Workflow Discovery from Dialogues in the Low Data Regime

Amine El Hattami [1 2 3] **Issam Laradji** [1] **Stefania Raimondo** [1] **David Vázquez** [1] **Pau Rodriguez** [1]
Christopher Pal [1 2 3]

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. In this work, 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 using multiple Seq2Seq models. We present experiments where we extract workflows from dialogues in the Action-Based Conversations Dataset (ABCD) and the MultiWOZ dataset. 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 in the out-of-distribution setting. Further, on ABCD a modified variant of our Seq2Seq method achieves state-of-the-art performance on related but different tasks of Action State Tracking (AST) and Cascading Dialogue Success (CDS) across many evaluation metrics. [1]

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.

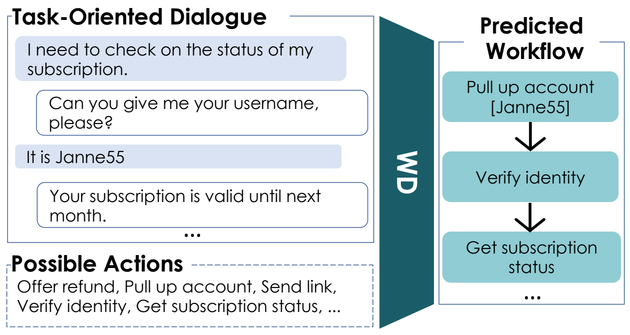

Task-Oriented Dialogue
I need to check on the status of my subscription.
Can you give me your username, please?
It is Janne55
Your subscription is valid until next month.
...
Possible Actions
Offer refund, Pull up account, Send link, Verify identity, Get subscription status, ...
WD
Predicted Workflow
Pull up account [Janne55]
Verify identity
Get subscription status
...

Figure 1. Our method for extracting workflows from dialogues. The input consists of the dialogue utterances and an optional list of possible actions. 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).

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 change over time. For instance, 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 by an agent to resolve a given task. These workflows consist of pre-defined terms for actions and slots when possible, but our formulation

[1]ServiceNow Research , Montréal, Canada [2]Polytechnique Montréal, Montréal, Canada [3]MILA, Montréal, Canada. Correspondence to: Amine El Hattami <amine.elhattami@servicenow.com>.

AI & HCI Workshop at the 40th International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

[1]Code available at https://github.com/ServiceNow/research-workflow-discovery

requires that new actions and slots can be invented on the fly when needed. We believe that WD differs from existing dialogue-related tasks (see Section 3.2 for more details) and 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. Our task is targeted toward 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 if workflows have organically emerged. WD is related to the well-established field of process mining which typically extracts workflow information from event logs instead of unstructured text dialogues. Our work shows that traditional process mining can be dramatically expanded using modern NLP and sequence modeling techniques, allowing systems to benefit from recent advances in large language models (LLMs), including enhanced zero-shot and few-shot learning performance, as we propose and explore here.

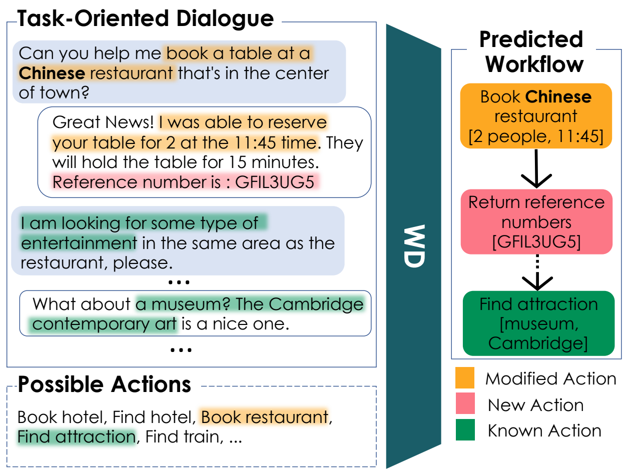

Task-Oriented Dialogue
Can you help me book a table at a Chinese restaurant that's in the center of town?
Great News! I was able to reserve your table for 2 at the 11:45 time. They will hold the table for 15 minutes. Reference number is : GFIL3UG5
I am looking for some type of entertainment in the same area as the restaurant, please.
What about a museum? The Cambridge contemporary art is a nice one.
Possible Actions
Book hotel, Find hotel, Book restaurant, Find attraction, Find train, ...
WD
Predicted Workflow
Book Chinese restaurant [2 people, 11:45]
Return reference numbers [GFIL3UG5]
Find attraction [museum, Cambridge]
Modified Action
New Action
Known Action

Figure 2. Our conditioning approach allows for better out-of-distribution performance. Entirely new workflow actions can be proposed (in pink), as well as those that are minor modifications to known actions (in yellow).

To address the challenges of WD introduced by dynamically changing actions, distributional shifts, and transferring to completely new domains with limited labeled data, we propose a text-to-text approach that can output the workflow consisting of the complete set of actions and slot values from a task-oriented dialogue. Further, to better adapt to unseen domains, we condition our method on the whole or partial set of possible actions as shown in Figure 3(a), 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 in a few-shot setting.

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.
- 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. 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) and MutliWOZ (Budzianowski et al., 2018).
- 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 and few-shot performance.
- 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 In task-oriented dialogues, the system must grasp the users' intentions from their utterances to select an appropriate system action. It should be able to understand and respond to a wide range of user inputs, handle complex tasks, and maintain coherence in the dialogue (Stolcke et al., 2000; Henderson et al., 2014). The two fundamental tasks are intent detection and named entity identification, which have comparable components in task-oriented dialogues. Intent detection may be viewed as a classification task, with user utterances allocated to one or more intent categories. The goal of named entity recognition is to extract entities from a given utterance, such as hotel names, room characteristics, and attribute values (Zang et al., 2020). Although most systems perform intent detection and named entity identification for generic conversation systems (Rafailidis & Manolopoulos, 2018; Yan et al., 2017), the main goal of our method is to output a set of actions that resolve the task specified in the dialogue that abides by the system's guidelines similar to the Action State Tracking (AST) task (Chen et al., 2021).

Intent and Slot Induction. The goal of intent/slot induction is to determine user intents and slots from user utterances. Recently, there have been several efforts on intent/slot detection in conversation systems based on methods like capsule networks and relation networks (Min et al., 2020; Qin et al., 2020; Zhang et al., 2018; Qin et al., 2019; Niu et al., 2019). However, they all assume that the intents are within a closed world in that the intents in the test set also appear in the training set. In contrast, while our work is about identifying the actions that address the task in a dialogue, we consider the possibility that the actions in the test set are novel. Prior work has also explored unsupervised methods. For example, (Brychcín & Král, 2016) proposed an unsupervised method for identifying intents without utilizing prior knowledge of known intents. (Perkins & Yang, 2019) used clustering with multi-view representation learning for intent induction, whereas (Zeng et al., 2021) proposed a method based on role-labeling, concept-mining, and pattern-mining for discovering intents on open-domain dialogues. Closer to our work is (Yu et al., 2022), which leveraged large language models for discovering slot schemas for task-oriented dialog in an unsupervised manner. They use unsupervised parsing to extract candidate slots, followed by coarse-to-fine clustering to induce slot types. In contrast, we use LLMs and a prompting technique to detect and even label out-of-distribution actions.

AST and CDS for Task-Oriented Dialogues. Action State Tracking (AST) is a task proposed by (Chen et al., 2021) which tries to predict relevant intents from customer utterances while taking Agent Guidelines into consideration, which goes steps beyond traditional dialog state tracking (DST) (Lee et al., 2021). For instance, a customer's utterance might suggest that the action is to update an order. However, following the agent's guidelines, the agent might need to perform other actions before addressing the customer's intent (e.g., validating the customer's identity). While AST shares some similarities with traditional DST tasks, its main advantage is parsing customer intents and agent guidelines to predict agent actions. However, AST relies on annotated action turns, making it difficult to use on existing dialogue datasets without substantial annotation effort. Further, AST relies on agent guidelines which require prior knowledge of all possible actions. (Chen et al., 2021) also proposed the Cascading Dialogue Success (CDS) task to access the model's ability to predict actions in context. CDS differs from AST since it does not assume that an action should occur in the current turn but adds the possibility to respond with an utterance (e.g., requesting information from the customer) or terminate the conversation when the agent completes the task.

3. Workflow Discovery

3.1. Task Definition

Workflow Discovery (WD) is the task of extracting a workflow from a task oriented dialogue, where the extracted workflow is a *sequence* of *actions* with their respective slot values in the same order in which they appeared during the conversation. Formally, we define WD as follows:

Definition 3.1. 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 is the j^{th} slot value and n_i is the number of available slot values for action a_i, and k is the number of workflow actions. Further, the model should also be able to formulate new compound keywords to characterize new actions as well as extract their slot values for actions that are not a part of the known action domain.

3.2. WD Compared to Existing Tasks

WD differs from DST since we are interested in extracting the sequence of actions followed to resolve an issue. We are particularly interested in the situation where the actions *are not known a priori* and must be invented by the WD model. This sharply contrasts with DST, which generally requires dialogue states to be known and pre-defined. DST also typically focuses on tracking one party's state in the conversation, e.g., the user. In contrast, WD extracts actions executed by the agent and slot values collected from user utterances.

WD is also different from standard (closed world) intent detection and slot filling (Liu & Lane, 2016), dialogue state tracking (DST) (Williams et al., 2014), and open world intent and slot induction (Perkins & Yang, 2019) in part because WD focuses on extracting actions taken by an agent as opposed to intents of the user in addition to slots obtained from both the user and the agent utterances.

WD differs from AST since it 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). WD could be seen as a generalization of the recently proposed AST task. In summary, WD differs from AST in that: 1) WD is performed offline, summarizing dialogues in terms of workflows; 2) unlike AST, WD does not require actions to be annotated at the turn level; 3) unlike AST, WD doesn't require known predefined actions and slots. One of the uses of WD is to create new names for new actions on the fly along with new slot types and extracted values; and 4) WD can be applied to entirely new, organic tasks where possible actions are unknown. Using models trained for AST can help agents select the next best action following the 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 to formalize the discovered processes for subsequent automation.

Our WD problem formulation is related to dialogue act modeling. Different steps along our workflows may be composed of dialogue acts similar to (Stolcke et al., 2000), such as: posing yes or no questions, acknowledging responses, thanking, etc. Since WD is performed offline and focuses on workflow extraction, more fine-grained dialogue acts can help guide a WD model to extract more abstract workflow steps.

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. 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.

4. Baselines and Methodology

This section describes our methodology and baselines for WD. Further, we included our text-to-text variants AST and CDS to test our casting scheme against existing benchmarks that we report in A.2.1 and A.2.2, respectively. We show examples for each task in Figure 3.

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 full or partial list of possible actions, formatted as

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

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 that separate the dialogue utterances from the list of actions. Further, we omit the "$Actions$:" delimiter when possible actions are not provided. Adding the possible actions to the input can be seen as a way to condition the model to use the provided actions rather than invent new ones. During training, the possible actions added after "$Actions$:" in the input is a *shuffled* set comprised of the current sample target actions and a randomly chosen number $r_{min} <= r <= z$ of actions, where r_{min} is a hyper-parameter. This technique makes 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 show the positive effect of this technique in an ablation study in Appendix A.4.1. For all other settings (i.e., during validation, testing, or the zero-shot setting), the list of possible actions contains *all* known actions without modification. 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 is shown in Figure 3(a).

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 (e.g., using an upstream text-to-speech model).

5. Experimental Setup

5.1. Implementation Details

In our experimentation, we used the T5 (Raffel et al., 2020b), 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 (Wolf et al., 2020) for all model variants and use the associated summarization checkpoints fine-tuned on CNN/DailyMail (Hermann et al., 2015) for all models, and we show its utility in Appendix A.6. 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 for the WD task. In all experiments, we use $r_{min} = 10$ as described in Section 4.1. Finally, 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 online shopping domains, 231 associated slots, and 55 unique user intents from online shopping scenarios. To adapt ABCD to WD, we choose agent actions with their slot values to represent the workflow actions. Further, we created natural language names for each action as shown in Table1 [2]. See Appendix A.9.1 for how we adapted the ABCD dataset for the WD task. Finally, We report the generated WD dataset statistics for both datasets in Appendix A.9.2.

Table 1. Example of natural language (NL) action names for ABCD and MultiWOZ dataset. See Table 13 and 14 for the full list for ABCD and MultiWOZ, respectively.

DATASET	ACTION NAME	NL ACTION NAME
ABCD	pull-up-account	pull up customer account
MutliWOZ	book_restaurant	book table at restaurant

MultiWOZ (Budzianowski et al., 2018) contains over 10k dialogues with dataset splits similar to ABCD across eight domains related to booking and getting information. We use MultiWOZ 2.2 (Zang et al., 2020) as it contains annotated per turn user intents and applies additional annotations correction similar to (Ye et al., 2022). In the MultiWOZ dataset, we chose customer intents to represent the workflow actions. We assume that the system will always perform a customer intent. Similar to ABCD, we use natural language action names as shown in Table 1 [2]. See Appendix A.9.1 for more details.

[2] Dataset available in our public code repository.

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).

Furthermore, we report an action-only Exact Match (Action-only EM) and Cascading Evaluation (Action-only CE) for some experiments to help isolate the task complexity added by predicting the slot values. Due to the text-to-text nature of our approach, we use stemming, and 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*). Finally, we evaluate the AST and CDS tasks similar to (Chen et al., 2021). However, We only report Recall@1 scores since our implementation only predicts a single next utterance. Nonetheless, the cascading evaluation calculation result remains valid since it only uses the Recall@1 scores.

6. Experimental Results

6.1. Workflow Discovery (WD)

6.1.1. In-Domain Workflow Actions

Table 2 shows the results of our methods trained on all ABCD domains and tested on the ABCD test set. We report each model variant's Cascading Evaluation (CE) and Exact Match (EM). Further, to understand the added complexity of predicting the slot values, we report the Action-only CE and Action-only EM results, where we are only interested in the models' ability to predict the actions correctly regardless of the predicted slot values.

In all configurations, the EM and CE scores show an expected improvement as we scale the model size except for the BART-Large variant that showed an interesting behavior when predicting slot values representing identifiers such as an account ID (See Appendix A.7.1 for more details). Our qualitative analysis showed that most performance improvements as we increase in the model size can be attributed to an enhanced slot value extraction capability. The challenge in predicting these slot values arises from the fact that ABCD contains 231 unique slots, and each action may have 0, or up to 3 slots. Moreover, some actions utilize optional slots based on the customer's response. For instance, when performing the "pull up customer account" action, an agent can use the account ID or the customer's full name. Further, Table 2 shows that even with a 770M parameter model, the best EM score is only slightly above 50%, indicating the difficulty of the WD task. Finally, the significant disparity

Table 2. WD in-domain test results on ABCD with both possible actions configurations. Action-only EM and CE are the EM and CE scores for Action-only prediction, excluding slot values.

Models	EM/CE		Action-only EM/CE	
	without possible actions	with possible actions	without possible actions	with possible actions
T5-Small (60M)	44.1/67.9	44.8/68.6	55.4/78.6	56.7/79.1
T5-Base (220M)	47.7/69.9	49.5/72.3	56.2/79.4	57.5/79.2
BART-Large (406M)	42.0/60.3	44.9/64.3	61.9/68.8	**64.3**/70.1
PEGASUS-Large (568M)	49.9/71.2	52.1/72.6	59.6/81.2	62.9/82.8
T5-Large (770M)	50.6/73.1	**55.7/75.8**	59.9/81.4	63.3/**83.1**

between the EM and CE scores across all configurations reveal that predicting the actions and respective slot values in the precise order are much more challenging, highlighting the usefulness of CE as an effective metric for evaluating WD's performance.

Adding possible actions enhances the performance of all model variants, especially larger ones like T5-Large, where the EM score increased by 5.1. This suggests that even larger models could yield improved results. However, adding possible actions did not have the same effect when using the smallest model (i.e., T5-Small), where the EM score improved by only 0.7, possibly due to difficulty in utilizing larger inputs, a known limitation of Transformer models (Tay et al., 2021; Ainslie et al., 2020). Similar to increasing the model size, most of the gained improvements are due to better slot value prediction, which might seem counterintuitive. We believe that in this configuration, predicting the actions becomes easier, enabling the models to allocate more capacity to extract the slot values. This is further supported by the improved Action-only EM and CE scores when adding possible actions. For a more detailed qualitative result analysis, see Appendix A.7.1.

To better understand the performance of our proposed baselines, We conducted a human evaluation on 100 randomly selected test samples, representing 10% of the ABCD test set [3]. We hired two annotators to label the samples and provided them with the complete list of actions with descriptions taken from the ABCD agent's guidelines. Moreover, we provided three fully annotated dialogues for each action from the ABCD training set. Finally, a third annotator was tasked with resolving any samples the initial annotators disagreed on. Table 3 shows the EM and CE scores on the annotated subset.

The results in Table 3 show that even our smallest model outperforms human performance by a large margin. Our analysis showed that our method outperforms human annotators in cases where the agent performs multiple actions after the same turn. Further, our approach outperformed human performance in dialogues with more than 13 turns. The low human performance can be attributed to the complexity of the ABCD's domain, also shown by the low Cohen kappa coefficient of 0.63. We believe that further annotator training might improve human performance. However, we don't expect the human performance to reach the model's performance. Nonetheless, we leave this for future work.

[3]The annotated human subset is available in our public code repository.

Table 3. Human evaluation results on ABCD test subset.

Methods	EM/CE
Human Performance	11.0/38.12
T5-Small without possible actions	**42.3/66.0**

6.1.2. 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., 2022b; 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. In our experiment, we excluded the "Shirt FAQ" and "Promo Code" domains by removing all training samples that contain actions unique to each domain. The "Shirt FAQ" domain comprises 34 dialogues representing 3.4% of the ABCD test set. It overlaps other domains, such as "Boots FAQ" and "Jacket FAQ" on two actions (i.e., "*search faq*" and "*select topic in faq*"). However, while other domains use the "*select topic in faq*" action, the "Shirt FAQ" domain has eight unique slot values for this action since they represent existing FAQ questions that agents have to choose from (described in ABCD's agent guidelines). The "Promo Code" domain consists of 53 dialogues representing 5.3% of the ABCD test set. It overlaps other domains on three actions (i.e., "*pull up customer account*", "*ask oracle*", and "*check membership level*"). However, the order in which these actions appear is unique to this domain. Table 4 shows the results of both experiments.

The results show that our approach achieves good zero-

Table 4. WD leave-one-out cross-domain zero-shot results on ABCD with both possible actions configurations using "Shirt FAQ" and "Promo Code" as target domains.

Models	Promo Code EM/CE		Shirt FAQ EM/CE	
	without possible actions	with possible actions	without possible actions	with possible actions
T5-Small (60M)	42.3/65.1	42.5/66.5	41.0/64.8	41.0/65.0
T5-Base (220M)	46.0/67.6	47.8/69.7	45.1/68.8	45.7/69.0
BART-Large (406M)	41.5/58.3	43.6/62.2	40.0/59.1	42.3/61.7
PEGASUS-Large (568M)	47.4/68.8	49.6/69.2	46.2/68.9	47.4/69.3
T5-Large (770M)	48.1/70.7	**51.8/72.3**	49.9/72.4	**50.2/72.4**

shot transfer performance in both domains. Overall, the behavior of all model variants on both domains follows the same trends as the in-domain results in Table 2. However, BART-Large has the least performance drop compared to the in-domain results since the target domains do not contain the slots in which this model performs poorly, as described in Section A.7.1. On average, the performance on the "Shirt FAQ" is higher than the "Promo Code" domain, where the latter has an EM score 3.5 points lower than the in-domain results, showing that the "Promo Code" domain presents a more difficult cross-domain zero-shot setup. Our qualitative analysis showed that smaller models have difficulty generating the omitted actions, and valid predictions increase as we increase the model size. However, we noticed that larger model variants tend to confuse "offer refund" and "offer promo code" actions since they are used similarly in the dialogues. See A.7.2 for more details.

6.1.3. MultiWOZ Cross-Dataset Zero-Shot

In this experiment, we evaluate the transfer performance on totally different domains in a zero-shot setup similar to (Zhao et al., 2022b). Therefore, we test the models trained on ABCD (Section 6.1.2) on all MultiWOZ test set domains. In this setting, there is no inter-dataset overlap. All the eight MultiWOZ domains are different from the ones in ABCD. We report the results of this experiment in Table 5.

Table 5. WD cross-dataset zero-shot test results on MultiWOZ with both possible actions configurations.

Models	EM/CE	
	without possible actions	with possible actions
T5-Small (60M)	0.0/0.0	5.3/8.4
T5-Base (220M)	3.6/10.0	23.0/38.3
BART-Large (406M)	5.1/12.2	24.1/39.9
PEGASUS-Large (568M)	9.8/15.2	27.4/41.2
T5-Large (770M)	9.0/13.1	**26.9/40.0**

Table 5 shows that all model variants performed poorly when the list of actions was not included in the input. For example, T5-Small could not predict any valid workflow, and our largest model achieved only a 9.0 EM score. In this setting, all valid predictions were for cases with a single slot value, and the actions were deemed correct using our evaluation method described in Section 5.3 (e.g., "book restaurant" and "reserve a restaurant" are both valid). Further, most failures are cases where the models copy parts of the input or use ABCD actions. When we add the possible actions, the performance improved for all variants with an average EM score increase of 18.3 points, showing the utility of this technique in the zero-shot setting. In this setting, 65% of the valid predictions match the one provided in the input, and the remaining are generated action names deemed correct. Moreover, most of the failures are due to invalid slot value predictions. One interesting observation is that the performance of T5-Large is lower than PEGASUS-Large even though it has 200M more parameters, and our analysis showed that T5-Large uses ABCD's action names more often than PEGASUS-Large. We report additional results analysis in Appendix A.7.3.

6.1.4. 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 6. WD cross-dataset few-shot test results on MultiWOZ with both possible actions configurations.

k	# Samples	EM/CE	
		without possible actions	with possible actions
1	11	5.9/54.8	8.2/58.7
5	55	24.4/65.4	61.7/84.4
10	106	43.2/73.0	**72.2/89.1**

We report the few-shot results of our WD-T5-Base in Table 6. Our approach shows a significant adaptation performance that keeps increasing as training samples increase, reaching an EM score of 72.2. Moreover, adding the possible actions to the input shows a considerable performance gain in all the settings, improving the EM score by an average of 32.8 points. Our analysis showed that the results follow the same behavior on the cross-dataset zero-shot experiment with $k = 1$. Then, all other values of k follow similar behaviors as in the ABCD in-domain experiment. Similar to the cross-dataset zero-shot experiment, the failures are mainly due to invalid slot values prediction; however, when $k = 10$, the slot values issues reduce considerably. Furthermore, the model generated 23 invalid actions instead of 5 when the list of actions was provided in the input. This behavior shows the utility of this conditioning mechanism and proves that it can restrict the model to the provided actions.

6.1.5. Limitations

As shown in the results above, adding the possible actions in all settings improves the performance. However, it presents a limitation of the proposed baselines since this list may be unknown or frequently updated. Our qualitative analysis also showed that predicting the slots is challenging. One possible solution is to include slot descriptions similar to Zhao et al. (2022b). However, similar to the actions, this information might be unknown. We describe additional limitations in Appendix A.8, and we leave solving these issues for future work.

6.2. AST and CDS

While our work mainly focuses on the WD task, we include our results on AST and CDS aimed to validate our text-to-text casting scheme against existing benchmarks.

AST We compared our AST-T5 model against ABCD-RoBERTa (Chen et al., 2021), the current state-of-the-art on the AST task. We describe the casting scheme in Appendix A.2.1. Table 7 shows the results of our experiment.

Table 7. Our AST-T5-Small results on the AST task using the ABCD test set.

Models	B-Slot	Value	Action
ABCD-RoBERTa (124M)	**93.6%**	67.2%	65.8%
AST-T5-Small (60M)	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 as opposed to encoder-only architectures like the one used in (Chen et al., 2021).

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. We describe the casting scheme in Appendix A.2.2. In Table 8, we report the best results (ABCD Ensemble) for each metric from (Chen et al., 2021).

Table 8. CDS-T5 results on the CDS task using the ABCD test set. ABCD Ensemble is the ensemble with the best scores from ABCD (Chen et al., 2021).

Metric	ABCD Ensemble (345M)	CDS-T5 Small (60M)	CDS-T5 Base (220M)
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 (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.

7. Conclusion

We have formulated a new problem called Workflow Discovery (WD), 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 evaluated our proposed various baselines in the in-domain and out-of-distribution settings and showed that WD could be cast as a text-to-text task. Further, we described some limitations of the proposed baselines to motivate future work. Our experiments also 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. We hope that our work sparks further NLP research applied to the field of process mining but allows for using logs, meta-data, and dialogues as input. Moreover, we believe that this method of characterizing complex interactions between users and agents could significantly impact how 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

Ainslie, J., Ontanon, S., Alberti, C., Cvicek, V., Fisher, Z., Pham, P., Ravula, A., Sanghai, S., Wang, Q., and Yang, L. ETC: Encoding long and structured inputs in transformers. *Empirical Methods in Natural Language Processing (EMNLP)*, November 2020.

Brychcín, T. and Král, P. Unsupervised dialogue act induction using gaussian mixtures. *European Chapter of the Association for Computational Linguistics (EACL)*, 2016.

Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., and Gašić, M. Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Chen, D., Chen, H., Yang, Y., Lin, A., and Yu, Z. Action-based conversations dataset: A corpus for building more in-depth task-oriented dialogue systems. *North American Chapter of the ACL (NAACL)*, 2021.

Henderson, M., Thomson, B., and Williams, J. D. The second dialog state tracking challenge. *Special interest group on discourse and dialogue (SIGDIAL)*, pp. 263–272, 2014.

Hermann, K. M., Kociský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340, 2015.

Hu, Y., Lee, C.-H., Xie, T., Yu, T., Smith, N. A., and Ostendorf, M. In-context learning for few-shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 2627–2643, Abu Dhabi, United Arab Emirates, December 2022a. Association for Computational Linguistics. URL `https://aclanthology.org/2022.findings-emnlp.193`.

Hu, Y., Lee, C.-H., Xie, T., Yu, T., Smith, N. A., and Ostendorf, M. In-context learning for few-shot dialogue state tracking. *arXiv*, abs/2203.08568, 2022b.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and tau Yih, W. Dense passage retrieval for open-domain question answering. *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Lee, C.-H., Cheng, H., and Ostendorf, M. Dialogue state tracking with a language model using schema-driven prompting. *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Association for Computational Linguistics (ACL)*, 2019.

Lin, Z., Liu, B., Moon, S., Crook, P., Zhou, Z., Wang, Z., Yu, Z., Madotto, A., Cho, E., and Subba, R. Leveraging slot descriptions for zero-shot cross-domain dialogue State-Tracking. *North American Chapter of the ACL (NAACL)*, 2021a.

Lin, Z., Liu, B., Moon, S., Crook, P. A., Zhou, Z., Wang, Z., Yu, Z., Madotto, A., Cho, E., and Subba, R. Leveraging slot descriptions for zero-shot cross-domain dialogue statetracking. *North American Chapter of the ACL (NAACL)*, 2021b.

Liu, B. and Lane, I. Attention-based recurrent neural network models for joint intent detection and slot filling. *International Speech Communication Association (INTERSPEECH)*, 2016.

Min, Q., Qin, L., Teng, Z., Liu, X., and Zhang, Y. Dialogue state induction using neural latent variable models. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

Niu, P., Chen, Z., Song, M., et al. A novel bi-directional interrelated model for joint intent detection and slot filling. *Association for Computational Linguistics (ACL)*, 2019.

Perkins, H. and Yang, Y. Dialog intent induction with deep multi-view clustering. *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Qin, L., Che, W., Li, Y., Wen, H., and Liu, T. A stack-propagation framework with token-level intent detection for spoken language understanding. *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Qin, L., Che, W., Li, Y., Ni, M., and Liu, T. Dcr-net: A deep co-interactive relation network for joint dialog act recognition and sentiment classification. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

Rafailidis, D. and Manolopoulos, Y. The technological gap between virtual assistants and recommendation systems. *arXiv*, 2018.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*, 2020a.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*, 21:1–67, 2020b.

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Ess-Dykema, C. V., and Meteer, M. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Association for Computational Linguistics (ACL)*, 2000.

Suhr, A., Yan, C., Schluger, J., Yu, S., Khader, H., Mouallem, M., Zhang, I., and Artzi, Y. Executing instructions in situated collaborative interactions. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena : A benchmark for efficient transformers. *International Conference on Learning Representations (ICLR)*, 2021.

Williams, J. D., Henderson, M., Raux, A., Thomson, B., Black, A., and Ramachandran, D. The dialog state tracking challenge series. *AI Magazine*, 35(4):121–124, 2014.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Yan, Z., Duan, N., Chen, P., Zhou, M., Zhou, J., and Li, Z. Building task-oriented dialogue systems for online shopping. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2017.

Ye, F., Manotumruksa, J., and Yilmaz, E. Multiwoz 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. *Special interest group on discourse and dialogue (SIGDIAL)*, 2022.

Yu, D., Wang, M., Cao, Y., Shafran, I., Shafey, L. E., and Soltau, H. Unsupervised slot schema induction for task-oriented dialog. *North American Chapter of the ACL (NAACL)*, 2022.

Zang, X., Rastogi, A., Sunkara, S., Gupta, R., Zhang, J., and Chen, J. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. *NLP4ConvAI*, 2020.

Zeng, Z., Ma, D., Yang, H., Gou, Z., and Shen, J. Automatic intent-slot induction for dialogue systems. *Proceedings of the Web Conference (WWW)*, 2021.

Zhang, C., Li, Y., Du, N., Fan, W., and Yu, P. S. Joint slot filling and intent detection via capsule neural networks. *Association for Computational Linguistics (ACL)*, 2018.

Zhang, J., Zhao, Y., Saleh, M., and Liu, P. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *International Conference on Machine Learning (ICML)*, 2020.

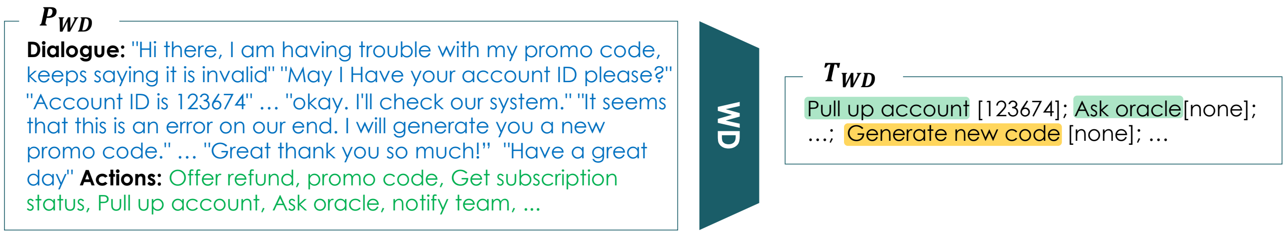
Zhang*, T., Kishore*, V., Wu*, F., Weinberger, K. Q., and Artzi, Y. Bertscore: Evaluating text generation with bert. *International Conference on Learning Representations (ICLR)*, 2020.

Zhao, J., Gupta, R., Cao, Y., Yu, D., Wang, M., Lee, H., Rastogi, A., Shafran, I., and Wu, Y. Description-driven task-oriented dialog modeling. *arXiv*, 2022a.

Zhao, J., Gupta, R., Cao, Y., Yu, D., Wang, M., Lee, H., Rastogi, A., Shafran, I., and Wu, Y. Description-driven task-oriented dialog modeling. *arXiv*, abs/2201.08904, 2022b.

A. Appendices

A.1. Prompt and Target Examples



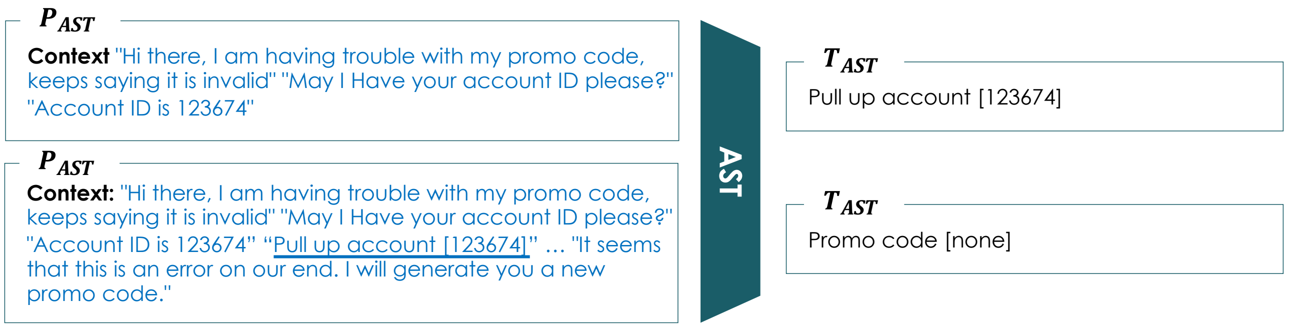
P_{WD}

Dialogue: "Hi there, I am having trouble with my promo code, keeps saying it is invalid" "May I Have your account ID please?" "Account ID is 123674" … "okay. I'll check our system." "It seems that this is an error on our end. I will generate you a new promo code." … "Great thank you so much!" "Have a great day" **Actions:** Offer refund, promo code, Get subscription status, Pull up account, Ask oracle, notify team, …

T_{WD}

Pull up account [123674]; Ask oracle[none]; …; Generate new code [none]; …

(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).



P_{AST}

Context "Hi there, I am having trouble with my promo code, keeps saying it is invalid" "May I Have your account ID please?" "Account ID is 123674"

P_{AST}

Context: "Hi there, I am having trouble with my promo code, keeps saying it is invalid" "May I Have your account ID please?" "Account ID is 123674" "Pull up account [123674]" … "It seems that this is an error on our end. I will generate you a new promo code."

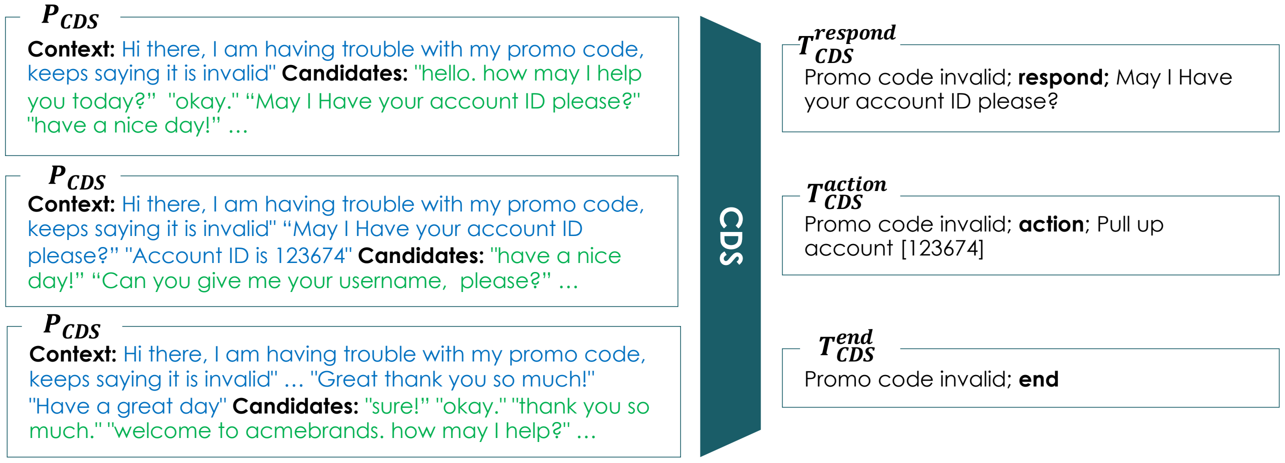
T_{AST}

Pull up account [123674]

T_{AST}

Promo code [none]

(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).



P_{CDS}

Context: Hi there, I am having trouble with my promo code, keeps saying it is invalid" **Candidates:** "hello. how may I help you today?" "okay." "May I Have your account ID please?" "have a nice day!" …

P_{CDS}

Context: Hi there, I am having trouble with my promo code, keeps saying it is invalid" "May I Have your account ID please?" "Account ID is 123674" **Candidates:** "have a nice day!" "Can you give me your username, please?" …

P_{CDS}

Context: Hi there, I am having trouble with my promo code, keeps saying it is invalid" … "Great thank you so much!" "Have a great day" **Candidates:** "sure!" "okay." "thank you so much." "welcome to acmebrands. how may I help?" …

$T_{CDS}^{respond}$

Promo code invalid; **respond;** May I Have your account ID please?

T_{CDS}^{action}

Promo code invalid; **action**; Pull up account [123674]

T_{CDS}^{end}

Promo code invalid; **end**

(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.

A.2. AST and CDS Text-to-Text Casting

In this section, we describe our methodology for both the AST and CDS tasks that we used to test our text-to-text task casting scheme against existing benchmarks.

A.2.1. 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 3(b).

A.2.2. 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 3(c).

A.3. Plausible Predictions in the Zero-Shot Setting

This section shows examples of plausible workflow actions that are not part of the possible actions predicted in the zero-shot setting. The generated actions are either ”fine-grained” versions of existing actions or entirely new ones. For example,

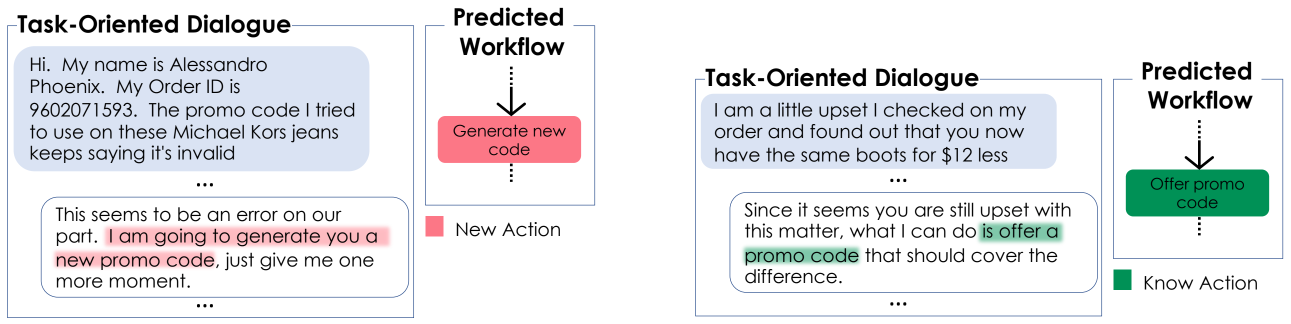

Task-Oriented Dialogue
Hi. My name is Alessandro Phoenix. My Order ID is 9602071593. The promo code I tried to use on these Michael Kors jeans keeps saying it's invalid
...
This seems to be an error on our part. I am going to generate you a new promo code, just give me one more moment.
...
Predicted Workflow
Generate new code
🟥 New Action
Task-Oriented Dialogue
I am a little upset I checked on my order and found out that you now have the same boots for $12 less
...
Since it seems you are still upset with this matter, what I can do is offer a promo code that should cover the difference.
...
Predicted Workflow
Offer promo code
🟩 Know Action

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

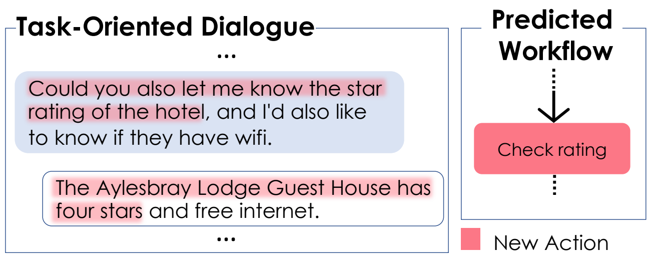

Task-Oriented Dialogue
...
Could you also let me know the star rating of the hotel, and I'd also like to know if they have wifi.
The Aylesbray Lodge Guest House has four stars and free internet.
...
Predicted Workflow
Check rating
🟥 New Action

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

Figure 4(a) shows a case where the model predicted *generate new code* while the closest possible action is *offer 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 4(b) and when they *generate* a new one if the old promo code is no longer working, as shown in Figure 4(a).

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 10 test samples where customers requested to get the rating of hotels or restaurants.

A.4. Ablation on the Possible Actions Randomization

A.4.1. In-Domain Setting

We run an ablation study using all our model variants to understand the effect of the randomization technique described in Section 4.1. The use of randomization is motivated by the fact that the number of possible actions can be different when adapting to a new domain or at test time. The result of this study is shown in Table 9.

The results show that adding the possible actions without randomization (i.e., w/ Possible Actions w/o randomization) 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 randomization 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, WD-T5-Large EM score increased by 5.1 points. The performance increase of WD-T5-Small is due to the fact that input length is reduced at training time.

A.4.2. Cross-Domain Zero-Shot Setting

Similar to the in-domain experimental setup, we performed an ablation study of the randomization technique described in Section 4.1 in the cross-domain zero-shot setting. Table 10 shows the result of this study.

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 2, 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^{+})

Table 9. Randomization ablation results of in-domain WD on ABCD test set. w/ Possible Actions w/o randomization represents the results without randomization, and w/ Possible Actions the results with randomization as described in Section 4.1. EM and CE are the Exact Match and Cascading Evaluation, respectively

MODEL	EM/CE
WD-T5-SMALL	44.1/67.9
w/ POSSIBLE ACTIONS W/O RANDOMIZATION	42.6/66.9
w/ POSSIBLE ACTIONS	44.8/68.6
WD-T5-BASE	47.7/69.9
w/ POSSIBLE ACTIONS W/O RANDOMIZATION	48.1/70.1
w/ POSSIBLE ACTIONS	49.5/72.3
WD-BART-LARGE	42.0/60.3
w/ POSSIBLE ACTIONS W/O RANDOMIZATION	43.2/61.0
w/ POSSIBLE ACTIONS	44.9/64.3
WD-PEGASUS-LARGE	49.9/71.2
w/ POSSIBLE ACTIONS W/O RANDOMIZATION	51.9/72.0
w/ POSSIBLE ACTIONS	52.1/72.6
WD-T5-LARGE	50.6/73.1
w/ POSSIBLE ACTIONS W/O RANDOMIZATION	54.6/74.8
w/ POSSIBLE ACTIONS	**55.7/75.8**

Table 10. Randomization ablation results of cross-domain zero-shot WD on ABCD test set with ”Shirt FAQ” and ”Promo Code” as target domains. w/ Possible Actions w/o randomization represents the results without randomization, and w/ Possible Actions the results with randomization as described in Section 4.1. EM and CE are the Exact Match and Cascading Evaluation metrics, respectively.

MODEL	SHIRT FAQ EM/CE	PROMO CODE EM/CE
WD-T5-SMALL	42.3/65.1	41.0/64.8
w/ POSSIBLE ACTIONS W/O RANDOMIZATION	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 W/O RANDOMIZATION	**47.8/69.7**	**45.7/69.0**

resolves this issue and improves performance on both model variants.

A.5. Ablation on Using Natural Language Action Names

In ABCD, the action names are given in a label-like format (e.g., ”pull-up-account”). However, this format could have a negative effect on the WD performance when using pre-trained models that might not have seen such a format during training since they are usually trained on well-written, well-structured text (Raffel et al., 2020a; Karpukhin et al., 2020). To validate this claim, we manually generated natural language (NL) action names for each action. For example, we choose ”pull up customer account” as the NL action name for ”pull-up-account” (See Table 13 for the full list). Then, we run an experiment similar to 6.1.1 to compare the performance of all the model variants using either the original label-like names (e.g., ”pull-up-account”) or the NL action names (e.g., ”pull up customer account”). For this experiment, we only compare the Action-only EM and Action-only CE since we are interested in the performance of correctly generating the action names and not the slot values. Table 11 shows the results of this experiment.

We observe that using NL names yields better performance on both metrics on all model variants, where it increased the Action-only EM by an average of 2.9. Our qualitative analysis showed that when using the original labels, most of the failures compared to using NL names are related to incorrectly generating the hyphen in the label. For example, the models generate ”pullup-account”, or ”pull-up account” instead of the target ”pull-up-account”. Further, this behavior is more prominent for labels that consist of a single word (i.e, ”instructions” and ”membership”) since they represent only 7% of the labels, where the models generate invalid labels like ”instructions - ” or ”-membership”. Moreover, 60% of the errors made on these single-word labels are for the ”membership” label. We believe this is due to the existence of a similarly named label (i.e., ”search-membership”). Some invalid predictions include stop words (e.g., ”pull-up-the-account”). While extending training could resolve these issues, it will negatively affect domain adaptation due to overfitting. Also, while

Table 11. Results of using original label-like names compared to natural language action names. Action-only EM/CE are the EM and CE scores for action-only prediction, excluding slot values.

PARAMS	MODELS	ACTION-ONLY EM/CE WITH ORIGINAL LABELS	ACTION-ONLY EM/CE WITH NL NAMES
60M	T5-SMALL	53.2/74.1	**55.4/78.6**
220M	T5-BASE	52.8/74.1	**56.2/79.4**
406M	BART-LARGE	59.2/66.3	**61.9/68.8**
568M	PEGASUS-LARGE	56.7/76.5	**59.6/81.2**
770M	T5-LARGE	56.6/85.9	**59.9/81.4**

some of these invalid predictions can be handled in a post-processing step, it will require additional effort to evaluate all the possible failures, which can be a lengthy task, especially when adapting to a new domain. When using NL action names, the models still predict names that are not an exact match to the target. For example, the models predict ”pull up the customer account” instead of ”pull up customer account”, or ”ask customer try again” instead of ”ask customer to try again”. However, using NL names enables us to use techniques like stemming and ignoring stop words, avoiding the need to create elaborate post-processing procedures. However, stemming and ignoring stop words were used at most on 23% of the predictions (for T5-Small). The remaining improvements compared to using the original labels resulted from cases where the NL names brought semantic richness. For example, using NL action names, the models performed better on differentiating ”check membership level” from ”get memberships information” as opposed to using the original labels where the models confuse ”search-membership” and ”membership”. However, manually generating the action names can be a laborious task when dealing with a large or frequently updated domain, and we leave exploring more automated ways to generate these names for future work.

A.6. Ablation on Using Models Fine-tuned on Summarization

To understand the effect of using models fine-tuned on summarization, we performed an experiment following the same setup as in Section 6.1.1 without adding the possible action names to the input, and we compare the performance using the same model with and without summarization fine-tuning. We report the results of this experiment in Table 12 on all model variants except PEGASUS-Large since it does not have a pre-trained model that has not been trained on summarization tasks. Further, all models have been fine-tuned on CNN/DailyMail (Hermann et al., 2015) dataset.

Table 12. Results of using original label-like names compared to natural language action names. With and Without Summarization Fine-tuning represents the results with and without the use of models fine-tuned on summarization task. EM and CE are the Exact Match and Cascading Evaluation, respectively.

PARAMS	MODELS	EM/CE WITHOUT SUMMARIZATION FINE-TUNING	EM/CE WITH SUMMARIZATION FINE-TUNING
60M	T5-SMALL	40.4/63.1	**44.1/67.9**
220M	T5-BASE	44.2/65.9	**47.7/69.9**
406M	BART-LARGE	**42.0**/60.2	**42.0/60.3**
770M	T5-LARGE	50.5/**73.1**	**50.6/73.1**

The results show that using models previously fine-tuned on summarization tasks improves the performance of models below 220 million parameters (i.e., T5-Small, and T5-Base). However, it did not have any effect on larger models’ performance. Our result analysis showed that using fine-tuned models mostly improved the performance on slot value extraction for smaller models, especially for slot values with multiple words like full names (e.g., ”joseph banter”). Further, the same performance was attained for larger models with 4.8% fewer training iterations when using summarization checkpoints. These results show that using summarization fine-tuned models for the WD task is beneficial.

A.7. Additional Result Analysis

A.7.1. In-domain Workflow Actions

As described in Section 6.1.1, one of the issues we observed while analyzing the results of the in-domain workflow actions is that all the models perform poorly on actions that have optional slot values. For example, the action ”*pull up customer account*” can use either the account ID or the full name. However, the gold truth will have only one of them. Further, our analysis showed that 37% of the slot values failures arise when both the ”*pull up customer account*” and ”*verify-identity*” occur in the same dialogue. In such a situation, a customer provides the account ID and the full name (present in 32.5% of test dialogues), and ”*pull up customer account*” can use either slot and ”*verify customer identity*” uses both. Using the account ID or the full name for ”*pull up customer account*” is valid from the ABCD's agent guidelines perspective. However, it might not match the gold truth. When increasing the model size, we observed a performance improvement on dialogues with turn counts larger than 20, which represents 38.2% of test dialogues. We believe that This improvement is related to large models' ability to handle longer context (Tay et al., 2021; Ainslie et al., 2020). Table 2 shows that the performance improves with the model size except for the BART-Large variant. Our qualitative analysis showed that it exhibits an interesting behavior where it performs poorly on extracting slot values representing either names, usernames, order IDs, or emails. For example, it predicts ”*crystm561*” instead of ”*crystalm561*”, knowing that the former does not exist in the dataset. The fact that 70% of the test workflows contain slots of this type explains the drop in performance. This observation is also confirmed by the high Action-only EM score, where slot value predictions are ignored. We also noticed that BART-Large performs better on target workflow with at most two actions. However, we did not find a clear explanation for this behavior, and we leave further analysis for future work.

A.7.2. ABCD Cross-Domain Zero-Shot

Our results analysis showed that on the ”Promo Code” domain, all the failures relate to the ”*offer promo code*” action. When the possible actions are not used, most of the failures when using T5- were caused by the model copying parts of the input. However, for larger models, 67% of the failures are caused by the models predicting ”*offer refund*” instead of ”*offer promo code*”. However, it is worth noting that dialogues from the ”Offer Refund” domain unfold similarly to the ”Promo Code” domain. In such situations, a customer is unhappy with a purchase, and the agent offers a refund, similar to offering a promo code. Further, the slot value (i.e., the promo code) is correctly predicted in 80% of these cases (i.e., when predicting ”*offer refund*”). We believe that this is caused by the fact that the slot value of ”*offer refund*” (.i.e., the refund amount) is close to the one of ”*offer promo code*”. However, this does not show in the results since both metrics require both the action and slot values to be correct. In the remaining 33% of the failures, the larger models generated plausible action names such as ”*generate new code*”, ”*create new code*”, and ”*create code*” (See Appendix A.3 for more details plausible action names). These predictions were not deemed valid since their BERTscore was lower than the threshold. On the other hand, larger models predicted actions that are considered valid such as ”*offer code*”, ”*offer the code*”, and ”*provide code*”. When the possible actions are used, the failures due to the ”*offer refund*” confusion are reduced to 33%, showing the efficacy of adding the possible actions to the input. The remaining errors were due to either invalid slot value prediction or invalid generated action. In this setup, 70% of valid predictions used the exact action name provided in the input, and the remaining were generated actions that are considered acceptable. On the ”Shirt FAQ” domain, When the possible actions are not used, 71% of the failures are due to confusion with similar actions such as ”search for jackets”, or ”search for jeans”. The remaining errors were due to bad slot value prediction. Further, larger models generated plausible actions (e.g., ”*search t-shirt*” or ”*information on shirts*” and slot values (e.g., ”does this *t-shirt* shrink” instead of ”does this *shirt* shrink”). See Appendix A.3 for more examples. When the possible actions are added to the input, only 15% of the failures were due to confusion with similar domains (mostly ”Jacket FAQ”), and the remaining failures were due to invalid slot value predictions. In this setup, 83% of valid predictions used the exact action name provided in the input, and the remaining were generated actions that are considered acceptable.

A.7.3. MultiWOZ Cross-Dataset Zero-Shot

We explored the T5-Small model results to better understand the source of the low performance in the configuration where the possible actions are not provided. Our analysis showed that 91% of the failures are cases where the model uses ABCD's actions, and the remaining are caused by copying the input. For all other models, 87% of the failures are due to the models using ABCD's actions. The remaining failures are cases where the models copy part of the input or invalid action or slot value predictions.

When using the possible actions, all large models showed significant performance improvement. However, the results are far from the ones in the few-shot setting. Our qualitative analysis showed that 80% of the failures are due to invalid slot value predictions, especially for actions with more than three slot values. We believe this is related to the fact that the source domains (i.e., ABCD) have at most three slot values per action. The remainder of the failures is due to using ABCD's action names. Further, 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, BART-Large performs better in this setting since MultiWOZ does not contain usernames or email slot values. Furthermore, we noticed that some predicted slot values are more refined than the gold ones (e.g., *museum of science* instead of *museum*). However, our evaluation method does not mark such predictions as valid.

In both with and without possible actions configurations, models larger than T5-Small were able to generate plausible action names. For example, the models generated the following: "*search for a swimming pool*", "*update booking information*", "*search for a table*", "*make a reservation*", "*search for a museum*", "*search for guesthouse*", and "*search for a room*". While some of these predictions can be ambiguous (e.g., "*make a reservation*"), others are very plausible. For example, "*search for a room*" or "*search for guesthouse*" are valid predictions for "*search for hotel*", and "*search for a museum*" is a valid prediction for "*search for attractions*" since the museum is an attraction. While such predictions are not exact matches, they provide valuable domain knowledge, especially in a zero-shot setting. However, our evaluation scheme cannot mark these predictions as valid since they yield a BERTscore lower than the threshold. While lowering the threshold could solve this issue, it will increase the false positives since the action names are short sentences. For example, "*search for hote*" and "*buy hotel*" has BERTScore of 89%, which shows one of the limitations of our proposed evaluation method.

Finally, it is worth noting that in the zero-shot setting, when the list of actions is not provided, the models trained on WD perform a form of intent (or action) and slot induction. Here the models generate the action names and extract the matching slot values rather than just clustering utterances (Hu et al., 2022a) and performing span extraction for slot values (Yu et al., 2022). However, one limitation of using WD as a way to perform intent and slot induction is that using WD on multiple dialogues could yield action names that are semantically similar but use a different syntax (e.g., "*searching for hotels*" and "*Looking for hotel*"). One way of solving such an issue is to use a similarity measure to group the generated actions (or intent) and choose one of them as the cluster label in a post-processing step. One could also feed the chosen action names to the model when performing the WD task, increasing performance as shown previously. However, we leave this exploration for future work.

A.8. Limitations

Due to the text nature of the output in our text-to-text approach, two issues arise during evaluation. First, when evaluating the zero-shot performance, we use the BERTScore to establish if an action was predicted correctly. We chose a high F1-score threshold (i.e., 95%) for a fair evaluation. However, a high threshold induces false negatives. For example, if the gold value is *offer promo code*, predicting *give promo code* and *offer code* give an F1-score of 96.6% and 91.1%, respectively, and with our setup *offer code* is deemed invalid even if it is semantically equivalent. Lowering the threshold can resolve the example described earlier. However, it will increase the number of false positives. For example, if the gold value is *book restaurant*, predicting *find restaurant* gives an F1-score of 93.4%, and here the predicted value is not equivalent to the gold one in MultiWOZ. This issue is more prominent in our case since the action names are short. Using longer descriptions might solve this issue. However, it will increase the model target length, reducing the performance. All this makes choosing the threshold a complex balancing act. Second, a similar issue occurs when evaluating the slot values where a model predicts a more refined value (e.g., predicting *cambridge museum of technology* instead of *museum*). While we can handle both cases with more clever evaluation methods, this adds more complexity to the evaluation process on top of the complexity of the task at hand (i.e., WD task). Finally, in all our experimental settings, we showed that using our conditioning technique drastically improves performance. However, this technique assumes that the possible actions are known, something not always true in a real-world application.

A.9. Datasets

A.9.1. Adapting Existing Dataset for WD

This section describes how we adapt the ABCD and MultiWOZ datasets to the WD task. The described method can be used to adapt any existing dataset to the WD task.

Adapting ABCD From each dialogue, we create the target workflow by extracting the actions and slot values from each action turn in the same order they appear in the dataset. Then, we remove all action turns from the dialogue keeping only agent and customer utterances. We use the same data splits as in ABCD. Furthermore, we use natural language action names (i.e., using ”pull up customer account” instead of ”pull-up-account”) as shown in Table 13. We also report the results of an ablation study in Appendix A.5 showing the benefits of using natural language action names.

Table 13. 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	offer 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	provide 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

Adapting MultiWOZ To Adapt MultiWOZ to the WD task, we create the target workflow by extracting the set of intents and slot values in the same order they appear in the dialogue. We don’t make any modifications to the dialogue utterance since MutliWoz does not include action turns or any extra metadata. Table 14 shows the list of natural language action names we used.

In our work, we manually generate the list of natural language action names and show that it does improve the performance (See Appendix A.5). However, manually generating these names is not scalable when the list of actions is large, and we leave the exploration of automatic generation approaches for future work.

A.9.2. WD DATASET STATISTICS

This section shows the dataset set statistics for the WD task for ABCD and MultiWOZ datasets. Table 15 shows the size for each split, the average workflow length across each dataset, and the number of domains. Figure 6 and Figure 9 show the workflow length distribution for each data split on ABCD and MultiWOZ, respectively. Figure 7 and Figure 8 show the action distribution for each data split on ABCD and MultiWOZ, respectively.

Table 14. 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

Table 15. WD DATASET STATISTICS FOR ABCD AND MULTIWOZ DATASETS.

	# TRAIN SAMPLES	# DEV SAMPLES	# TEST SAMPLES	AVERAGE WORKFLOW LENGTH	# DOMAINS
ABCD	8034	1004	1004	18.5	30
MULTIWOZ	5048	527	544	3	8

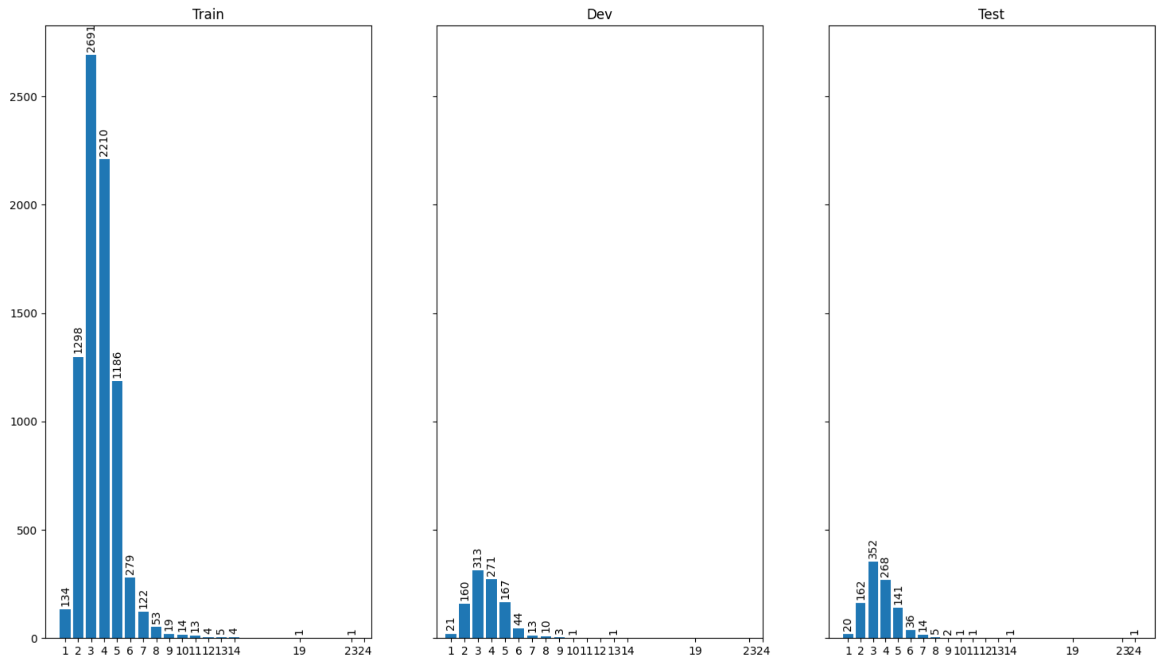


Figure 6. Workflow length distribution for each ABCD dataset split.

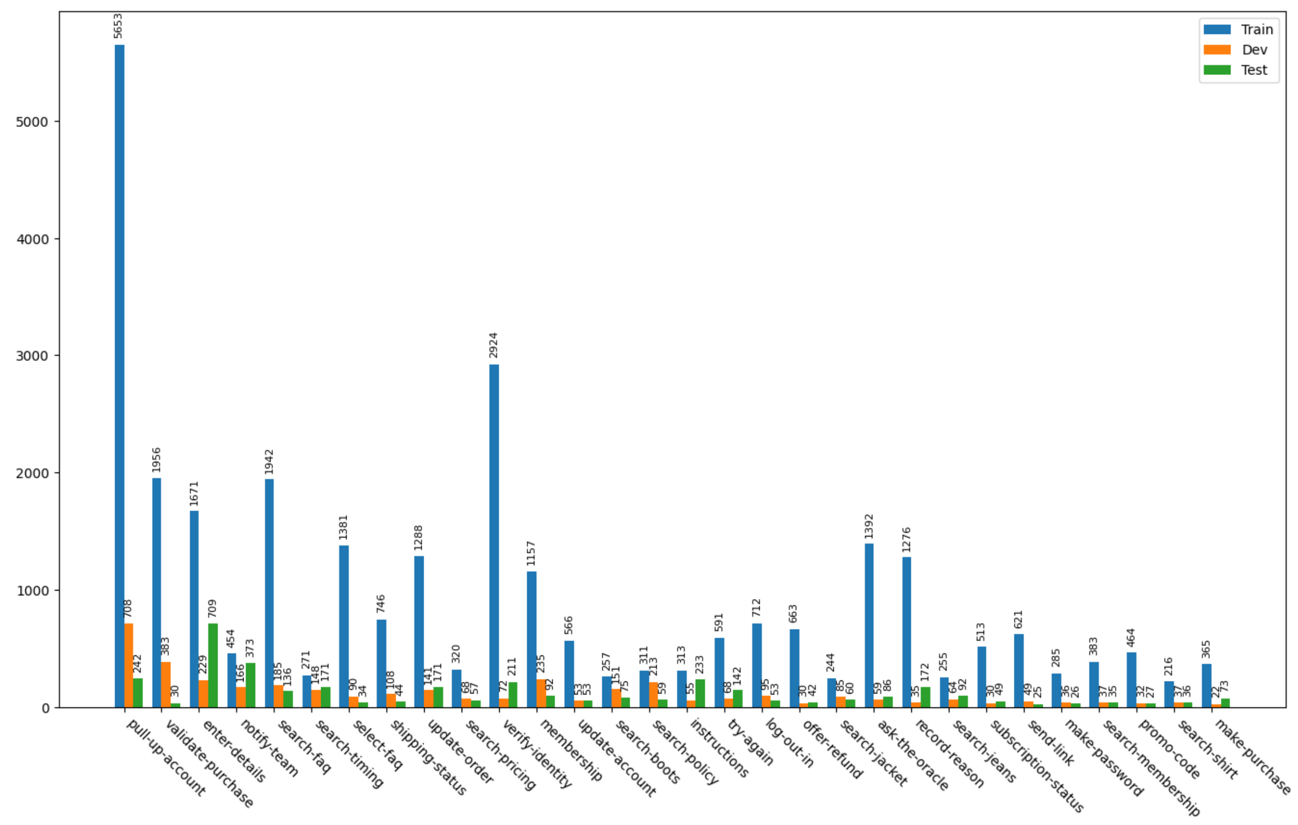


Figure 7. Action distribution for each ABCD dataset split.

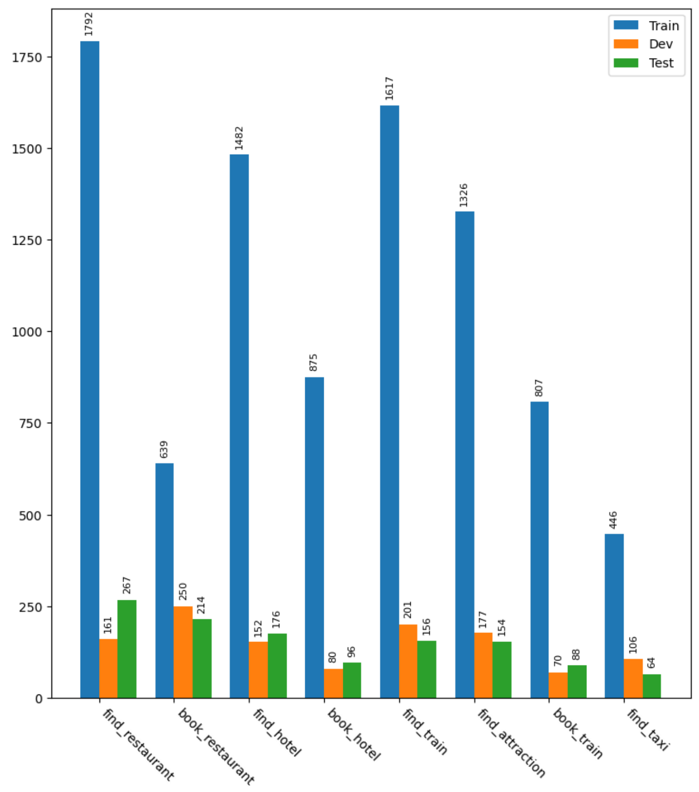


Figure 8. Action (intent) distribution for each MultiWOZ dataset split.

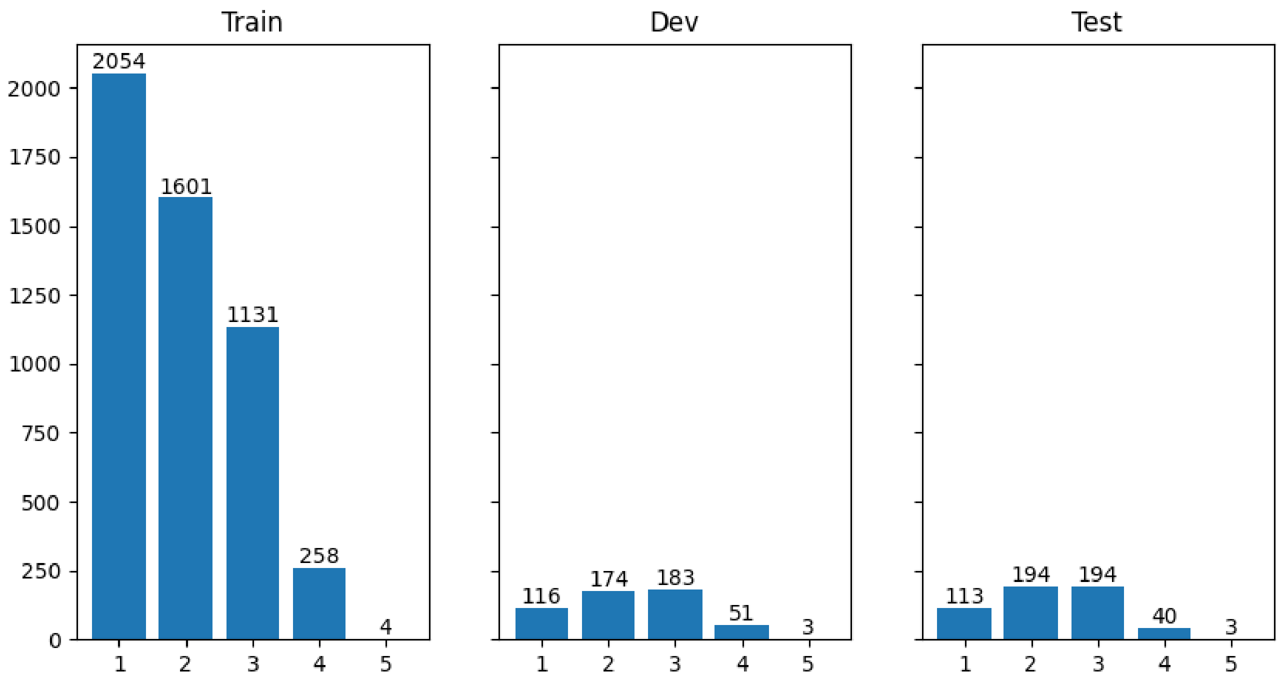


Figure 9. Workflow length distribution for each MultiWOZ dataset split.