Workflow Discovery from Dialogues in the Low Data Regime

Anonymous authors
Paper under double-blind review

Abstract

Text-based dialogues are now widely used to solve real-world problems. In cases where solution strategies are already known, they can sometimes be codified into *workflows* and used to guide humans or artificial agents through the task of helping clients. We introduce a new problem formulation that we call Workflow Discovery (WD) in which we are interested in the situation where a formal workflow may not yet exist. Still, we wish to discover the set of actions that have been taken to resolve a particular problem. We also examine a sequence-to-sequence (Seq2Seq) approach for this novel task. We present experiments where we extract workflows from dialogues in the Action-Based Conversations Dataset (ABCD). Since the ABCD dialogues follow known workflows to guide agents, we can evaluate our ability to extract such workflows using ground truth sequences of actions. We propose and evaluate an approach that conditions models on the set of possible actions, and we show that using this strategy, we can improve WD performance. Our conditioning approach also improves zero-shot and few-shot WD performance when transferring learned models to unseen domains within and across datasets. Further, a modified variant of our Seq2Seq method achieves state-of-the-art performance on related but different problems of Action State Tracking (AST) and Cascading Dialogue Success (CDS) on the ABCD dataset.

1 Introduction

Task-oriented dialogues are ubiquitous in everyday life and customer service in particular. Customer support agents use dialogue to help customers shop online, make travel plans, and receive assistance for complex problems. Behind these dialogues, there could be either implicit or explicit *workflows* – actions that the agent has followed to ensure the customer request is adequately addressed. For example, booking an airline ticket might comply with the following workflow: pull up an account, register a seat, and request payment. Services with no formal workflows struggle to handle variations in how a particular issue is resolved, especially for cases where customer support agents tend to follow "unwritten rules" that differ from one agent to another, significantly affecting customer satisfaction and making training new agents more difficult. However, correctly identifying each action constituting a workflow can require significant domain expertise, especially when the set of possible actions and procedures may change over time. For instance,

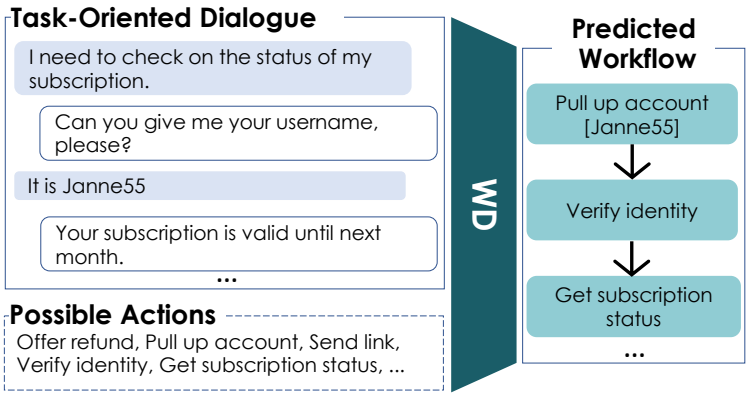


Figure 1: Our method for extracting workflows from dialogues. The input consists of the dialogue utterances and the list of possible actions if available. The output is the workflow‘ followed to resolve a specific task, consisting of the actions (e.g., Verify identity) and their slot values (e.g., Janne55).

newly added items or an update to the returns policy in an online shopping service may require modifying the established workflows.

In this work, we focus on "workflow discovery" (WD) – the extraction of workflows that have either implicitly or explicitly guided task-oriented dialogues between two people. Workflows extracted from a conversation consist of a summary of the key actions taken during the dialogue. These workflows consist of pre-defined terms for actions and slots when possible, but our approach also allows for actions that are not known to be invented by the model online and used as new steps in the generated workflow. Our approach is targeted toward the task of analyzing chat transcripts between real people, extracting workflows from transcripts, and using the extracted workflows to help design automated dialogue systems or to guide systems that are already operational. Alternatively, extracted workflows might also be used for human agent training. One might imagine many scenarios where an analyst might use WD to understand if an unresolved problem is due to a divergence from a formal workflow or to understand if workflows have organically emerged, even in cases where no formal workflow documentation exists. WD is related to the well-established field of process mining, but process mining typically extracts workflow information from event logs as opposed to unstructured dialogues encoded as natural language. Our work shows that traditional process mining can be dramatically expanded using modern NLP and sequence modeling techniques, as we propose and explore here. Our approach correspondingly allows traditional event logging information to be combined with NLP allowing systems constructed using our approach to benefit from recent advances in large language models (LLMs), including enhanced zero-shot and few-shot learning performance.

To address the challenges of WD introduced by dynamically changing actions, distributional shifts, and the challenges of transferring to completely new domains with limited labeled data, we propose a text-to-text approach that can output the workflow consisting of the full set of actions and slot values from a task-oriented dialogue. Further, to better adapt to unseen domains, we condition our method on the full or partial set of possible actions as shown in Figure 2, enabling us to perform zero-shot and few-shot learning. Figure 2 also illustrates the type of generalization possible with our approach in these regimes where we can propose new actions never seen during training. We investigate four scenarios for extracting workflows from dialogues under differing degrees of distributional shift: (1) *In-Domain workflows*, where all workflow actions have been seen during training. (2) *Cross-domain zero-shot*, where actions from selected domains have been omitted during training but are present in the valid and test sets, (3) *Cross-dataset zero-shot*, where a model trained on all the domains of one dataset is applied to another domain in a zero-shot setting, and (4) *Cross-dataset few-shot*, where a model is trained on all the domains of one dataset, then trained on another domain is a few-shot setting.

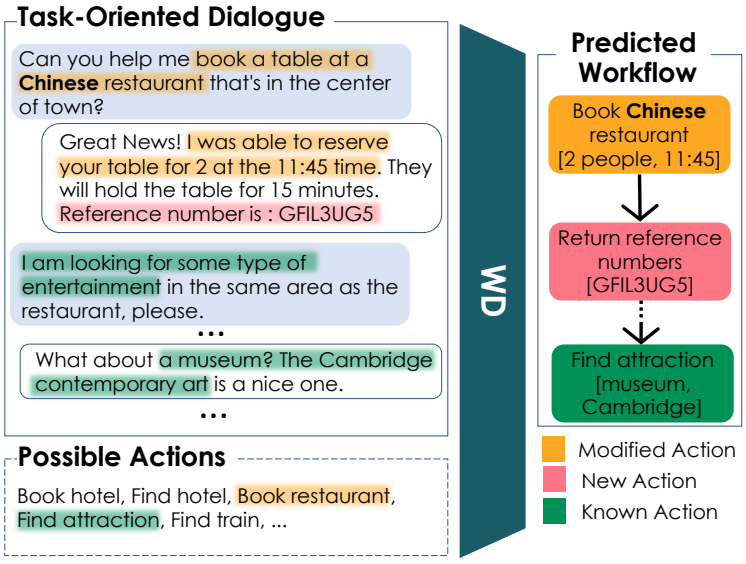


Figure 2: Our conditioning approach allows for better zero-shot and few-shot performance. Entirely new workflow actions (i.e, not in the list of possible actions) can be proposed, as well as those based on minor modifications to known actions.

Our contributions can be summarized as follows:

- We propose a new formulation to the problem of workflow extraction from dialogues, which, to our knowledge, has not been examined or framed as we present here. We cast the problem as summarizing dialogues with workflows and call the task Workflow Discovery (WD).

- We propose a text-to-text approach to solve the WD task that takes entire dialogues as input and outputs workflows consisting of sequences of actions and slot values that can be used to guide future dialogues. We test our approach using various state-of-the-art text-to-text models and show its efficacy on the Action Based Conversations Dataset (ABCD) (Chen et al., 2021).
- We propose a conditioning mechanism for our text-to-text approach, providing the model with a set of possible actions to use as potential candidates. We show that this mechanism allows our method to be used in dramatically different domains from the MultiWOZ dataset (Budzianowski et al., 2018), yielding good cross-dataset zero-shot workflow extraction performance. Moreover, it allows an important performance increase in the cross-dataset few-shot setting.
- Using a variant of our approach, we achieve state-of-the-art results on related but different and more standard tasks of Action State Tracking (AST) and Cascading Dialogue Success (CDS) on the ABCD evaluation.

2 Related Work

Task-Oriented Dialogues The goal of task-oriented dialogue systems is to assist users in completing certain tasks for specific domains, such as restaurant and travel bookings, as exemplified by the MultiWOZ dialogues of Budzianowski et al. (2018) and Zang et al. (2020). In the more recent ABCD (Chen et al., 2021), dialogues have been captured around a retail theme in which tasks such as seeking a refund or updating addresses can only be accomplished when relatively long sequences of actions have been successfully accomplished. In this work, we define a new problem for task-oriented dialogue where our goal is to identify or extract a set of actions constituting a workflow from a dialogue consisting of the interactions between a human agent and a customer. We focus on achieving this goal for new domains in the zero-shot and few-shot learning setting.

Sequence-to-Sequence Text Generation In this work, we use Encoder-Decoder-based sequence-to-sequence models since we wish to take a whole dialogue as input and output a workflow. Our work is related to models developed for translation (Sutskever et al., 2014) and summarization (Gliwa et al., 2019; Zhang et al., 2020), with summarization being the closer of the two. However, here we define and examine a new problem entirely – where dialogues are transformed into sequences of steps that an agent has used to solve a problem, using a special vocabulary for workflow action steps.

Summarization Models State-of-the-art summarization methods include PEGASUS (Zhang et al., 2020) and BART (Lewis et al., 2019), both of which are based on encoder-decoder transformers. With PEGASUS, models are trained by removing or masking key sentences from an input document and training models to generate a single output sequence containing the concatenation of the masked content, similar to extractive summarization. The BART approach uses denoising autoencoder pretraining. The classical formulations of extractive and abstractive summarization (e.g., for news and scientific papers) are different from our setting here. However, since PEGASUS, BART and a recent T5-based summarization model available from HuggingFace all yield strong performance for classical summarization tasks, we build upon and fine-tune these base models for our workflow discovery problem setting and experiments.

Process Mining & Discovery The goal of process mining is to extract processes from event logs. Influential prior work includes that of Van der Aalst et al. (2004), which examines a series of event logs and generates workflow networks (WF-nets), which are special types of Petri networks for describing workflows (Van Der Aalst & Dongen, 2013). Their approach is unsupervised whereas the more recent work of Sommers et al. (2021) trained graph convolutional neural networks on synthetic data to produce Petri nets from event logs in a supervised manner. To the best of our knowledge, our work here is the first to use a large language model for workflow discovery.

3 Workflow Discovery (WD)

We propose a novel task we call Workflow Discovery (WD) to extract the actual workflow followed by an agent from a task-oriented dialogue. A workflow is a *sequence* of *actions* with their respective slot values in the same order in which they appeared during the conversation. Formally, given a dialogue $D = \{u_1, u_2, ..., u_n\}$, where

n is the total number of utterances in the dialogue, and an *optional* list of possible actions $\delta = \{a_1, ..., a_z\}$, where z is the number of known actions, and a_z is a unique workflow action. A model should predict the target workflow $W = \{(a_1, \{v_1^j | 0 <= j <= n_1\}), ..., (a_k, \{v_k^i | 0 <= j <= n_k\})\}$, where $a_i \in \delta$, v_i^j and n_i are the j^{th} slot value and the number of available slot values respectively for action a_i, and k is the number of workflow actions.

Workflow Discovery differs from dialogue state tracking (DST) where in WD, we are interested in extracting the sequence of actions that have been followed to resolve an issue. We are particularly interested in the situation where the actions that are needed to resolve a problem associated with a given intent or task *are not known a priori* and must be invented by the WD model. This is in sharp contrast to DST which generally requires dialogue states to be known and pre-defined. DST also generally focuses on tracking the state of one party in the conversation, e.g., the user. In contrast, WD extracts actions executed by the agent and slot values collected from user utterances. Furthermore, WD differs from Action State Tracking (AST), since WD aims to extract *all* actions and their matching slot values, using *all* dialogue utterances without any extra metadata like the annotated action turns at once (no online tracking). In contrast, AST predicts the next action at a given turn, given the previous turns only. Models trained for AST help agents select the next action following the agent guidelines. In contrast, models trained for WD extract all actions from chat logs between real people, including those that might deviate from the agent guidelines. WD is aimed at agent training or workflow mining with the goal of formalizing the discovered processes for subsequent automation. WD can be applied to completely new, organic tasks where possible actions are unknown. Some aspects of AST could be seen as a subset of WD when actions and slots are known. However, we believe that WD is more closely related to the concept of summarization using a specialized vocabulary for actions and slots, when possible, but inventing new terms when needed.

We compare and contrast WD, AST, and DST in Figure 3. We note that Figure 3a shows a situation where our WD approach has succeeded in inventing a new term (i.e., "Generate new code") which describes the exact action the agent performed. This result matches the agent guidelines (Chen et al., 2021), where there is a clear distinction between offering a promo code when a customer is unhappy in which our approach uses the known "promo code" action, and generating a new code since the old one is no longer working. A good WD method should exhibit compositional generalization when creating new terms for actions and their arguments. For example, if a system has been trained to summarize dialogues that involve granting a refund for ordering a pair of pants, the WD summary of a new dialogue involving the refund of a shirt should use a vocabulary consistent with the manner in which the refund of a pair of pants has been expressed.

Finally, in contrast to policy learning, WD aims to extract a workflow from dialogue logs describing the sequence of agent actions that depend on slot values extracted from user utterances. The extracted sequence may differ significantly from the actions of an optimal or estimated policy. WD focuses on extracting potentially out-of-distribution workflows, where new action names and slot values derived from new domains must be invented. This makes WD very different from policy learning and more similar to dialogue policy act extraction, but where new acts and slot values must be invented on the fly to summarize new concepts at test time.

4 Methodology

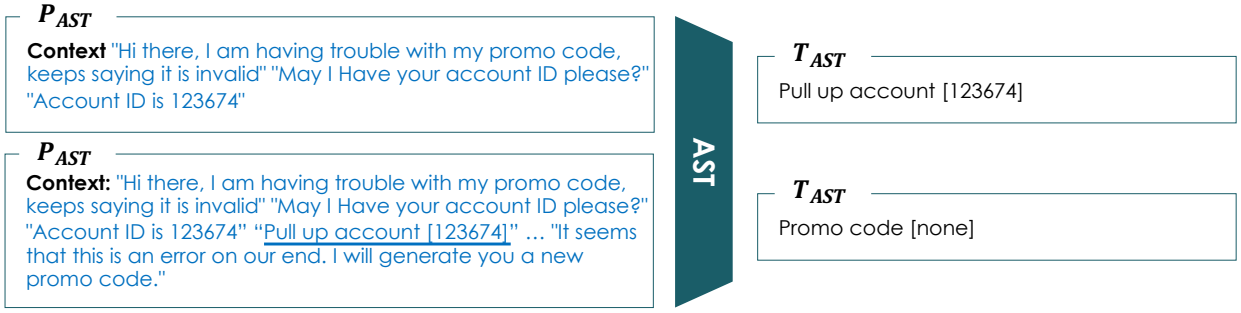
In this section, we describe our methodology for the WD task. Moreover, since there is no existing baseline for our novel WD task, we included our text-to-text variants for both the AST and CDS tasks that we used to test our text-to-text task casting scheme against existing benchmarks.

4.1 Text-to-Text Workflow Discovery

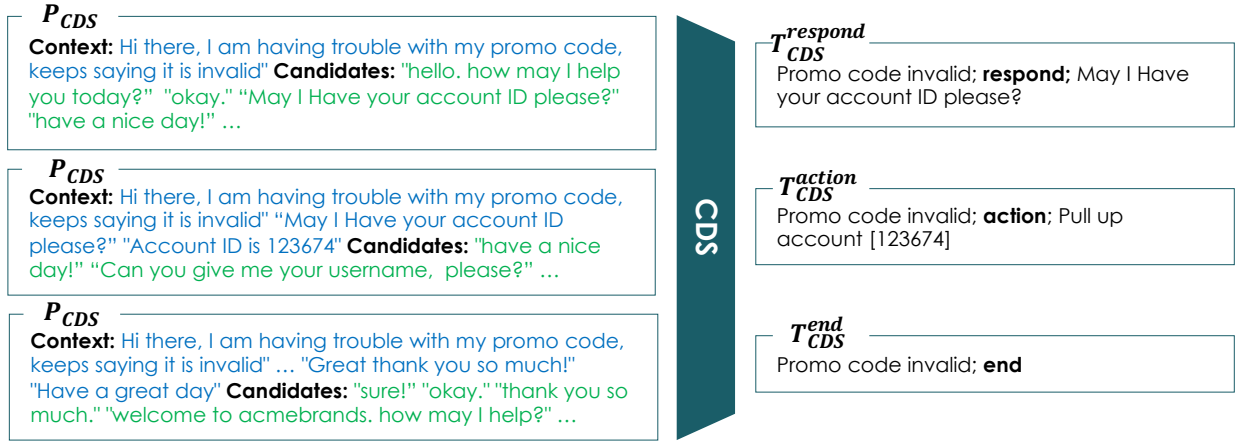
We cast the WD task as a text-to-text task where the input of the model P_{WD} consists of all utterances and the list of possible actions, if available or partially available, formatted as

$$P_{WD} = \text{Dialogue: } u_1, ..., u_n \text{ Actions: } a_1, ..., a_z$$

(a) **Workflow Discovery (WD)**. Illustration of a prompt (P_{WD}) – consisting of an entire dialogue, and the target (T_{WD}) – consisting of a structured summary for the WD task. P_{WD} is composed of the delimiters in bold that separate the dialogue utterances in blue and the optional list of possible actions in green. T_{WD} is composed of the predicted workflow actions, and matching slot values in brackets. The actions are either known (highlighted in green) or invented (highlighted in orange).



(b) **Action State Tracking (AST)**. Illustration of prompts (P_{AST}) and targets (T_{AST}) for the AST task at different turns in the dialogue. P_{AST} is composed of the context delimiter in bold and the context utterances in blue that can contain previous action turn text (underlined).



(c) **Cascading Dialogue Success (CDS)**. Illustration of prompts (P_{CDS}) and targets T_{CDS} for the CDS task at different turns in the dialogue. P_{CDS} is composed of the delimiters in bold that separate the dialogue utterances in blue and the list of possible agent response candidates in green. $T_{CDS}^{respond}$, T_{CDS}^{action}, and T_{CDS}^{end} are the targets when the next step is to **respond** with an utterance, perform an **action**, and **end** the conversation, respectively. "Promo code invalid" is the entire dialogue intent.

Figure 3: Illustration of prompt and target formats for WD, AST, and CDS tasks.

where u_n is a dialogue utterance, n is the total number of utterances in the dialogue, a_z is a workflow action, and z is the total number of available actions. "*Dialogue*:" and "*Actions*:" are delimiters. We omit the last delimiter if the possible actions are not available. Finally, We use the source prefix "*Extract workflow*:" The target workflow T_{WD} is formatted as

$$T_{WD} = a_1[v_1^1, ..., v_1^{n_1}]; ...; a_k[v_k^1, ..., v_k^{n_k}]$$

where a_k is a workflow action, k is the number of actions, $v_k^{n_k}$ is a slot value, and n_k is the number of slot values for action k. "[" and "]" encapsulate the slot values. "," and ";" are the separators for the slot values and actions. Moreover, if an action has no slot values, the target slot value list is set explicitly to $[none]$. An example of this casting scheme is shown in Figure 3a. To make the model invariant to the position and the number of workflow actions, especially when adapting to a new domain in the zero-shot and few-shot settings. We set the possible actions to a *shuffled* set comprised of the currents sample target actions and a randomly chosen number $r_{min} <= r <= z$ of actions while training the models in the in-domain setting *only*, where r_{min} is a hyper-parameter. For all other settings, the list of possible actions is not modified.

Our text-to-text framework differs from other work since our prompts don't include slot descriptions or value examples compared to Zhao et al. (2022a). Moreover, Adding lists of possible actions to the prompt makes our method more flexible in the zero-shot setup (allowing new actions to be added on-the-fly) and improves performance in the few-shot setup. Finally, we don't prefix utterances with speaker prefixes (e.g., "User: ") compared to Zhao et al. (2022a); Lin et al. (2021a), making our technique usable when this information is unavailable or inaccurate; for example, in cases where the chat transcript originated from an upstream text-to-speech model.

4.2 Text-to-Text Action State Tracking

The goal of the AST task is to predict a *single* action and its slot values given only the previous turns. This task is similar to the traditional DST with the difference that agent guidelines constrain the target actions. For example, an utterance might suggest that the target action is "validate-purchase", but the agent's proposed gold truth is "verify-identity" per the agent guideline because an agent needs to verify a customer's identity before validating any purchase.

We cast the AST task as a text-to-text task where the input of the model P_{AST} consists of all the utterances formatted as

$$P_{AST} = \text{Context: } u_1, ..., u_t$$

where u_t is a dialogue turn, including action turns, and t is the index of the current turn. "Context:" is a delimiter. We use the source prefix "$Predict\ AST$: ". The target T_{AST} is formatted as

$$T_{AST} = a_t[v_t^1, ..., v_t^m]$$

where a_t is an action, v^m is a slot value, and m is the number of slot values. "[" and "]" encapsulate the slot values. "," is the separator for the slot values. Moreover, if an action has no slot values, the target slot value list is set explicitly to $[none]$. An example of this casting scheme is shown in Figure 3b.

4.3 Text-to-Text Cascading Dialogue Success

The CDS task aims to evaluate a model's ability to predict the dialogue intent and the next step given the previous utterances. The next step can be to take action, respond with a text utterance, or end the conversation. Moreover, the CDS task is evaluated over successive turns. In contrast, AST assumes that the current turn is an action turn and is evaluated over individual turns.

We cast the CDS task as a text-to-text task where the model input P is formatted as

$$P_{CDS} = \text{Context: } u_1, ..., u_t \text{ Candidates:} c_1, ..., c_v$$

where u_t is a dialogue turn, including action turns, and t is the index of the current turn. $c_v \in C_t$ is a text utterance from the current agent response candidate list C_t, and v is the size of C_t. "Context:" and "Candidates:" are delimiters. Finally, We use the source prefix "$Predict\ CDS$: ". The target T_{CDS} is formatted differently depending on the target next step. When the next step is to take an action, the target is formatted as follows

$$T_{CDS}^{action} = i;\ action;\ a_t[v_t^1, ..., v_t^m]$$

where i is the dialogue intent, and values $a_t[v_t^1, ..., v_t^m]$ is the step name and slots similar to the AST task formulation above. When the next step is to respond, the target is formatted as follows

$$T_{CDS}^{respond} = i;\ respond;\ c_{t+1}$$

where $c_{t+1} \in C_t$ is the expected response utterance. When the next step is to terminate the dialogue, the target is formatted as follows

$$T_{CDS}^{end} = i;\ end$$

An example of this casting scheme is shown in Figure 3c.

5 Experimental Setup

5.1 Implementation Details

In our experimentation, we used the T5 (Raffel et al., 2020), BART (Lewis et al., 2019) and PEGASUS (Zhang et al., 2020) models for the WD task, which we call WD-T5, WD-BART, and WD-PEGASUS, respectively. Furthermore, we use a T5 model for the text-to-text AST, and CDS tasks, which we call, AST-T5, and CDS-T5, respectively. We use the Huggingface Transformers Pytorch implementation[1] for all model variants, and use the associated summarization checkpoints[2] fine-tuned on CNN/DailyMail (Hermann et al., 2015) for all models. For T5, we use the small (60M parameters), base (220M parameters), and large (770M parameters) variants, and the large variant for both BART (400M parameters) and PEGASUS (569M parameters). We fine-tuned all models on the WD tasks for 100 epochs for all experiments with a learning rate of 5e-5 with linear decay and a batch size of 16. We set the maximum source length to 1024 and the maximum target length to 256. For the BART model, we set the label smoothing factor to 0.1. We fine-tuned AST-T5, and CDS-T5 for 14 and 21 epochs, respectively, matching the original work of Chen et al. (2021), and used similar hyper-parameters used for the WD task. We ran all experiments on 4 NVIDIA A100 GPUs with 80G memory, and the training time of the longest experiment was under six hours.

5.2 Datasets

ABCD (Chen et al., 2021) contains over 10k human-to-human dialogues split over 8k training samples and 1k for each of the eval and test sets. The dialogues are divided across 30 domains, 231 associated slots, and 55 unique user intents from online shopping scenarios. Each intent requires a distinct flow of actions constrained by agent guidelines. To adapt ABCD to WD, we set actions to represent the workflow actions. Moreover, we do not use the provided sub-flows. Instead, we extract the actions directly from the annotated dialogues. Furthermore, we use natural language action names (i.e., using "pull up customer account" instead of "pull-up-account"). Table 1 shows a subset of the natural language action names and we report the full list of action names we used and the results of an ablation study showing the performance improvement gained by using natural language action names in Appendix A.5.

Table 1: Subset of ABCD workflow actions

Action Name	Natural Language Action Name
pull-up-account	pull up customer account
enter-details	enter details
verify-identity	verify customer identity

MultiWOZ (Budzianowski et al., 2018) contains over 10k dialogues with dataset splits similar to ABCD across eight domains: *Restaurent, Attraction, Hotel, Taxi, Train, Bus, Hospital, Police*. We use MultiWOZ 2.2 (Zang et al., 2020) as it contains annotated per turn user intents and apply additional annotations correction similar to Ye et al. (2021). In the MutliWOZ dataset, customer intents represent the workflow actions. We assume that the system will always perform a customer intent. Similar to ABCD, we use natural language action names. Table 2 shows a subset of the natural language action names. See Appendix A.4 for the full list.

[1] https://huggingface.co/transformers
[2] https://huggingface.co/models

Table 2: Subset of MultiWOZ workflow actions

Action Name	Natural Language Action Name
find_hotel	search for hotel
find_train	search for train
book_restaurant	book table at restaurant

5.3 Metrics

We evaluate the WD task using the Exact Match (EM) and a variation of the Cascading Evaluation (CE) (Suhr et al., 2019) metrics similar to Chen et al. (2021). The EM metric only gives credit if the predicted workflow (i.e., actions and slot values) matches the ground truth. However, it is not a good proxy of model performance on the WD task due to the sequential nature of workflows. For example, a model that predicts all workflow actions correctly but one will have an EM score of 0, similar to another model that predicted all actions wrong. In contrast, the CE metric gives partial credit for each successive correctly predicted action and its slot values. Nonetheless, we kept the EM metric for applications where predicting the exact workflow is critical. Furthermore, we report an action-only Exact Match (EM*) and Cascading Evaluation (CE*) for some experiments to help isolate the task complexity added by predicting the slot values. Finally, due to the text-to-text nature of our approach, we ignore any failure that occurs due to a stop word miss-match (e.g., we assume that *the lensfield hotel* is equivalent to *lensfield hotel*). Moreover, we use BERTScore (Zhang* et al., 2020) to evaluate the action in all zero-shot experiments. We assume that a predicted action is valid if it has an F1-score above 95% (e.g., *check customer identity* is equivalent to *verify customer identity*).

We evaluate the AST task similar to Chen et al. (2021) where B-Slot and Value are accuracy measures for predicting action names and values, respectively. Action is a joint accuracy for both B-Slot and Value metrics. Furthermore, We evaluate the CDS task similar to Chen et al. (2021) using the Cascading Evaluation (CE) metric that relies on the Intent, NextStep, B-Slot, Value, and Recall@1 metrics that are accuracy measures for the dialogue intent, next step, action, value, and next utterance predictions. We omit the Recall@5, and Recall@10 metrics results since our model only predicts a single next utterance instead of ranking the top 5 or 10 utterances. However, the cascading evaluation calculation result remains valid since it only uses the Recall@1 scores.

6 Experimental Results

To validate the efficacy of our approach against existing baselines, we first compare our results to the current state-of-the-art on the AST and CDS tasks and show that our method achieves state-of-the-art results on both tasks. Then, we describe the experiments we used to evaluate our approach on the WD task, report the results, and discuss their conclusions.

6.1 Action State Tracking

Following the same evaluation method used in Chen et al. (2021), we compared our AST-T5 model against ABCD-RoBERTa (Chen et al., 2021), the current state-of-the-art on the AST task. We report the results in Table 3. Moreover, we only report the results of AST-T5-Small (60M parameters) variant for a fair comparison since the ABCD-RoBERTa model is around 125M parameters while our AST-T5-Base is 220M parameters.

Table 3: Our AST-T5-Small results on the AST task using the ABCD test set.

Model	B-Slot	Value	Action
ABCD-RoBERTa	**93.6%**	67.2%	65.8%
AST-T5-Small	89.1%	**89.2%**	**87.9%**

Our AST-T5-Small variant achieves state-of-the-art results on the AST task while using 50% less trainable parameters. Furthermore, our text-to-text approach is easily adaptable to any new domain without any model change. In contrast, an encoder-only approach like the one used by ABCD-RoBERTa requires an update to the classification head to add a new action or slot value.

6.2 Cascading Dialogue Success (CDS)

Following the same evaluation method used in Chen et al. (2021), we compared our CDS-T5 models against the current state-of-the-art on the CDS task. In Table 4, we report the best results (ABCD Ensemble) for each metric from Chen et al. (2021).

Table 4: CDS-T5 results on the CDS task using the ABCD test set. ABCD Ensemble is the result of a model ensemble with the best scores from ABCD.

Metric	ABCD Ensemble	CDS-T5 Small	CDS-T5 Base
Intent	**90.5%**	85.7%	86.0%
Nextstep	87.8%	99.5%	**99.6%**
B-Slot	**88.5**%	85.9%	87.2%
Value	73.3%	75.1%	**77.3%**
Recall@1	22.1	40.7	**49.5**
CE	32.9%	38.3%	**41.0%**

Our CDS-T5-Small (60M parameters) outperforms the current state-of-the-art (with up to 345M parameters) while using 80% less trainable parameters. Furthermore, Our CDS-T5-Base achieves a new state-of-the-art on the CDS task while using 36% fewer trainable parameters, showing the advantage of our text-to-text approach. Specifically, our CDS-T5-Base outperformed the ABCD Ensemble on the next utterance prediction (recall@1) by 27.4 points. Furthermore, both our models scored exceptionally on predicting the next step. Finally, CDS-T5-Base outperforms the human performance (Chen et al., 2021) on value accuracy by 1.8 points.

6.3 Workflow Discovery (WD)

We present the results of all model variants on the WD task in the following setups: in-domain, cross-domain zero-shot, and cross-dataset zero-shot and few-shot. We report the results with and without our Actions conditioning technique (i.e., w/ Possible Actions). Furthermore, we report the results of our randomized actions conditioning technique (i.e., w/ Possible Actions$^+$) with $r_{min} = 10$ used during in-domain training only. Finally, we report results using multiple model architectures to show that our formulation is model-independent and to understand performance improvement as the model size scales and in different learning setups.

6.3.1 In-Domain Workflow Actions

Table 5 shows the results of our methods trained on all ABCD domains and tested on the ABCD test set. The significant difference between the Cascading Evaluation (CE) and Exact Match (EM) results across all configurations shows that it is much harder to predict the exact workflow and shows the CE metric's importance. Moreover, the EM* and CE* results, where we are only interested in the prediction of actions and not the slot values, show that predicting the slot values increases task complexity even more.

Our conditioning technique (i.e., w/ Possible Actions) improves the performance of all model variants except WD-T5-Small, where the EM score drops by 1.5 points. One explanation is that this model reaches its capacity since adding the possible actions lengthens the model input, which increases the complexity and reduces the performance, a known limitation of Transformer models (Tay et al., 2021; Ainslie et al., 2020). Nonetheless, our randomized actions conditioning technique (i.e., w/ Possible Actions$^+$) resolves this issue and improves, even more, the performance of all other model variants, especially larger ones. For example,

Table 5: WD in-domain test results on ABCD. w/ Possible Actions represents the results with the actions conditioning technique, and w/ Possible Actions$^+$ the results with the randomized actions conditioning technique.

Model	EM/CE	EM*/CE*
WD-T5-Small	44.1/67.9	55.4/78.6
w/ Possible Actions	42.6/66.9	53.6/77.6
w/ Possible Actions$^+$	44.8/68.6	56.7/79.1
WD-T5-Base	47.7/69.9	56.2/79.4
w/ Possible Actions	48.1/70.1	56.7/79.5
w/ Possible Actions$^+$	49.5/72.3	57.5/79.2
WD-BART-Large	42.0/60.3	61.9/68.8
w/ Possible Actions	43.2/61.0	62.3/69.0
w/ Possible Actions$^+$	44.9/64.3	**64.3**/70.07
WD-PEGASUS-Large	49.9/71.2	59.6/81.2
w/ Possible Actions	51.9/72.0	61.4/81.7
w/ Possible Actions$^+$	52.1/72.6	62.9/82.8
WD-T5-Large	50.6/73.1	59.9/81.4
w/ Possible Actions	54.6/74.8	62.3/83.0
w/ Possible Actions$^+$	**55.7/75.8**	63.3/**83.1**

WD-T5-Large EM score increased by 4.9 points. Moreover, the results show that the performance of our approach improves as the number of model parameters increases, which suggests that the results should improve with even larger models (i.e., more than 770M parameters).

Our analysis of these results shows that the main improvement of larger models on ABCD is their ability to predict slot values correctly since the dataset contains 30 unique actions with 231 associated slots. Moreover, larger models can better predict longer workflows (i.e., longer than four actions) which link to the Tranformers input sequence length issue discussed earlier. However, our WD-BART-Large shows an interesting behavior where it can predict the actions better than larger models, but performs poorly on slot values that represent either names, usernames, or emails (e.g., predicting *crystm561* instead of *crystalm561*, knowing that the former does not exist in the dataset). This observation, coupled with the fact that 70% of the test set contains at least one action that includes slot value with these types, explains the poor results.

6.3.2 ABCD Cross-Domain Zero-Shot

We performed a "leave-one-out" cross-domain zero-shot experiment similar to Lin et al. (2021b); Hu et al. (2022); Zhao et al. (2022b) on ABCD. In this setup, we train a model on samples from all the domains except one. Then, we evaluate the performance on the test set that includes the omitted domain. Table 6 shows the results of the experiments in which we excluded the "Shirt FAQ" and "Promo Code" domains. The domains were selected to better assess the transfer performance since the "Shirt FAQ" domain has similarities with three other domains present during training (i.e., "Boots FAQ", "Jeans FAQ", and "Jacket FAQ"). In contrast, the "Promo Code" domain is unique. A good approach should avoid confusing similar domains and be able to generalize to a different one.

Overall, our approach achieves good zero-shot transfer performance in both domains. Our randomized actions conditioning technique (i.e., w/ Possible Actions$^+$) improved the results in both domains. However, on average, the "Promo Code" EM score is 3.5 points lower than the in-domain results. This might be related to the uniqueness of the domain, which might lead to less positive transfer between the source and target domains. On the "Shirt FAQ" domain, only 15% of the failures are due to confusion with similar domains (mostly "Jacket FAQ"). In some cases the models generate plausible predictions (e.g., "does this *t-shirt* shrink" instead of "does this *shirt* shrink"). On the "Promo Code" domain, most of the failures were either invalid predictions where the models copy part of the input or predict an invalid slot value. Similarly, in some cases the models generate plausible predictions (e.g., "*generate new code*" instead of "*promo code*", see

Table 6: WD cross-domain zero-shot results on ABCD with "Shirt FAQ" and "Promo Code" as target domains. w/ Possible Actions$^+$ represents the results with the randomized actions conditioning technique.

Model	Shirt FAQ EM/CE	Promo Code EM/CE
WD-T5-Small	42.3/65.1	41.0/64.8
w/ Possible Actions$^+$	42.5/66.5	41.0/65.0
WD-T5-Base	46.0/67.6	45.1/68.8
w/ Possible Actions$^+$	47.8/69.7	45.7/69.0
WD-BART-Large	41.5/58.3	40.0/59.1
w/ Possible Actions$^+$	43.6/62.2	42.3/61.7
WD-PEGASUS-Large	47.4/68.8	46.2/68.9
w/ Possible Actions$^+$	49.6/69.2	47.4/69.3
WD-T5-Large	48.1/70.7	49.9/72.4
w/ Possible Actions$^+$	**51.8/72.3**	**50.2/72.4**

Appendix A.3 for more examples). We also performed an ablation study on the randomization conditioning technique and showed that it has a similar effect to the one seen in the in-domain setting. See Appendix A.2 for more details.

6.3.3 MultiWOZ Cross-Dataset Zero-Shot

In this setting, we evaluate if a model trained on one dataset can be used on another in a zero-shot setup similar to Zhao et al. (2022b). Therefore, we test our models trained on ABCD from Section 5 on MultiWOZ test set that includes all domains. We report the results of this experiment in Table 7. However, we excluded the results of our WD-T5-Small since it performed poorly in this setting.

Table 7: WD cross-dataset zero-shot test results on MultiWOZ. w/ Possible Actions$^+$ the results with the randomized actions conditioning technique.

Model	EM/CE
WD-T5-Base	3.6/10.0
w/ Possible Actions$^+$	23.0/38.3
WD-BART-Large	5.1/12.2
w/ Possible Actions$^+$	24.1/39.9
WD-PEGASUS-Large	9.8/15.2
w/ Possible Actions$^+$	**27.4/41.2**
WD-T5-Large	9.0/13.1
w/ Possible Actions$^+$	26.9/40.0

Without our action conditioning mechanism, all model variants did not perform well. However, adding the conditioning mechanism (i.e., w/ Possible Actions$^+$) improved the performance for all variants with an average EM score increase of 18.3 points, showing the importance of this technique in the zero-shot setting.

Most of the failures in this setting are due to invalid slot value predictions, especially for actions with more than three slot values. We believe that this is related to the fact the source domains (i.e., ABCD) have at most three slot values. Compared to the results of other settings, WD-PEGASUS-Large performs better than WD-T5-Large even if it has 200M fewer parameters. Our qualitative results analysis shows that WD-T5-Large uses actions from the source domains (i.e., ABCD) more often than WD-PEGASUS-Large. For example, it uses *search-timing* for cases where a customer asks about the train departure time. Moreover, our WD-BART-Large performs better in this setting since MultiWOZ does not contain usernames or email slot values. We noticed that some predicted slot values are more refined than the gold ones (e.g., *museum of science* instead of *museum*). While we can handle such cases during evaluation, it will make the testing process more complex. Finally, the models generate plausible actions that are not part of the list of possible actions (e.g., *check rating*, see Appendix A.3 for more details).

6.3.4 MultiWOZ Cross-Dataset Few-Shot

We consider the case where only a few samples are available for each workflow action. To this end, we conducted a cross-dataset few-shot experiment, where we trained a model on all ABCD domains, then fine-tuned it with randomly selected k samples per action from all MultiWOZ domains. We picked $k = 1$, $k = 5$, and $k = 10$ that yielded a training set of 11, 55, and 106 samples, respectively. Due to the nature of workflows containing multiple actions, some of these actions might have more than k samples.

Table 8: WD cross-dataset few-shot test results on MultiWOZ. w/ Possible Actions$^+$ the results with the randomized actions conditioning technique, and k is the number of samples per action.

Method	**EM/CE**
WD-T5-Base $k = 1$	5.9/54.8
w/ Possible Actions$^+$	38.2/58.7
WD-T5-Base $k = 5$	24.4/65.4
w/ Possible Actions$^+$	61.7/84.4
WD-T5-Base $k = 10$	43.2/73.0
w/ Possible Actions$^+$	**72.2**/**89.1**

We report the few-shot results of our WD-T5-Base in Table 8. Our approach shows a significant adaptation performance that keeps increasing as the number of training samples increases, reaching an EM score of 72.2. Moreover, our randomized actions conditioning technique shows an important performance gain in all the settings, where it improved the EM score by an average of 32.8 points. Similar to the cross-dataset zero-shot experiment, the failures are mainly due to invalid slot values prediction. Furthermore, the model generated 23 invalid actions as opposed to 5 when we use the conditioning technique (i.e., w/ Possible Actions$^+$). This behavior shows the utility of our conditioning mechanism and proves that it can restrict the model to the provided actions.

7 Conclusion

We have formulated a new problem called Workflow Discovery, in which we aim to extract workflows from dialogues consisting of sequences of actions and slot values representing the steps taken throughout a dialogue to achieve a specific goal. We have examined models capable of performing three different tasks: Workflow Discovery, Action State Tracking, and Cascading Dialogue Success. Our experiments show that our sequence-to-sequence approach can significantly outperform existing methods that use encoder-only language models for the AST and CDS tasks, achieving state-of-the-art results. Furthermore, our method is able to correctly predict both in-domain and out-of-distribution workflow actions, and our action conditioning approach affords much better performance for out-of-domain few-shot and zero-shot learning. We hope that our work sparks further NLP research applied to the field of process mining, but allowing for the use logs, meta-data and dialogues as input. Moreover, we believe that this method of characterizing complex interactions between users and agents could have significant impact on the way researchers and developers create automated agents, viewing workflow discovery as a special type of summarization that produces a form of natural language based program trace for task oriented dialogues.

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 268–284, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.19. URL `https://aclanthology.org/2020.emnlp-main.19`.

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

Derek Chen, Howard Chen, Yi Yang, Alex Lin, and Zhou Yu. Action-based conversations dataset: A corpus for building more in-depth task-oriented dialogue systems. *CoRR*, abs/2104.00783, 2021. URL https://arxiv.org/abs/2104.00783.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*, 2019.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340, 2015. URL http://arxiv.org/abs/1506.03340.

Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A. Smith, and Mari Ostendorf. In-context learning for few-shot dialogue state tracking. *ArXiv*, abs/2203.08568, 2022.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. Leveraging slot descriptions for zero-shot cross-domain dialogue StateTracking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5640–5648, Online, June 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.448. URL https://aclanthology.org/2021.naacl-main.448.

Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul A. Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. Leveraging slot descriptions for zero-shot cross-domain dialogue statetracking. In *NAACL*, 2021b.

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

Dominique Sommers, Vlado Menkovski, and Dirk Fahland. Process discovery using graph neural networks. In *2021 3rd International Conference on Process Mining (ICPM)*, pp. 40–47. IEEE, 2021.

Alane Suhr, Claudia Yan, Jack Schluger, Stanley Yu, Hadi Khader, Marwa Mouallem, Iris Zhang, and Yoav Artzi. Executing instructions in situated collaborative interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2119–2130, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1218. URL https://aclanthology.org/D19-1218.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qVyeW-grC2k.

Wil Van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE transactions on knowledge and data engineering*, 16(9):1128–1142, 2004.

Wil MP Van Der Aalst and Boudewijn F van Dongen. Discovering petri nets from event logs. In *Transactions on Petri nets and other models of concurrency vii*, pp. 372–422. Springer, 2013.

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. Multiwoz 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. *CoRR*, abs/2104.00773, 2021. URL `https://arxiv.org/abs/2104.00773`.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. *arXiv preprint arXiv:2007.12720*, 2020.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pp. 11328–11339. PMLR, 2020.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SkeHuCVFDr`.

Jeffrey Zhao, Raghav Gupta, Yuan Cao, Dian Yu, Mingqiu Wang, Harrison Lee, Abhinav Rastogi, Izhak Shafran, and Yonghui Wu. Description-driven task-oriented dialog modeling. *CoRR*, abs/2201.08904, 2022a. URL `https://arxiv.org/abs/2201.08904`.

Jeffrey Zhao, Raghav Gupta, Yuanbin Cao, Dian Yu, Mingqiu Wang, Harrison Lee, Abhinav Rastogi, Izhak Shafran, and Yonghui Wu. Description-driven task-oriented dialog modeling. *ArXiv*, abs/2201.08904, 2022b.

A Appendices

A.1 Ablation on Using Natural Language Action Names

We performed an ablation study to quantify the performance difference between using actions names in a format similar to *pull-up-account* compared to using natural language names (e.g., *pull up customer account)*. We followed an experimental setup similar to the one described in Section 6.3.1.

Table 9: Ablation study results on the effectiveness of using natural language action names (w/ NL Names). EM* and CE* are the action-only EM and CE metrics, respectively.

Models	**EM*/CE***
WD-T5-Small	53.2/74.1
WD-T5-Small w/ NL Names	**55.4/78.6**
WD-T5-Base	52.8/74.1
WD-T5-Base w/ NL Names	**56.2/79.4**
WD-BART-Large	59.2/66.3
WD-BART-Large w/ NL Names	**61.9/68.8**
WD-PEGASUS-Large	56.7/76.5
WD-PEGASUS-Large w/ NL Names	**59.6/81.2**
WD-T5-Large	56.6/85.9
WD-T5-Large w/ NL Names	**59.9/81.4**

We show the result of this ablation study in Table 9 in which we only report the EM* and CE* results since we are interested in the performance of predicting the actions and not the slot values for this study. We observe that using natural language names (i.e., w/ NL Names) yields better performance on both metrics on all model variants where it increased the EM* score by an average of 2.9. We think this is related to the fact that all the pre-trained models we used are fine-tuned for English summarization and thus should perform better on a formally-written text like *pull up customer account* as opposed to *pull-up-account*.

A.2 Ablation on the Conditioning Mechanism in the Cross-Domain Zero-Shot Setting

Similar to the in-domain experimental setup in Section 6.3.1, we performed an ablation study on the effect of the different conditioning methods (i.e., w/ Possible Actions, and w/ Possible Actions$^+$). Table 10 shows the result of this study.

Table 10: Cross-domain zero-shot results on the ABCD dataset with "Shirt FAQ" and "Promo Code" as target domains. EM and CE are the Exact Match and Cascading Evaluation metrics, respectively. w/ Possible Actions represents the results with possible actions added to the input, and w/ Possible Actions$^+$ represents the results with actions randomization technique.

Model	Shirt FAQ EM/CE	Promo Code EM/CE
WD-T5-Small	42.3/65.1	41.0/64.8
w/ Possible Actions	41.2/64.4	40.9/64.5
w/ Possible Actions$^+$	42.5/66.5	41.0/65.0
WD-T5-Base	46.0/67.6	45.1/68.8
w/ Possible Actions	46.9/68.7	45.3/68.8
w/ Possible Actions$^+$	**47.8/69.7**	**45.7/69.0**

Adding the possible actions (i.e., w/ Possible Actions) to the input has a similar negative effect on the smaller model (i.e., WD-T5-Small) as observed in the in-domain experiment results in Table 5, and similarly, it increased the performance of the larger variant (i.e., WD-T5-Base). However, our randomized actions conditioning technique (i.e., w/ Possible Actions$^+$) resolves this issue and improves performance on both model variants.

A.3 Plausible Predictions in the Zero-Shot Setting

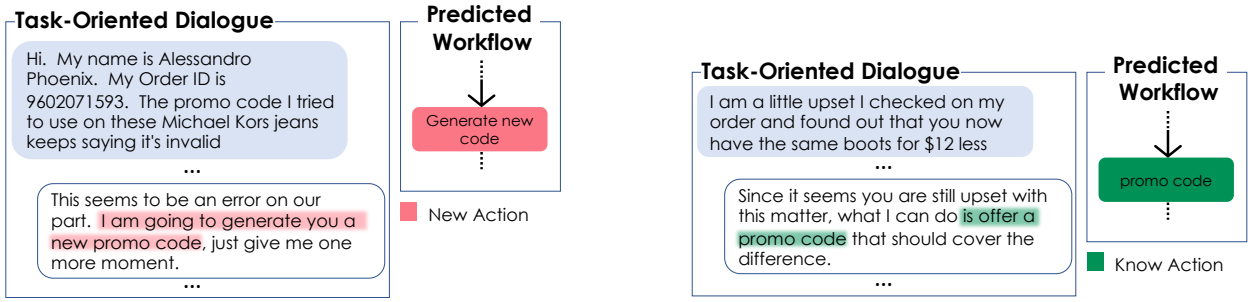


Figure 4: Plausible (a) and known (b) workflow actions generated during the ABCD cross-domain zero-shot experiment.

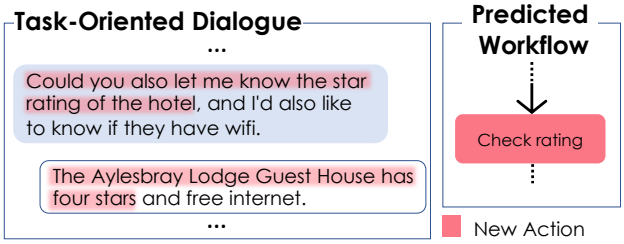


Figure 5: Plausible workflow action generated during the MultiWOZ cross dataset zero-shot experiment.

This section shows cases of plausible workflow actions predicted in the zero-shot setting that are not part of the possible actions. The generated actions are either "fine-grained" versions of existing actions or entirely new ones. For example, Figure 4a shows a case where the model predicted *generate new code* while the closet possible action is *promo code*. However, our analysis showed that there is a clear separation between cases in which agents offer a promo code as shown in Figure 4b and when they *generate* a new one if the old promo code is no longer working, as shown in Figure 4a.

Another example is shown in Figure 5 of an entirely new action that has no similarities with any other possible action. Here the customer is clearly asking for the rating of the hotel Hence the *check rating* action. Our dataset analysis showed that there are 10 test samples where customers requested to get the rating of hotels or restaurants.

A.4 MultiWOZ Workflow Actions

Table 11 shows the MultiWOZ workflow natural language action names we used in all our experiments.

Table 11: MultiWOZ workflow actions

Action Name	**Natural Language Action Name**
find_hotel	search for hotel
book_hotel	book hotel
find_train	search for train
book_train	book train ticket
find_attraction	search for attractions
find_restaurant	search for restaurants
book_restaurant	book table at restaurant
find_hospital	search for hospital
book_taxi	book taxi
find_taxi	search for taxi
find_bus	search for bus
find_police	search for police station

A.5 ABCD Workflow Actions

Table 12 shows the ABCD workflow natural language actions names we used in all our experiments.

Table 12: ABCD workflow actions

Action Name	**Natural Language Action Name**
pull-up-account	pull up customer account
enter-details	enter details
verify-identity	verify customer identity
make-password	create new password
search-timing	search timing
search-policy	check policy
validate-purchase	validate purchase
search-faq	search faq
membership	check membership level
search-boots	search for boots
try-again	ask customer to try again
ask-the-oracle	ask oracle
update-order	update order information
promo-code	promo code
update-account	update customer account
search-membership	get memberships information
make-purchase	make purchase
offer-refund	offer refund
notify-team	notify team
record-reason	record reason
search-jeans	search for jeans
shipping-status	get shipping status
search-shirt	search for shirt
instructions	check instructions
search-jacket	search for jacket
log-out-in	ask customer to log out log in
select-faq	select topic in faq
subscription-status	get subscription status
send-link	send link to customer
search-pricing	check pricing