Schema-Guided Paradigm for Zero-Shot Dialog

Shikib Mehri and **Maxine Eskenazi**
Language Technologies Institute, Carnegie Mellon University
{amehri,max}@cs.cmu.edu

Abstract

Developing mechanisms that flexibly adapt dialog systems to unseen tasks and domains is a major challenge in dialog research. Neural models implicitly memorize task-specific dialog policies from the training data. We posit that this implicit memorization has precluded zero-shot transfer learning. To this end, we leverage the **schema-guided paradigm**, wherein the task-specific dialog policy is explicitly provided to the model. We introduce the Schema Attention Model (SAM) and improved schema representations for the STAR corpus. SAM obtains significant improvement in zero-shot settings, with a **+22 F_1** score improvement over prior work. These results validate the feasibility of zero-shot generalizability in dialog. Ablation experiments are also presented to demonstrate the efficacy of SAM.

1 Introduction

Task-oriented dialog systems aim to satisfy user goals pertaining to certain tasks, such as booking flights (Hemphill et al., 1990), providing transit information (Raux et al., 2005), or acting as a tour guide (Budzianowski et al., 2018). Neural models for task-oriented dialog have become the dominant paradigm (Williams and Zweig, 2016; Wen et al., 2016; Zhao et al., 2017). These data-driven approaches can potentially learn complex patterns from large dialog corpora without hand-crafted rules. However, the resulting models struggle to generalize beyond the training data and underperform on unseen dialog tasks and domains (Zhao and Eskenazi, 2018; Rastogi et al., 2020b).

A long-standing challenge in dialog research is to flexibly adapt systems to new dialog domains and tasks (Zhao and Eskenazi, 2018; Mosig et al., 2020). Consider a system that has been trained to handle several different tasks (e.g., restaurant reservations, ride booking, weather, etc.). How can

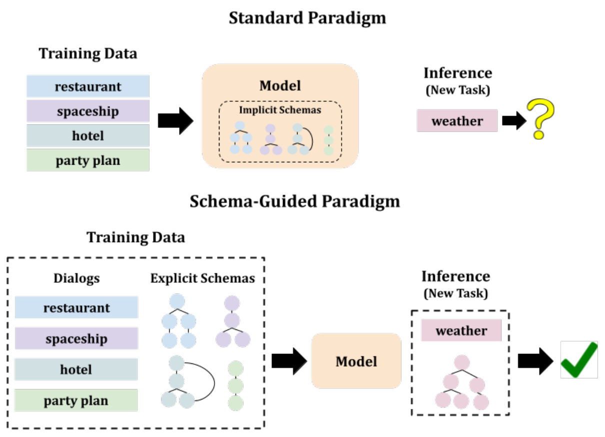

Figure 1: In the standard paradigm, data driven models implicitly learn the task-specific dialog policies (i.e., schemas). This precludes generalization to an unseen task at inference time. In contrast, in the schema-guided paradigm, dialog policy is explicitly provided to the model through a schema graph. At inference time, the model is given the schema for the new task and can therefore generalize in a zero-shot setting.

this dialog system be *extended to handle a new task* (e.g., hotel booking), without collecting additional data? This paper tackles this challenge and aims to address the problem of zero-shot generalization using the **schema-guided paradigm**.

The advent of large-scale pre-training (Devlin et al., 2019; Radford et al., 2019) has led to significant progress in domain adaptation across areas in NLP, including natural language understanding (Wang et al., 2018, 2019), open-domain dialog (Zhang et al., 2020; Adiwardana et al., 2020) and language understanding for task-oriented dialog (Wu et al., 2019; Mehri et al., 2020). Generalization in end-to-end task-oriented dialog has proven to be significantly more difficult, particularly in zero-shot settings where there is no training data (Mosig et al., 2020). We posit that it is inherently difficult to generalize to unseen dialog tasks because of the **dialog policy**.

Traditionally, an end-to-end dialog system must perform three distinct tasks. First, it must understand the dialog history and identify any relevant user intents or slots. Next, it must decide on the appropriate system action, according to a task-specific dialog policy. Finally, it must generate a natural language utterance corresponding to the system action. In a pipeline dialog system, these three steps are performed by the NLU, DM and NLG respectively (Jurafsky, 2000). Prior work has exhibited generalizability in language understanding and, to a lesser extent, in language generation. However for end-to-end dialog, the task-specific dialog policy inherently precludes zero-shot generalization. An end-to-end dialog model trained on several tasks, will *implicitly* learn the dialog policies from the data. However, when generalizing to a new task in a zero-shot setting, the model has no knowledge of the dialog policy for the *new* task.

To address the difficulty of generalizing to new task-specific dialog policies and in order to facilitate zero-shot generalization, we present the **schema-guided paradigm**. Generally, end-to-end neural models implicitly learn the task-specific dialog policies from large corpora. In contrast, in the schema-guided paradigm, we explicitly provide the task-specific dialog policies to the model in the form of a **schema graph**. The schema graphs define the system's behavior for a specific task (e.g., when the user provides the reservation time, ask them for the number of people). When transferring to an unseen task, the corresponding schema graph is explicitly provided to the model. This enables language understanding and the dialog policy to be decoupled. The model no longer needs to implicitly memorize the task-specific policies from the training data. Instead, the model learns to interpret the dialog history and align it to the schema graph. As such, when transferring to a new task, the schema graph serves as an inductive bias that provides the model with the task-specific dialog policy.

To address the challenge of zero-shot transfer learning, Mosig et al. (2020) presented the STAR corpus and several baseline experiments. We extend their baselines for the task of *next action prediction*. We introduce the **Schema Attention Model** (SAM) and thorough schema representations for the 24 different tasks in the STAR dataset. SAM obtains a **+22 $\mathbf{F_1}$** score improvement over baseline approaches in the zero-shot setting, validating the schema-guided paradigm and demonstrating the feasibility of zero-shot generalization for task-oriented dialog. Our code and model checkpoints are open-sourced and be found at `https://github.com/shikib/schema_attention_model`.

2 Related Work

2.1 Zero-Shot Dialog

Zero-shot transfer learning has been of interest to the dialog research community. Many approaches have been proposed for zero-shot adaptation of specific dialog components. Chen et al. (2016) present a zero-shot approach for learning embeddings for unseen intents. Bapna et al. (2017) show that slot names and descriptions can be leveraged to implicitly align slots across domains and achieve better cross-domain generalization. Wu et al. (2019) similarly use slot names, in combination with a generative model for state tracking, to obtain strong zero-shot results. Shah et al. (2019) leverage examples for zero-shot slot filling. Generally, approaches for zero-shot generalizability leverage the similarity across domains (e.g., *restaurant-area* and *hotel-area* are conceptually similar). The advent of large-scale pre-training (Devlin et al., 2019; Radford et al., 2019) allows for language understanding across dissimilar domains. Rastogi et al. (2020a) address zero-shot domain adaptation in state tracking by leveraging BERT (Devlin et al., 2019) with a domain-specific API specification.

Zhao and Eskenazi (2018) present an approach for zero-shot end-to-end dialog. They leverage the Action Matching framework to learn a cross-domain latent action space. Qian and Yu (2019) use model-agnostic meta learning to attain stronger results in zero-shot dialog. Both these approaches rely on additional annotations, which make them unsuitable for the STAR corpus. While there is a significant amount of work in zero-shot generalizability for language understanding, there is considerably less research in adaptation for end-to-end dialog[1]. This is in part because of the difficulty of generalizing to unseen task-specific policies. To this end, Mosig et al. (2020) presented STAR, a corpus consisting of 24 different dialog tasks, and several baseline models for zero-shot adaptation on STAR. The results in this paper significantly

[1]While we focus on next action prediction, in the STAR dataset it is trivial to go from a system action to a natural language response and as such we consider our task to be end-to-end dialog.

outperform the baselines introduced by Mosig et al. (2020) as we leverage the schema-guided paradigm for zero-shot generalizability in dialog.

2.2 Schema-Guided Paradigm

Plan-based dialog systems (Ferguson and Allen, 1998; Rich and Sidner, 1998; Bohus and Rudnicky, 2009) reason about user intent, in the context of a *dialog plan*. RavenClaw (Bohus and Rudnicky, 2009) consists of a task specification that defines the behavior of a system depending on various user actions. Plan-based dialog systems decouple the task-specific dialog policy from the task-agnostic components of the system. This allows a system to be extended to a new task by updating the task specification. The schema-guided paradigm shares a similar motivation, and aims to disentangle the dialog policy in neural, data-driven dialog systems.

Several approaches have been presented to discover dialog structure graphs (similar to the schemas in this paper) from data in an unsupervised manner (Shi et al., 2019; Qiu et al., 2020; Xu et al., 2020; Hu et al., 2019). These approaches have been used to enhance generation for open-domain dialog (Qiu et al., 2020; Hu et al., 2019). To the best of our knowledge, these dialog structures have neither been used for generation in task-oriented dialog nor in zero-shot settings. While our schemas are similar to these structure graphs, they are hand-crafted similar to those in plan-based dialog systems. Future work may extend our work by leveraging unsupervised structure graph discovery as an alternative to hand-crafted schemas.

3 Task Definition

We address the problem of transferring dialog models to unseen tasks and domains (Zhao and Eskenazi, 2018). This problem is especially important in real world settings. It is impossible to preconceive every dialog task that users may need (e.g., a COVID-19 information dialog system). Furthermore, collecting new dialog data for each new task is inherently unscalable (Rastogi et al., 2020b). While rule-based/pipeline dialog systems may be easier to extend to new tasks (Bohus and Rudnicky, 2009), there is a tradeoff between the adaptability of non-neural systems and the performance of neural models.

3.1 STAR Dataset

The STAR dataset (Mosig et al., 2020) was collected for the purpose of studying transfer learning in dialog. The dataset spans 24 different tasks in 13 different domains (e.g., the restaurant domain has *'restaurant-search'* and *'restaurant-reservations'*). The data collection procedure was designed to reduce ambiguity in the system responses and make system actions deterministic. As such, Amazon Mechanical Turk (AMT) workers were given a flow chart diagram for each task. This flow chart defined the task, including the order in which questions should be asked (e.g., ask date before city), how to respond to various user responses and how to query a database. Additionally, in order to minimize variance in the responses from the wizard, Mosig et al. (2020) incorporate a *suggestions module* during data collection. This module maps the wizard utterance to the closest pre-written response (e.g., *'Give me your name'* $\rightarrow$ *'What is your name?'*). In some cases, it is not possible for the AMT worker to use the suggestions module. Nonetheless, the module increases the consistency of the system actions.

Mosig et al. (2020) present baseline results on the tasks of next action prediction and response generation. The present paper focuses on *next action prediction*. The objective of next action prediction is to predict the correct system action conditioned on the dialog history. Since there is a one-to-one mapping between system actions and corresponding natural language responses, the primary challenge in extending a next action prediction model to response generation resides in learning to accurately fill in the response templates (e.g., *'Your reservation is confirmed for* $\{$*date*$\}$*'*).

The STAR dataset consists of three different types of dialogs: (1) *happy* single-task dialogs, (2) *unhappy* single-task dialogs and (3) *multi-task* dialogs. Here, *happy* refers to dialogs where the users are cooperative and complete the task. In contrast, *unhappy* dialogs consist of uncooperative users that may change the subject, engage in irrelevant chit-chat and otherwise aim to push the system beyond its capabilities. Since our primary objective is to address zero-shot transfer, we only consider the *happy* single-task dialogs. There are 1537 happy single-task dialogs and 10,364 turns.

3.2 Zero-Shot Setting

In the STAR dataset, there are 23 dialog tasks (13 domains) with *happy* single-task dialogs. We perform two types of transfer learning experiments: task transfer and domain transfer. In task transfer, a model is trained on $n-1$ tasks (i.e., 22) and evaluated on the last one. This is repeated for each of the 23 tasks. For domain transfer, a model is trained on $n-1$ domains (i.e., 12) and evaluated on the last one. In task transfer, there may be some overlap between the training and testing, for example, the domain-specific terminology. In contrast, in domain transfer there is very limited overlap. When the model is tasked with generalizing to the restaurant domain, it has seen nothing related to restaurants during training.

In both of these settings, the model is aware of which task it is being evaluated on, meaning that it can leverage a task specification (e.g., schema) for the new task. This experimental design resembles a real-world setting where a system developer would be aware of the new task. For example, if a developer wanted to extend a dialog system to handle a COVID-19 related question, they would be able to create a new task specification. As such, our goal is to develop a model that can generalize to an unseen task conditioned on a task specification.

4 Methods

In order to enable zero-shot transfer to new dialog tasks and domains, the Schema Attention Model (SAM) is introduced. It leverages an external dialog policy representation (i.e., the schema) to predict the next system action. This section begins by describing the baseline model for the task of next action prediction. Next, the schema-guided paradigm is introduced (Figure 1). It includes a graph-based representation of the task-specific schema and SAM, a model that identifies the next system action by attending to a task-specific schema representation.

4.1 Baseline

This section describes the baseline model proposed by Mosig et al. (2020). Given an arbitrary language encoder, denoted as $\mathcal{F}$, the baseline model obtains a vector representation of the dialog history, c. This representation is then passed through a softmax layer to obtain a probability distribution over the actions.

$$\boldsymbol{h} = \mathcal{F}(c) \tag{1}$$

$$P_{\text{clf}} = \text{softmax}(\boldsymbol{W}\boldsymbol{h}^T + \boldsymbol{b}) \tag{2}$$

Throughout this paper, BERT-base (Devlin et al., 2019) is used as the language encoder.

4.2 Schema-Guided Paradigm

Our baseline model simultaneously needs to (1) interpret the dialog context and identify the relevant intents and slots, and (2) learn the task-specific dialog policies (i.e., if the user wants the weather, ask the city) for the different tasks in the training data. This model is incapable of generalizing to a new task in a zero-shot setting, as it would lack knowledge of the task-specific policy for the new task. To mitigate this problem and to enable zero-shot task transfer, we present the schema-guided paradigm which decouples the task-specific dialog policy from the language understanding.

An example is shown in Figure 1: the schema-guided paradigm decouples the the dialog policy from language understanding by explicitly providing task-specific schema graphs as input to the model. These schema graphs serve as complete representations of the dialog policy for a given task. Therefore, while the baseline needs to implicitly learn the dialog policies, a schema-guided model instead learns to leverage the explicit schema graphs. As such, a schema-guided model can generalize to a new task as long as it is provided with the corresponding schema graph.

In this paradigm, the role of the model is to interpret a dialog context and align it to the explicit schema graph. The role of the schema graph is to determine the next action according to the dialog policy. In this manner, the language encoder is being trained for the task of sentence similarity. With the help of pre-trained models, language understanding in a schema-guided paradigm can be considered to be task-agnostic. By decoupling the task-agnostic language understanding and the task-specific dialog policy, the schema-guided paradigm better facilitates zero-shot transfer learning.

The schema-guided paradigm consists of the representation of the schema graph, and a neural model which interprets the dialog context and aligns it to the schema graph.

4.2.1 Schema Representation

In the schema-guided paradigm, the schema representation is the task-specific dialog policy. To ensure the efficacy and robustness of the dialog system, it is important that the schema representation be complete and informative. In the case of ambiguity or incompleteness in the schema representation, the next action will fail to be correctly predicted, regardless of the strength of the model. The schema representations are manually constructed for every task. In the schema-guided paradigm, to transfer to a new task, a system developer would simply need to construct a new schema representation.

Mosig et al. (2020) propose a baseline schema representation wherein the nodes of the graph correspond to system actions and database states. There are nodes for user states only in situations where the system behavior differs depending on the user's actions (e.g., *'Yes'* $\rightarrow$ *ask-time*, *'No'* $\rightarrow$ *ask-date*). The consequence of this representation is that when the model aligns the dialog history to the schema, it largely relies on the system utterances. However, this representation fails to account for realistic user behavior and therefore yields only marginal improvement over the baseline.

Specifically, users will often provide information out of turn (e.g., *System: 'Where would you like to go?'* $\rightarrow$ *User: 'Leaving from the airport and going downtown'*). In this example, it is difficult for the model to realize that the question *System: 'Where are you leaving from?'* has also been answered and therefore should not be the next system action. Users can also ignore the system utterance (e.g., *System: 'Where would you like to go?'* $\rightarrow$ *User: 'Actually, what's the weather?'*). It is thus ineffective to represent dialog policy only in terms of the system utterances. To this end, we extend the schema representation by incorporating user utterances into the schema graph.

As shown in Figure 2, our schema graph incorporates nodes corresponding to user utterances. As such, if a user provides information out of turn or changes the subject, our model will be able to effectively align the dialog to the schema. To account for variance in the user utterances, future work could extend this schema representation to include multiple variations of a given user utterance. However, as the schema graphs are manually constructed for every task, there is a trade-off between manual effort and efficacy[2].

The schema graph has several noteworthy properties. First, the system actions are consistently deterministic. Nodes corresponding to a database response or to a user utterance will always have a single outgoing edge to a system response node.

[2]Constructing the schema graphs is not particularly labor-intensive. It took the first author between 15 and 45 minutes to create each schema graph, depending on the complexity of the task.

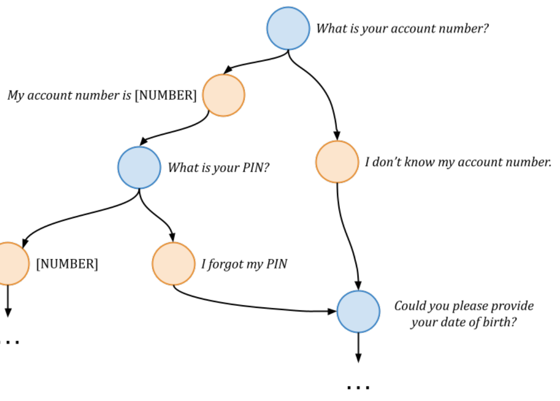
What is your account number?
My account number is [NUMBER]
I don't know my account number.
What is your PIN?
[NUMBER]
I forgot my PIN
Could you please provide your date of birth?
...
...

Figure 2: A section of the task-specific schema graph for the *bank-balance* task. The system must authenticate the user with their account number and PIN. However, if the user has forgotten either of these, it must ask backup security questions. The blue nodes correspond to system actions and the yellow nodes denote user utterances.

Furthermore, such nodes will also have a single incoming edge from a system response node. For a given user/database node, u, we denote the previous system response node as **prev**(u) and the following system response as **next**(u). Each node has some text associated with it, denoted as **text**(u). This text is a template for either a system utterance, database response or user utterance. System nodes will also have an associated system action, **act**(u). There is a one-to-one mapping between the system actions and the system response templates.

4.2.2 Schema Attention Model

In the schema-guided paradigm, the role of the model is to understand the dialog history and align it to the schema representation. We introduce the **Schema Attention Model**, SAM, which attends between the dialog history, $c = c_1, \ldots, c_N$ and the schema graph. SAM extends the schema-guided model presented by Mosig et al. (2020) by (1) leveraging a stronger attention mechanism, (2) improving the training algorithm, and (3) removing the linear classification layer which is detrimental to zero-shot performance.

The objective of SAM is to predict the node in the schema graph that best corresponds to the dialog context. SAM will produce a probability distribution over the nodes corresponding to user utterances and database responses. Given an attention distribution over the nodes, we can obtain a probability distribution over the set of actions by propagating the attention probabilities over the graph.

Concretely, if node u has an attention weight of p, we add p to the probability of **action**(**next**(u)).

We consider every node u that corresponds to either a database response or a user utterance. We then represent each node u as the concatenation of the previous node and the current node, i.e., **text**(**prev**(u)) + **text**(u). For all nodes $u \in U$, we obtain this textual representation denoted as $s \in S$.

We are given a language encoder, $\mathcal{F}$, the dialog context, $c = c_1, \ldots, c_N$, the nodes U, their corresponding textual representations S, and the set of possible actions A. Note that unlike in Equation 1, $\mathcal{F}$ is used to produce a vector representation of each word in the input. SAM produces a probability distribution over the actions as follows:

$$\boldsymbol{h}_{1,\ldots,N} = \mathcal{F}(c\colon c_1, \ldots, c_N) \tag{3}$$

$$\boldsymbol{S}_{i;1,\ldots,M} = \mathcal{F}(S_i\colon s_1, \ldots, s_M) \tag{4}$$

$$\boldsymbol{w}^i_{j,k} = \boldsymbol{h}_j^T \boldsymbol{S}_{i;k} \tag{5}$$

$$\alpha = \text{softmax}(\boldsymbol{w}^{1,\ldots,|S|}) \tag{6}$$

$$p_i = \sum_{j \le N} \sum_{k \le M} \alpha^i_{j,k} \tag{7}$$

Here, $\boldsymbol{w}^i$ is an $N \times M$ dimensional matrix corresponding to the dot product between the N words of the dialog history and the M words of the i-th textual representation in S. To get the attention weights over all of the words of the schema, we perform a softmax over all $\boldsymbol{w}^i, 1 \le i \le |S|$. By summing over the attention weights in α^i, we get p_i, a scalar value which denotes the attention between the dialog history and the i-th node (i.e., the corresponding textual representation S_i). Given p_i we produce a probability distribution over the actions A as follows:

$$g(i,a) = \begin{cases} p_i, & \text{if } \textbf{action}(\textbf{next}(u_i)) = a \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$P(a) = \sum_{i \le |S|} g(i,a) \tag{9}$$

To align the dialog history to the schema graph, SAM performs word-level attention using a BERT-base model. In contrast, the schema-guided model of Mosig et al. (2020) attends with the sentence level vector representation produced by BERT. With the word-level attention, SAM can better align ambiguous dialog contexts, such as situations where the user provides multiple pieces of information in a single utterance. Since this word-level attention operates on the sub-word tokens used in BERT, it can also potentially handle spelling errors in the user utterances.

Furthermore, in their schema-guided model, Mosig et al. (2020) combine the probability distribution produced by attending to the schema graph with their baseline model (i.e., Section 3.1). While this may result in better performance on the tasks the model is trained with, the baseline model will not generalize to unseen tasks. In contrast, SAM computes the probability for an action using only the attention over the schema graph.

Mosig et al. (2020) train their schema-guided model to predict the appropriate node, u_i, from a set of nodes U' (s.t., $U' \subset U$). At training time, for efficiency reasons, the set of nodes U' is obtained by using the corresponding node for every dialog context in the training batch. Since the training batches are randomly sampled, this results in U' including nodes from a variety of different schema graphs. At inference time, the dialog task is known and therefore only the corresponding schema graph needs to be attended to (i.e., U' will contain nodes from a single schema graph). It is valuable to train the model to distinguish between different nodes of the same schema graph. Specifically, the attention mechanism (i.e., Equations 5 - 6) will learn stronger fine-grained relationships when trained with negative samples from the *same* domain. As such, we augment the training algorithm to sample batches from the same dialog task, meaning that U' will only include nodes from a single schema.

SAM improves on the baseline schema-guided model introduced by Mosig et al. (2020) by (1) leveraging a stronger attention mechanism that better handles realistic user behavior, (2) computing a probability distribution *only* by attending to the schema graph and (3) modifying the training algorithm to have in-domain negative samples which result in the model learning to identify fine-grained relationships. In combination with the improved schema representation, SAM is better suited to handle realistic user behavior in zero-shot settings.

5 Results

To validate the effectiveness of SAM, a number of *next action prediction* experiments are carried out on the STAR dataset (Mosig et al., 2020). First, SAM is evaluated in the standard experimental set-

Model	F_1 score	Accuracy
Baseline ◇	**73.79**	**74.85**
BERT+S ◇	71.59	72.27
SAM − [1]	54.35	60.51
SAM − [2,3,4]	70.22	71.01
SAM − [2]	70.27	71.93
SAM − [3]	70.18	71.64
SAM − [4]	69.68	69.79
SAM	70.38	71.45

Table 1: Performance in the standard experimental setting. Models marked with ◇ are attributed to Mosig et al. (2020). We denote their schema-guided model, 'BERT + *Schema*', as BERT+S. SAM consists of four improvements upon BERT+S: (1) user-aware schema, (2) word-level attention, (3) using negative samples from the same task at training, (4) removing the linear classification layer. Results in boldface are statistically significant by t-test ($p < 0.01$)

ting, i.e., training and testing on the same tasks. Next, we carry out zero-shot transfer experiments, as defined in Section 3. The evaluation uses accuracy and weighted F_1 score.

We rerun the experiments presented by Mosig et al. (2020) using code shared by the authors. In our results, the model introduced by Mosig et al. (2020) is denoted as BERT+S. Their original results were obtained on an older version of STAR, with annotation errors[3] that have since been fixed.

5.1 Standard Experiments

In the standard experimental setting, models are trained and tested on the same tasks. Following Mosig et al. (2020), 80% of the dialogs are used for training and 20% for testing. All models are trained for 50 epochs.

The results shown in Table 1 show SAM to be comparable to the baseline model on the standard setting. Since the augmentations to SAM are primarily intended to improve zero-shot performance, it is unsurprising that there is no performance improvement compared to the standard setting. When evaluating on *seen* tasks, the linear classification layer is significantly more effective than attending to the schema. This suggests that a large neural model (i.e., BERT) is able to implicitly learn meaningful dialog policies from dialog data. It is possible that this performance difference may decrease with more expressive schemas (e.g., having multiple examples for each user utterance, automatically learning schemas from the dataset). The value of our schema graphs is nonetheless shown when comparing SAM to SAM$-[1]$ (i.e., the old schema graphs). These experiments provide an upper bound for the performance in zero-shot transfer.

[3]Specifically, certain dialogs were misattributed as being *happy* single-task dialogs.

5.2 Zero-Shot Transfer

Table 2 shows the results of the zero-shot experiments. SAM obtains strong improvements over the baseline models for both zero-shot task transfer and domain transfer. These experimental results validate the effectiveness of the schema-guided paradigm, as well as the specific design of SAM.

Compared to the baseline model (described in Section 3.1), SAM obtains a $+\mathbf{22}\ \mathbf{F_1}$ score improvement in task transfer and a $+\mathbf{24}\ \mathbf{F_1}$ score improvement in domain transfer. Since the baseline model is unable to predict classes it has not observed at training time, its performance is limited to actions that are consistent across domains (e.g., '*hello*', '*goodbye*', '*anything-else*'). This improvement highlights the effectiveness of the schema-guided paradigm for zero-shot transfer learning.

BERT+S also leverages schemas for transfer learning. Yet, it under-performs relative to the baseline model. SAM attains even larger improvements over this baseline schema-guided model. As described in Section 4.2, the weak performance of BERT+S is largely a consequence of it being incapable of handling realistic user behavior. The design of BERT+S (i.e., the schema only having system nodes) results in the model essentially predicting the subsequent system actions. This is equivalent to sequentially predicting the next system action, regardless of user behavior. With improved schema representations and model architecture, SAM achieves much stronger performance in zero-shot transfer.

Our ablation experiments shed more light on the performance of SAM relative to BERT+S. A significant performance drop is observed when removing the newly constructed schema representations (i.e., SAM$-[1]$). In contrast, adding the schema graphs to BERT+S (i.e., SAM$-[2,3,4]$) results in a strong performance improvement of $+\mathbf{15}\ \mathbf{F_1}$ score. This confirms the hypothesis that the schema graphs of Mosig et al. (2020), which are largely comprised of system action nodes are insufficient for modelling

Model	Task Transfer		Domain Transfer	
	F_1 score	Accuracy	F_1 score	Accuracy
Baseline ♢	31.23	30.65	31.82	33.92
BERT+S ♢	28.12	28.28	29.70	32.43
SAM − [1]	33.81	37.84	41.77	45.64
SAM − [2,3,4]	43.28	46.11	43.78	45.19
SAM − [2]	50.72	53.69	52.20	54.68
SAM − [3]	45.54	49.29	50.56	52.13
SAM − [4]	47.26	47.99	47.67	48.92
SAM	**53.31**	**55.51**	**55.74**	**57.75**

Table 2: Performance in zero-shot transfer. We present results on both task transfer and domain transfer. Models marked with ♢ are attributed to Mosig et al. (2020). SAM consists of four improvements upon BERT+S: (1) user-aware schema, (2) word-level attention, (3) using negative samples from the same task at training, (4) removing the linear classification layer. Results in bold-face are statistically significant by t-test ($p < 0.01$).

realistic user behavior.

Word-level attention is shown to give moderate, albeit statistically significant, improvement. In contrast to SAM−[2], SAM obtains a **+3 $\mathbf{F_1}$** score improvement. While word-level attention allows the model to better align the dialog to the schema, it is an architectural improvement that is not central to the schema-guided paradigm.

Modifying the training algorithm to sample batches from the same task results in better negative samples during training. This allows the model to learn to distinguish between nodes from the same schema graph when aligning the dialog to the schema graph. When this modification is removed (i.e., SAM−[3]), the performance of SAM drops by **8 $\mathbf{F_1}$** score for zero-shot task transfer.

The fourth and final component of SAM is the removal of the linear classification layer. Since this classification layer is unable to predict classes it has not seen at training time, it is ineffective in zero-shot settings. Unsurprisingly, removing it increases performance and SAM obtains a **+6 $\mathbf{F_1}$** score improvement over SAM−[4].

The zero-shot experiments shown in Table 2 empirically validate several hypotheses presented in this paper. First, the strong improvement over the baseline demonstrates the efficacy of the schema-guided paradigm for zero-shot generalizability in end-to-end dialog. Decoupling dialog policy and the language understanding by explicitly representing the task-specific dialog policies as schema graphs results in an improved ability to transfer to unseen tasks. Next, we improve over the schema-guided model of Mosig et al. (2020) through (1) an improved schema representation and (2) a collection of modifications to the model. The improved schema representation better models realistic user behaviors in dialog, and therefore results in better alignment of the dialog and the schema. Our model modifications result in the model being able to learn better fine-grained relationships during alignment (e.g., through better negative sampling and word-level attention) and better handle zero-shot transfer (e.g., by removing the linear layer).

In contrast to prior work on zero-shot generalizability (Zhao and Eskenazi, 2018; Qian and Yu, 2019), our approach is shown to effectively transfer between the vastly dissimilar domains of the STAR corpus (Mosig et al., 2020) (e.g., trivia or spaceship maintenance). Rather than modelling a cross-domain mapping and leveraging similar concepts across different domains, the schema-guided paradigm *decouples* the domain-specific (i.e., the dialog policy) and domain-agnostic (i.e., language understanding) aspects of dialog systems. Through the schema-guided paradigm, we achieve strong performance in the zero-shot setting and take an important step towards zero-shot dialog.

6 Conclusion

This paper shows strong results in zero-shot task transfer and domain transfer using the schema-guided paradigm. We hypothesized that the difficulty of zero-shot transfer in dialog stems from the dialog policy. When neural models implicitly memorize dialog policies observed at training time, they struggle to transfer to new tasks. To mitigate this, we explicitly provide the dialog policy to the model, in the form of a schema graph. This paper introduces the Schema Attention Model (SAM) and shows improved schema graphs for the STAR corpus. This approach attains significant improvement over prior work in the zero-shot setting, with a **+22 F_1 score improvement**. Furthermore, the ablation experiments demonstrate the effectiveness of both SAM and the improved schema representations. Future work may explore (1) improved schema representations to better capture dialog policy, (2) improved model architectures to better align the dialog to the schema, and (3) extensions to other problems (e.g., response generation).

References

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. *arXiv preprint arXiv:1707.02363*.

Dan Bohus and Alexander I Rudnicky. 2009. The ravenclaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23(3):332–361.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6045–6049. IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

George Ferguson and James Allen. 1998. Trips: An intelligent integrated problem-solving assistant. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 567–573.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Wenpeng Hu, Zhangming Chan, Bing Liu, Dongyan Zhao, Jinwen Ma, and Rui Yan. 2019. Gsn: A graph-structured network for multi-party dialogues. *arXiv preprint arXiv:1905.13637*.

Dan Jurafsky. 2000. *Speech & language processing*. Pearson Education India.

Shikib Mehri, Mihail Eric, and Dilek Hakkani-Tur. 2020. Dialoglue: A natural language understanding benchmark for task-oriented dialogue. *arXiv preprint arXiv:2009.13570*.

Johannes EM Mosig, Shikib Mehri, and Thomas Kober. 2020. Star: A schema-guided dialog dataset for transfer learning. *arXiv preprint arXiv:2010.11853*.

Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. *arXiv preprint arXiv:1906.03520*.

Liang Qiu, Yizhou Zhao, Weiyan Shi, Yuan Liang, Feng Shi, Tao Yuan, Zhou Yu, and Song-Chun Zhu. 2020. Structured attention for unsupervised dialogue structure induction. *arXiv preprint arXiv:2009.08552*.

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

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020a. Schema-guided dialogue state tracking task at dstc8. *arXiv preprint arXiv:2002.01359*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020b. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Let's go public! taking a spoken dialog system to the real world. In *Ninth European conference on speech communication and technology*.

Charles Rich and Candace L Sidner. 1998. Collagen: A collaboration manager for software interface agents. In *Computational models of mixed-initiative interaction*, pages 149–184. Springer.

Darsh J Shah, Raghav Gupta, Amir A Fayazi, and Dilek Hakkani-Tur. 2019. Robust zero-shot cross-domain slot filling with example values. *arXiv preprint arXiv:1906.06870*.

Weiyan Shi, Tiancheng Zhao, and Zhou Yu. 2019. Unsupervised dialog structure learning. *arXiv preprint arXiv:1904.03736*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. *arXiv preprint arXiv:1905.08743*.

Jun Xu, Zeyang Lei, Haifeng Wang, Zheng-Yu Niu, Hua Wu, Wanxiang Che, and Ting Liu. 2020. Discovering dialog structure graph for open-domain dialog generation. *arXiv preprint arXiv:2012.15543*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

Tiancheng Zhao and Maxine Eskenazi. 2018. Zero-shot dialog generation with cross-domain latent actions. *arXiv preprint arXiv:1805.04803*.

Tiancheng Zhao, Allen Lu, Kyusong Lee, and Maxine Eskenazi. 2017. Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability. *arXiv preprint arXiv:1706.08476*.

A Example Outputs

We show a few example outputs in Table 3. These examples are model outputs produced in the zero-shot domain transfer experiments. None of these models have seen any training data pertaining to the specific domains. These examples demonstrate the effectiveness of SAM to handle ambiguous user intents and effectively align them to the dialog schema.

B Example Schemas

All of the schema graphs produced for this paper are available in our GitHub Repository[4]. These schemas improve upon the schemas provided by Mosig et al. (2020). Nonetheless, this schema representation is still far from perfect. Future work may explore designing more expressive and thorough schemas. There are several approaches for doing this: (1) having multiple examples for each user utterance, (2) better representing loops in the schema graphs (e.g., query the database until you find the cheapest flight), (3) representing more sophisticated (potentially dynamic) scenarios (e.g., if the user's answer matches the correct answer to a trivia question, inform them that they are correct). The proposed schema representations address several weaknesses, however there is still significant room for improvement.

[4] https://github.com/Shikib/schema_attention_model

Domain:	Spaceship
Task:	Spaceship Access Codes
Dialog History:	USER: The ship is under attack! Need help ASAP SYSTEM: Please provide your name. USER: Rey. I'm the first officer SYSTEM: Please enter the code. USER: I think it's 9315831
Baseline Output:	Query
BERT+S Output:	Query
SAM Output:	Spaceship Ask Code Type (*'Please specify the code type.'*)
SAM Schema Node:	SYSTEM: Please enter the code. USER: [NUMBER]
Domain:	Bank
Task:	Check Bank Balance
Dialog History:	USER: Hi, can you help me check my balance? SYSTEM: Could I get your full name, please? USER: Jane Doe SYSTEM: Could you tell me your account number, please? USER: I don't remember it or my PIN number unfortunately.
Baseline Output:	Query
BERT+S Output:	Inform Cannot Authenticate (*'I am sorry, but I cannot authenticate you with the information you have provided.'*)
SAM Output:	Ask Date of Birth (*'Could you provide your date of birth, please?'*)
SAM Schema Node:	SYSTEM: Could you tell me your account number, please? USER: I don't know my account number.

Table 3: Example outputs in the zero-shot domain transfer setting. The models have not seen any data pertaining to the domains that are tested here. The first example shows that SAM can effectively align to the schema. The second example is more complicated and requires SAM to identify that the user indicated that they don't remember their account number (note that the most typical system response following *'Can you tell me your account number, please?')* is asking for the user's PIN.