works well in generating meaningful responses regarding task completion. It covers essential details for the dialogues. More detailed transfer examples can be found in Appendix C.2.

Table 4: Natural response generation results from dialogue act sequence on MultiWoz 2.0.

	BLEU	**Entity F1**
HDSA	26.5	87.3
SC-LSTM	21.6	80.4
SC-GPT	**30.8**	**88.4**
DSSR (ours)	22.9	88.0

6 CONCLUSION

In conclusion, we proposed to decouple the strategy learning and surface realization of response for task-oriented dialogues. It deliberately separates the content and surface style to facilitate more targeted optimization and avoid impinging on each other. Hence on one hand, the content part is decided by the dialogue context, constrained by the content part of various responses, and further fine-tuned by RL towards task completion; on the other hand, an encoder-decoder close-loop learns to extract the content with surface styles stripped off as well as generate the response back with indicated styles, which is further optimized with RL asynchronously. We carried out extensive experiments on two public datasets in comparison with a wide range of baselines. The superior performance results demonstrate that the proposed DSSR scheme not only makes the whole learning process more effective, but also enables better interpretability of the learned content representation and flexible control of the response generation.

In the future, we look forward to applying our method for personalized response generation when more such data is available where surface style is an important factor to model. We would also like to further improve the strategy part in handling dialogue situations unseen during training and analyze how our model performs in real dialogue interaction with more unseen situations.

Reproducibility Statement

The source codes and saved model checkpoints have been uploaded as part of the supplementary material and are publicly accessible. For theoretical results, we give a clear proof in Appendix A to demonstrate the feasibility of the learning problem by assuming z as a Gaussian mixture with multiple components and requiring styles being distinguishable. We also provide detailed implementation details in Section 4.3. For datasets, we follow standard pre-processing codes provided in their official repository [2] and adopt the public data split to ensure that results are comparable. More details about our training procedure and hyper-parameter setting can be found in Appendix B.

References

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

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. In *ACL*, pp. 3696–3709, 2019.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. Few-shot nlg with pre-trained language model. In *ACL*, pp. 183–190, 2020.

Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *ICCV*, pp. 2951–2960, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pp. 4171–4186, 2019.

Mihail Eric, Rahul Goel, Shachi Paul, Adarsh Kumar, Abhishek Sethi, Peter Ku, Anuj Kumar Goyal, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tur. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines, 2019.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. Decoupling strategy and generation in negotiation dialogues. In *EMNLP*, pp. 2333–2343, 2018.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*, 2020.

Mihir Kale and Abhinav Rastogi. Template guided text generation for task oriented dialogue. In *EMNLP*, pp. 6505–6520, 2020.

Irene Langkilde and Kevin Knight. Generation that exploits corpus-based statistical knowledge. In *COLING*, 1998.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning of negotiation dialogues. In *EMNLP*, pp. 2443–2453, 2017.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. Deep reinforcement learning for dialogue generation. In *EMNLP*, pp. 1192–1202, 2016.

Bing Liu and Ian Lane. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In *INTERSPEECH*, pp. 2506–2510, 2017.

Nurul Lubis, Christian Geishauser, Michael Heck, Hsien-chin Lin, Marco Moresi, Carel van Niekerk, and Milica Gasic. Lava: Latent action spaces via variational auto-encoding for dialogue policy optimization. In *COLING*, pp. 465–479, 2020.

Shikib Mehri, Tejas Srinivasan, and Maxine Eskenazi. Structured fusion networks for dialog. In *SIGdial*, pp. 165–177, 2019.

[2] `https://github.com/budzianowski/multiwoz`

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pp. 311–318, 2002.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. Soloist: Few-shot task-oriented dialog with a single pre-trained auto-regressive model. *arXiv e-prints*, pp. arXiv–2005, 2020a.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. Few-shot natural language generation for task-oriented dialog. In *EMNLP*, pp. 172–182, 2020b.

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

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *NeurIPS*, pp. 6833–6844, 2017.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Dingmin Wang, Ziyao Chen, Wanwei He, Li Zhong, Yunzhe Tao, and Min Yang. A template-guided hybrid pointer network for knowledge-basedtask-oriented dialogue systems. *arXiv preprint arXiv:2106.05830*, 2021.

Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. Modelling hierarchical structure between dialogue policy and natural language generator with option framework for task-oriented dialogue system. In *ICLR*, 2020a.

Kai Wang, Junfeng Tian, Rui Wang, Xiaojun Quan, and Jianxing Yu. Multi-domain dialogue acts and response co-generation. In *ACL*, pp. 7125–7134, 2020b.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*, pp. 1711–1721, 2015.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*, pp. 438–449, 2017.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Yu Wu, Furu Wei, Shaohan Huang, Yunli Wang, Zhoujun Li, and Ming Zhou. Response generation by context-aware prototype editing. In *AAAI*, pp. 7281–7288, 2019.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. Ubar: Towards fully end-to-end task-oriented dialog system with gpt-2. In *AAAI*, pp. 14230–14238, 2021.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. In *ACL*, pp. 270–278, 2020.

Tiancheng Zhao and Maxine Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *SIGdial*, pp. 1–10, 2016.

Tiancheng Zhao, Kaige Xie, and Maxine Eskenazi. Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. In *NAACL*, pp. 1208–1218, 2019.

Chenguang Zhu, Michael Zeng, and Xuedong Huang. Multi-task learning for natural language generation in task-oriented dialogue. In *EMNLP*, pp. 1261–1266, 2019.

A Proof

Lemma 1. *Let $\boldsymbol{z}$ be a mixture of Gaussians $p(\boldsymbol{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. Assume $K \geq 2$, and there are two different $\boldsymbol{\Sigma}_i \neq \boldsymbol{\Sigma}_j$. Let $\mathcal{S} = \{(\boldsymbol{A}, \boldsymbol{b}) | |\boldsymbol{A}| \neq 0\}$ be all invertible affine transformations, and $p(\boldsymbol{r}|\boldsymbol{s}, \boldsymbol{z}) = \mathcal{N}(\boldsymbol{r}; \boldsymbol{A}\boldsymbol{z} + \boldsymbol{b}, \epsilon^2 \boldsymbol{I})$, in which ϵ is a noise. Then for all $\boldsymbol{s} \neq \boldsymbol{s}' \in \mathcal{S}$, $p(\boldsymbol{r}|\boldsymbol{s})$ and $p(\boldsymbol{r}|\boldsymbol{s}')$ are different distributions.*

Proof.

$$p(\boldsymbol{r} \mid \boldsymbol{s} = (\boldsymbol{A}, \boldsymbol{b})) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{r}; \boldsymbol{A}\boldsymbol{\mu}_k + \boldsymbol{b}, \boldsymbol{A}\boldsymbol{\Sigma}_k \boldsymbol{A}^T + \epsilon^2 \boldsymbol{I})$$

For different $\boldsymbol{s} = (\boldsymbol{A}, \boldsymbol{b})$ and $\boldsymbol{s}' = (\boldsymbol{A}', \boldsymbol{b}')$, $p(\boldsymbol{r}|\boldsymbol{s}) = p(\boldsymbol{r}|\boldsymbol{s}')$ entails that for $k = 1, \cdots, K$,

$$\begin{cases} \boldsymbol{A}\boldsymbol{\mu}_k + \boldsymbol{b} = \boldsymbol{A}'\boldsymbol{\mu}_k + \boldsymbol{b}' \\ \boldsymbol{A}\boldsymbol{\Sigma}_k \boldsymbol{A}^T = \boldsymbol{A}'\boldsymbol{\Sigma}_k \boldsymbol{A}'^T \end{cases}$$

Since all $\mathcal{S}$ are invertible,

$$(\boldsymbol{A}^{-1}\boldsymbol{A}')\boldsymbol{\Sigma}_k (\boldsymbol{A}^{-1}\boldsymbol{A}')^T = \boldsymbol{\Sigma}_k$$

Suppose $\boldsymbol{\Sigma}_k = \boldsymbol{Q}_k \boldsymbol{D}_k \boldsymbol{Q}_k^T$ is $\boldsymbol{\Sigma}_k$'s orthogonal diagonalization. If $k = 1$, all solutions for $\boldsymbol{A}^{-1}\boldsymbol{A}' = \boldsymbol{I}$, *i.e.* $\boldsymbol{A} = \boldsymbol{A}'$, and thus $\boldsymbol{b} = \boldsymbol{b}'$.

Therefore, for all $\boldsymbol{s} \neq \boldsymbol{s}'$, $p(\boldsymbol{r}|\boldsymbol{s}) \neq p(\boldsymbol{r}|\boldsymbol{s}')$.

B Extra experimental details

To train the proposed DSSR model, we first pretrain it using supervised learning and select the checkpoint model with the best performance on the validation set. Then for asynchronous RL, we initialize parameters with the pretrained model and select the best model according to the greatest reward on the validation set. For efficiency, we use greedy search for decoding in validation. During the testing phase, we also apply greedy search on the testing data and obtain our final results. We use stochastic gradient descent (SGD) for RL and Adam optimizer for SL pretraining. For the baselines, we train them with the original source codes. Note that due to the upgrade of the official evaluator [3], we re-run these models from their original open source codes and obtain the updated performance.

Here we list the specific hyper parameters for DSSR SL and RL training. We set both the maximum length for the user's utterance in context and the maximum length for the system's utterance in response to 50. A 44-dimension act vector is separately predicted through a BERT model trained as described in HDSA. We load the ground truth act vector with a dropout rate of 0.6 during training and use predicted act vector during validation and testing. For constructing the DSSR model, the embedding size for each word is set to 100. For the input user utterance, target system action sequence and target system response, both the user utterance encoder (which is part of the context mapping network ϕ) and the surface form encoder E are a one-layer bidirectional RNN that uses GRU cells of size 300. The encoded result is projected to the latent content space, where the size of the shared latent variable $\boldsymbol{z}$ is 500. The input style label, which is either 0 or 1, is processed by a label encoder and outputs a 200-dimension vector for each style label. The style generator G is shared for system action generation and natural response generation. It is a one-layer RNN with a separate embedding layer and GRU cells. Since the initial state for the generator is a concatenation of encoded style label and the latent variable $\boldsymbol{z}$, the cell size for the generator is 700.

To balance the training loss, we have different weights: α and β for the KL divergence loss and λ for the reconstruction loss. They are all set to be 1.0 in our experiments. During SL training, we set batch size as 32 and the number of training epochs as 50. During RL training, we fix each batch as a complete dialogue. For SL training, Adam optimizer is used with an initial learning rate of 0.001 and weight decay 1e-05. For RL training, Stochastic gradient descent (SGD) is used, and the learning rate in the asynchronous optimization process for the two policies is both 0.09 with weight decay 1e-05. Generally speaking, the experiments of DSSR were run on a Nvidia GeForce RTX 2080Ti graphic card, which consumed around 2.5 hours for SL training and less than 1 hours for RL finetuning. Hence, it is not very expensive to reproduce our results as shown in Table 1.

[3]Please check it under the section Response Generation on the official website of MultiWoz: `https://github.com/budzianowski/multiwoz`.

C EXTRA EXPERIMENTAL RESULTS

C.1 EXAMPLE DIALOGUES GENERATED

We showcase some system utterances generated in the same dialogue by the baselines and our proposed DSSR, to give a better sense of what has been generated. Since most of the dialogues in MultiWoz 2.0 and MultiWoz 2.1 are similar, we only show some results on MultiWoz 2.0. As listed in Table 5, the generated utterances of DSSR is apparently more fluent and task completion oriented. For example, it manages to keep the whole dialogue on the topic of college recommendation and successfully book train tickets for the user.

In more details, although all being fine-tuned with RL, DSSR is able to foresee the future better for response generation than other baselines such as HDNO and LAVA. This is evidenced by the response on 'no colleges' in the first turn, while other models such as HDNO and LAVA mention 'swimming pools' and 'multiple sports attractions' instead. This is also evidenced in the second turn that DSSR manages to generate 'free to get in' which corresponds to 'entrance fee' asked in the subsequent turn. Moreover, in comparison with other baselines trained with RL, the generated utterances of DSSR is more stringent in generating slots. Especially, LAVA tends to generate as many slots as possible so as to increase the success rate. *e.g.* generating the extra *[attraction_address]* in the third turn. This is the common issue of most RL methods on ToD system as discussed. However, in DSSR the situation is better. This might be due to our asynchronous RL optimization scheme where task completion goals and surface realization goals are separately optimized in an iterative fashion. The improved *BLEU* score in Table 1 for DSSR also demonstrates this.

It is interesting that we also observe the phenomenon of over-generating slot placeholders in UBAR generated responses, such as the one in the second turn. It generates *[value_choice], [value_type], [value_area], [value_name], [value_address] and [value_price]* in a single turn, which is rather different from the ground truth response where only one *[value_count]* is contained. Since UBAR purely relies on the powerful language modeling GPT-2 model and does not leverage RL, this might be due to the context seen in former turns.

C.2 EXAMPLES FOR SURFACE STYLE TRANSFER

We also showcase surface style transfer results for our proposed DSSR. The examples on labeling dialogue acts for natural responses is shown in Table 6, and the examples on natural response generation from dialogue act sequence is shown in Table 7. The term **NR** stands for natural language response, while the term **DA** stands for dialogue act sequence. Since most of the dialogues in MultiWoz 2.0 and MultiWoz 2.1 are similar, we only show some examples on MultiWoz 2.0.

For changing natural language response to dialogue act sequence, the examples in Table 6 show that the resulting dialogue act sequences are generally close to the ground truth ones. Indeed, there are also some mistakes. For example, in the third example, the *(train, inform, time, [value_count])* is missed while an extra *(train, request, depart, ?)* is predicted. This might be due to some common patterns learned from training data. However, we also found that certain discrepancies are not real mistakes. For instance, in the fourth example, we notice that the *(attraction inform name [attraction_name])* in ground truth becomes *(attraction recommend name [attraction_name])* in generated DA. It is actually unfair to treat it as wrong prediction. Because we need context information to further decide whether the exact intent is *inform* or *recommend*. Also, the ground truth DA misses the label for *'free admission'* while the generated DA contains it.

For changing dialogue act sequence to natural language response, the examples in Table 7 show that the transferred responses are fluent and contain important slot details. For example, the generated responses in the first two examples exactly express the content in dialogue act sequence. Although they share the same content with the ground truth responses, their word overlapping ratio is limited, which might be the reason for the relatively low *BLEU* score of DSSR in Table 4. In the third example, we also observe some information loss in the generated responses from dialogue acts, *e.g.* the *(train request leave ?)* is not expressed properly. However, we also observe that the generated response expresses certain tuples correctly while the ground truth one does not, *e.g.* *(attraction request price ?)* is expressed in the generated NR while not in the ground truth NR.

Table 5: Some delexicalized responses generated by the baselines and DSSR on MultiWoz 2.0.

Method	**Delexicalized Dialogue**
User:	**i am looking to go somewhere in the [value_area] to enjoy some entertainment. can you help?**
GT Sys:	**i cannot find any entertainment venue -s in the [value_area]. would you like to try something different ? a museum perhaps?**
DSSR:	there are no colleges in the [value_area]. would you like to try a different area or attraction type?
HDNO:	i am sorry, there are no swimming pools in the [value_area]. would you like me to look in a different area?
LAVA:	i am sorry, there are no multiple sports attractions in the [value_area]. would you like to try another area?
UBAR:	i am sorry, there are no [value_type] in the [value_area]. would you like to try a different area?
User:	**no museums, what colleges are there around town?**
GT Sys:	**sure, there are [value_count] colleges in that area.**
DSSR:	there are [value_count] colleges in the [value_area]. i recommend the [attraction_name]. it s free to get in.
HDNO:	[attraction_name] is a great place. would you like me to book a ticket for you?
LAVA:	there are [value_count] colleges in the [value_area]. i recommend [attraction_name]. would you like more information?
UBAR:	there are [value_choice] [value_type] in the [value_area]. [value_name] on [value_address] is [value_price] and [value_name] on [value_address] is [value_price].
User:	**can i please get the phone number, postcode and entrance fee?**
GT Sys:	**[attraction_name] is [value_count] i would recommend, their phone number is [attraction_phone], post code is [attraction_postcode], and there is no entrance fee. can i help you with anything else?**
DSSR:	[attraction_name] is a great college in the [value_area]. the phone number is [attraction_phone] and the postcode is [attraction_postcode].
HDNO:	the phone number is [attraction_phone]. the postcode is [attraction_postcode]. is there anything else i can help you with?
LAVA:	the [attraction_name] is located at [attraction_address], postcode [attraction_postcode]. the phone number is [attraction_phone].
UBAR:	the phone number is [value_phone]. the postcode is [value_postcode] and the entrance fee is [value_price].
User:	**i also need a train that leaves on [value_day] and arrives by [value_time].**
GT Sys:	**where are you traveling from? there are [value_count] trains from [value_place] that arrive by that time.**
DSSR:	there are several trains that meet your criteria. where are you departing from and what is your destination?
HDNO:	where will you be departing from?
LAVA:	i have [value_count] trains that meet your criteria. would you like to narrow it down by departure site or arrival time?
UBAR:	where will you be departing from?
User:	**i am leaving [value_place] and going to [value_place].**
GT Sys:	**the first train leaves at [value_time].**
DSSR:	i have train [train_id] that leaves at [value_time] and arrives at [value_time]. would you like me to book it?
HDNO:	[train_id] leaves at [value_time]. would you like me to book a ticket for you?
LAVA:	i have train [train_id] that leaves at [value_time] and arrives at [value_time]. would you like me to book that for you?
UBAR:	[value_id] leaves at [value_leave] and arrives at [value_arrive]. would you like me to book that for you?
User:	**can you book [value_count] tickets?**
GT Sys:	**your reservation has been made. the total fee is [value_price] gbp payable at the station. reference number is [train_reference].**
DSSR:	booking was successful, the total fee is [value_price] gbp payable at the station. reference number is [train_reference]. can i help with anything else?
HDNO:	booking was successful, the total fee is [value_price] gbp payable at the station. reference number is [train_reference]. is there anything else i can help you with?
LAVA:	booking was successful, the total fee is [value_price] gbp payable at the station. reference number is [train_reference].
UBAR:	booking was successful, the total fee is [value_price] payable at the station. reference number is: [value_reference]. can i help you with anything else today?
User:	**ok, thank you. that all the information i need today!**
GT Sys:	**i am glad i can help and enjoy.**
DSSR:	you are welcome. have a great day.
HDNO:	you are welcome. have a great day.
LAVA:	you are welcome. have a great day!
UBAR:	thank you for using our system!

Table 6: Examples on labeling dialogue acts for natural responses on MultiWoz 2.0.

Surface Style	Examples
NR	**glad i could help, have a great day.**
True DA	general bye none none; general welcome none none
Generated DA	general bye none none; general welcome none none
NR	**certainly, how many people will be in your party total?**
True DA	booking request people ?
Generated DA	booking request people ?
NR	**what day would you like to travel? also, when will you need to leave. this train is [value_count] travel minutes.**
True DA	train inform time [value_count]; train request day ?; train request leave ?
Generated DA	train request day ?; train request depart ?; train request leave ?
NR	**the [attraction_name] museum is located at [attraction_address] [attraction_postcode]. it has free admission.**
True DA	attraction inform post [attraction_postcode]; attraction inform name [attraction_name]; attraction inform addr [attraction_address]
Generated DA	attraction recommend name [attraction_name]; attraction nobook type none; attraction nobook offerbooked [attraction_address]; attraction nobook fee none

Table 7: Examples on natural response generation from dialogue act sequence on MultiWoz 2.0.

Surface Style	Examples
DA	**taxi request leave ?**
True NR	what time do you want to leave?
Generated NR	i need to know what time you will be leaving?
DA	**booking inform none none; restaurant recommend area [value_area]; restaurant recommend name [restaurant_name]**
True NR	you must try [restaurant_name] in the [value_area] of town! want a reservation?
Generated NR	i have found [restaurant_name] in the [value_area] area. would you like me to make a reservation for you?
DA	**train inform choice [value_count]; train request leave ?; train request day ?**
True NR	i have [value_count] trains matching your request. is there a specific day and time you would like to travel?
Generated NR	i found [value_count] trains. what day would you like to leave?
DA	**attraction request price ?; attraction recommend type none; attraction nooffer area [value_area]; attraction nooffer type none**
True NR	i cannot find any entertainment venue -s in the [value_area]. would you like to try something different? a museum perhaps ?
Generated NR	i apologize but there are not any colleges in the [value_area]. would you like to try a different part of town, or a different price range?