Addressing Domain Changes in Task-oriented Conversational Agents through Dialogue Adaptation

Tiziano Labruna
Fondazione Bruno Kessler - Trento, Italy and Free University of Bozen-Bolzano, Italy
tlabruna@fbk.eu

Bernardo Magnini
Fondazione Bruno Kessler - Trento, Italy
magnini@fbk.eu

Abstract

Recent task-oriented dialogue systems are trained on annotated dialogues, which, in turn, reflect certain domain information (e.g., restaurants or hotels in a given region). However, when such domain knowledge changes (e.g., new restaurants open), the initial dialogue model may become obsolete, decreasing the overall performance of the system. Through a number of experiments, we show, for instance, that adding 50% of new slot-values reduces of about 55% the dialogue state-tracker performance. In light of such evidence, we suggest that automatic adaptation of training dialogues is a valuable option for re-training obsolete models. We experimented with a dialogue adaptation approach based on fine-tuning a generative language model on domain changes, showing that a significant reduction of performance decrease can be obtained.

1 Introduction

Most of the recent approaches in task-oriented dialogue systems (McTear, 2020) assume that each component is trained using annotated dialogues. Such data-driven approaches are common both for intent detection and slot filling (Louvan and Magnini, 2020a) and for dialogue state tracking (Balaraman and Magnini, 2021). In this paper, we deal with the situation where we have a conversational dataset, i.e., a collection of annotated dialogues for a certain domain (e.g., booking a restaurant), and then the domain changes (e.g., new restaurants open, or some restaurants change food or price). We investigate to what extent dialogue models trained on the initial dialogues are adequate for the occurred changes, and whether those dialogues can be automatically adapted to domain changes. Being able to manage domain changes is highly relevant for practical purposes, as the process of data collection (in the order of several thousand dialogues for a medium-size domain) for an application domain is both rather complex (e.g., Wizard of Oz (Kelley, 1984)) and very expensive. On the other side, automatic adaptation of training dialogues (Labruna and Magnini, 2021a,b) such that they reflect domain changes, is still a challenging research goal. In the paper, we first provide a definition for the domain changes we are interested in. Second, we set an experimental framework, including evaluation metrics, where we can simulate how the performance of current dialogue models is sensitive to different kinds and amounts of domain changes. Finally, a relevant contribution of the paper is the design and experimentation of unsupervised approaches for training dialogue models from synthetically adapted dialogues.

To be more concrete, Figure 1 presents an example of the situation we are addressing. On the left side we have an initial knowledge base (a) for the restaurant booking domain, and, below it, we show a reservation dialogue (c) between a user and a system. We assume that this dialogue has been human-generated (e.g., through Wizard of Oz) and that it is coherent with the content of the knowledge base. Then, we assume that the knowledge base changes (reported on the right side of Figure 1 (b)), as a new restaurant has opened offering a new kind of food, and one restaurant closed. According to these changes, the initial dialogue is no more consistent with the new domain.

Our intuition is that an adapted dialogue should preserve as much as possible the dialogic structure (e.g., user communicative goals, order of turns, conversation style) of the original dialogue, while it should be adapted to changes occurred in the domain knowledge. In principle, both the user requests and the system responses might be affected by domain changes. On one side the user might be (partially) aware of changes, and adapt its goals and requests (e.g., in Figure 1 the user might be aware of a new restaurant with Poke food, and ask about it). On the other side we assume that the system is fully aware of domain changes and that, in

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop, pages 149–158

Name	Food	Price	Area
Oak Bistro	British	moderate	centre
One Seven	British	expensive	centre
Fitxbillies	British	expensive	centre

(a) Original Knowledge Base

Name	Food	Price	Area
Poke House	Poke	cheap	centre
One Seven	British	expensive	centre
Fitxbillies	British	expensive	centre

(b) Adapted Knowledge Base

USER: I am seeking a restaurant that serves **British** food in the centre.
SYSTEM: I have **3** different options for you. Do you have a price range in mind?
USER: I 'd like a **moderately priced** one.
SYSTEM: **The Oak Bistro** matches your needs.

(c) Original Dialogue

USER: I am seeking a restaurant that serves **Poke** food in the centre.
SYSTEM: I have **one** option for you. Do you have a price range in mind?
USER: I 'd like a **cheap** one.
SYSTEM: **Poke House** matches your needs.

(d) Adapted Dialogue

Figure 1: Domain Dialogue Adaptation. (a) shows the initial situation of the KB, with 3 British restaurants in the center. In (b) the first instance is removed and a new instance is added. (c) is the original dialogue, consistent with the KB, and (d) is the adapted dialogue, according to the changes in the new KB.

order to provide correct information to the user, its responses have to be coherent with such changes. In this context, dialogue adaptations are well defined changes of an original dialogue that make the dialogue coherent with a new domain knowledge. Such adaptation include changing the name of a slot value (e.g., *Poke* in place of *British* in Figure 1), change number of instances (e.g., *one* in place of *3*), change name of instances (e.g., *Poke House* in place of *The Oak Bistro*). Overall, the structure of the initial dialogue has been as much as possible preserved, while modifications aim at reflecting the occurred changes and reconstructing consistency with the new domain.

Although the long-term perspective of our research is to fully automatize dialogue adaptation, the goal of this paper is limited to an investigation of the main issues behind it, following the research directions mentioned above. Particularly, we are interested in two aspects: first, assessing the impact of domain changes in current dialogue state tracking models; second taking advantage of the generative capacity of large pre-trained language models (e.g., (Chen et al., 2019) (Raffel et al., 2020) (Li et al., 2022)) to provide appropriate dialogue adaptations.

More specifically, Section 2 presents the relevant background on task-oriented dialogues, particularly on dialogue state tracking and dialogue manager. Sections 3.1 and 3.2 face domain changes due to, respectively, slot-value and instance alterations, showing their impact on dialogue models. Section 4 introduces dialogue adaptation, which is put into practice with some initial experiments presented in Section 5 and corresponding results, which are discussed in Section 6. Finally, Section 7 presents dialogue adaptation in the context of recent work in the field.

2 Background on Task-oriented Dialogues

This section provides background on task-oriented dialogues, with a focus on how domain knowledge is represented, data-driven approaches, dialogue state tracking and dialogue manager.

2.1 Domain Knowledge

According to most of the recent literature (Budzianowski et al., 2018; Bordes et al., 2017; Mrkšić et al., 2017), we consider a task-oriented dialogue between a system and a user as composed of a sequence of turns $\{t_1, t_2, ...t_n\}$. The goal of the dialogue system is to retrieve a set of entities (possibly empty) in a domain knowledge base (KB) that satisfy the user's needs. A domain ontology O provides a schema for the KB and typically represents entities (e.g., RESTAURANT, HOTEL, MOVIE) according to a pre-defined set of slots S (e.g., FOOD, AREA, PRICE, for the RESTAURANT domain), and values that a certain slot can assume (e.g., EXPENSIVE, MODERATE and CHEAP, for the slot PRICE).

On the basis of the entities defined in the domain ontology, the KB is then populated with instances of such entities. As in most of the literature, we distinguish *informable slots*, which the user can use to constraint the search (e.g., AREA), and *requestable slots* (e.g., PHONENUMBER), whose values are typ-

ically asked only when a certain entity has been retrieved through the dialogue. At each turn in the dialogue, both the user and the system may refer to facts in the KB, the user with the goal of retrieving entities matching his/her needs, and the system to propose entities that can help the user to achieve the dialogue goals.

2.2 Dialogue State Tracking

In a task-oriented system, the Dialogue State Tracker (DST) is responsible for maintaining a record of all information exchanged throughout the entire dialogue history, up to the current step. A dialogue state d_i for a turn t_i is typically represented as a set of slot and slot-value pairs, such as {PRICE=MODERATE, FOOD=ITALIAN}, meaning that at t_i the system assumes that the user is looking for an Italian restaurant with a moderate price. A common method for collecting task-oriented conversational data-sets is through the Wizard of Oz technique. This approach involves two individuals: one person plays the role of a user who asks for information on a particular topic, while the other person acts as a system and provides the requested information. After being collected through Wizard of Oz, the turns of each dialogue are annotated with the corresponding *dialogue state*, consisting of an intent and a set of slot and slot-value pairs. The following is an example of the annotation provided in MultiWOZ 2.1 (Budzianowski et al., 2018):

USER: I would like a moderately priced restaurant in the west part of town.
INFORM(PRICE=MODERATE, AREA=WEST)

SYSTEM: There are three moderately priced restaurants in the west part of town. Do you prefer Indian, Italian or British?
REQUEST(FOOD)

USER: Can I have the address and phone number of the Italian location?
INFORM(PRICE=MODERATE, AREA=WEST, FOOD=ITALIAN)
REQUEST(ADDRESS,PHONE-NUMBER)

Evaluating dialogue state tracking. The most common metric for dialogue state tracking is the Joint Goal Accuracy (JGA), which measures the proportion of correct dialogue states predictions at each dialogue turn. A prediction is considered correct if the slot-values v_i for all slots s_i in the dialogue turn are correctly predicted. Assuming that we have n slot-values in the utterance, the JGA for a single dialogue turn t can be defined as follows (Kumar et al., 2020):

$$JGA(t) = \mathbb{1}_{((\sum_{i=1}^{n} \mathbb{1}_{y_i=\hat{y}_i})=n)} \quad (1)$$

where y_i is the ground truth slot-value, $\hat{y}_i$ is the predicted slot-value and $\mathbb{1}_{x=y}$ is a variable that takes the value of 1 if $x = y$, 0 otherwise.

2.3 Dialogue Manager

Given a certain dialogue state, the goal of the Dialogue Manager (DM) component (Kwan et al., 2022; Liu and Lane, 2017) is to decide the best action to take next, which typically consists of an intent and a list of slot-value pairs. As an example, an output action for the DM can be RESTAURANT-INFORM (FOOD_TYPE=ITALIAN, AREA=CENTER), where the system decides to return a response message with intent RESTAURANT-INFORM and ITALIAN and CENTER as slot values for the slot names FOOD_TYPE and AREA. Then, a generation component will be in charge of actually generating responses, like *We have several Italian restaurants in the city centre.*

Evaluating dialogue manager. Dialogue manager is typically evaluated in terms of effectiveness of suggested dialogue actions to achieve the user's goals (Papangelis et al., 2012). Evaluation is carried out in a reinforcement learning setting with rewards, either through interactions with users or, more frequently, using a dialogue simulator (El Asri and Trischler, 2017). For the purposes of our investigation, we are not interested in measuring action effectiveness. Rather, we aim at estimating the impact of domain changes on the dialogue manager behaviour. According to this perspective, we evaluate the correctness of the system responses provided by the DM, given the dialogue history (i.e., the dialogue states). A response is judged as correct if the information it conveys is consistent with the current knowledge base of the dialogue system. For instance, in the example reported in Figure 1, the response *I have 3 different options for you.* is correct if in the knowledge base there are three restaurants that serve British food and they are located in the centre, otherwise, this response is considered as incorrect. We check the system's utterance correctness considering both the case where

the system presents instances whose slot-values do not correspond in the KB, and cases where the number of instances in the response does not match with the KB. Accordingly, we define the dialogue manager correctness DMC as follows:

$$DMC = (\sum_{i=1}^{n} \mathbb{1}_{C(u_i)})/n \qquad (2)$$

where n is the total number of utterances in the dialogue, $\mathbb{1}_{\mathbb{x}}$ is equal to 1 if $\mathbb{x}$ is True, 0 otherwise. $C(u)$ is True if the utterance u is evaluated as correct with respect to the current KB, False if at least one of the domain information in the utterance is evaluated as incorrect with respect to the KB.

3 Domain Changes

In this section we define a number of domain changes that will be investigated in the paper.

3.1 Changing Slot-values

The first type of domain change is *slot-value change*. This occurs every time a slot-value v used to describe an existing instance in the initial knowledge base is changed with another slot-value (see Figure 1 for an example). This change may involve an already existing slot-value (e.g., a certain restaurant moved from INDIAN to PIZZA food, assuming that PIZZA was already used for other instances), or a new slot-value (e.g., moving from INDIAN to MEDITERRANEAN, which was never used before). An important side effect of slot-value changes is that they modify the distribution of slot-values through instances. For example, after some changes, it might be the case that the initial distribution (e.g., 30% INDIAN restaurants and 10% PIZZA restaurants) is significantly modified (e.g., 20% INDIAN and 20% PIZZA). This is relevant because we need to reflect the same distribution in the test dialogues. As it will be clarified in section 4, this is achieved by substituting occurrences of the initial slot-value (e.g., INDIAN) with occurrences of the new slot-value (e.g., PIZZA) till the domain distribution is reached.

3.2 Changing Instances

The second type of domain change is *changing instances*, where a new instance (e.g., a new restaurant) is added to the domain knowledge, or an existing instance is removed. Adding a new instance implies that the KB slot-value distribution varies, as it has been already noted in Section 3.1, and we assume that its impact follows a similar pattern. However, changing instances may also affect the system's responses, as the dialogue in Figure 1 illustrates. Here, in the initial dialogue, there are three restaurants in the KB that satisfy the user query (FOOD=BRITISH, AREA=CENTER), while in the new KB one restaurant has been removed, and therefore the same query would require a different response from the system. Particularly, assuming that such responses have to be consistent with the KB, the initial dialogue should be adapted so as to be consistent with instance changes.

4 Dialogue Adaptation

In this section, we provide a definition for the *dialogue adaptation* task, as well as its main features.

4.1 Task Definition

Dialogue adaptation consists in modifying a task-oriented dialogue D_0, collected for a certain knowledge base KB_0, with the goal of reflecting a modified knowledge base KB_1, where KB_0 and KB_1 share the same domain ontology O (i.e., they share domain entities and thier slots). The resulting dialogue D_1 is adapted to KB_1 if: (i) D_1 is still a coherent dialogue; (ii) D_1 is consistent with the domain KB_1. We distinguish between initial and adapted training dialogues (notated, respectively, with D_0_train and D_1_train), and initial and adapted test dialogues (D_0_test and D_1_test). Although our definition is neutral with respect to how D_0 is collected, we assume that D_0 are human-generated dialogues, and that the adapted D_1 should maintain the main characteristics of human-human dialogues. We will detail those requirements in the next sections.

4.2 Maintaining Dialogue Coherence

As a first requirement, dialogue adaptations need to maintain the internal coherence of a dialogue, meaning that if an entity is mentioned in a portion of a dialogue, then appropriate references have to be maintained in the rest of the dialogue. As an example of dialogue coherence, in Figure 1(c) the user is looking for *British* food in the *centre*, the system asks for the price range, and the user indicates *moderate* as his/her preference; finally, the system proposes a restaurant that is consistent to all the requests made throughout the dialogue. If the

system proposed a *Poke* restaurant instead, the dialogue coherence would not have been maintained. An automatic adaptation procedure should preserve the coherence of all turns of the dialogue.

4.3 Preserve Domain Adherence

A second requirement for dialogue adaptation is the need to preserve consistency with domain knowledge. Here there are two aspects to consider: adaptation of the system's responses and adaptation of the user's queries. As for the system's responses, the assumption is that the system has complete knowledge of the domain, and adaptations are necessary so that the training dialogues contain responses based on correct information, this way allowing correct training of the DM component. As for user utterance adaptation, users may be partially aware of the domain, and of the changes that may have occurred (e.g., a new popular type of food is served in many restaurants). This means that the user goals may also change, and this fact has to be reflected through dialogue adaptation.

4.4 Maintaining Language Variability

Dialogue adaptations should preserve as much as possible the language variability of human-like dialogues. This is relevant both for maintaining the naturalness of dialogues and for favouring the robustness of the models that are trained. There are several aspects of language variability that need to be considered, including lexical variability, e.g., semantically related expressions, like synonyms, and syntactic variability, e.g., passive forms, or left dislocation. In terms of automatic adaptation procedures, rule-based adaptation (e.g., based on patterns) is likely to produce repetitive dialogues with low variability, while approaches based on generative language models may work better.

4.5 Respect Morpho-Syntactic Constraints

A quite obvious requirement is that adaptation should respect morpho-syntactic constraints of the language, such as the agreement for genre and number, and tense for verbs. As an example, in Figure 1, dialogue adaptation has involved changing the plural *options* into the singular *option*, to respect the agreement with, respectively, *three* and *one*.

5 Experiments

We now define an experimental framework for simulating domain changes in a conversational system. We have two main goals: (i) investigate the impact of the domain changes defined in section 3 on a model trained on D_0_train dialogues and tested on D_1_test adapted dialogues; (ii) simulate the use of a model trained on adapted D_1_train dialogues and tested on D_1_test adapted dialogues. For the first goal, we consider both a DST model (for slot-value changes) and a DM model (for instance changes), while for the second goal we carry on a comparison on a DST model.

5.1 General Experimental Setting

We assume the availability of a data-set of annotated training and test dialogues, mostly following the MultiWOZ (Budzianowski et al., 2018) style. We also assume that such dialogues reflect the information described in the knowledge base of the dialogue system (cfr. Section 2.1), in the sense that the system responses should be as much as possible coherent with the domain knowledge. The experiments presented in the paper are all carried out on MultiWOZ 2.3 (Eric et al., 2020). For all the experiments we consider four incremental amount of changes (25%, 50%, 75%, and 100%), randomly selected from the different domains (e.g., RESTAURANT, HOTEL, ATTRACTION) of the MultiWOZ 2.3 knowledge base, and from their slots (e.g., FOOD, PRICE, DESTINATION).
As for training DST models, we used two well-known approaches: TRADE and TripPy.

TRADE (Wu et al., 2019) consists of three components: an utterance encoder based on bi-directional GRU, responsible for transforming the utterance into a fixed-size vector; a slot gate, which determines whether a slot-value appears in the utterance; a state generator, which predicts the slot that is triggered by the dialogue. The model shares all parameters across multiple domains, being able to perform few-shot and zero-shot learning for the DST task.

TripPy (Heck et al., 2020) is a DST model based on a triple copy strategy that, in order to select the best slot-value for its predictions, performs the following operations: copying the value from user utterances, copying values from system utterances, inferring new values from values that are already present in the dialogue state. It involves BERT as the front-end context encoder and it is equipped with a slot gate for each domain-slot pair.

As for dialogue manager (DM), we employ

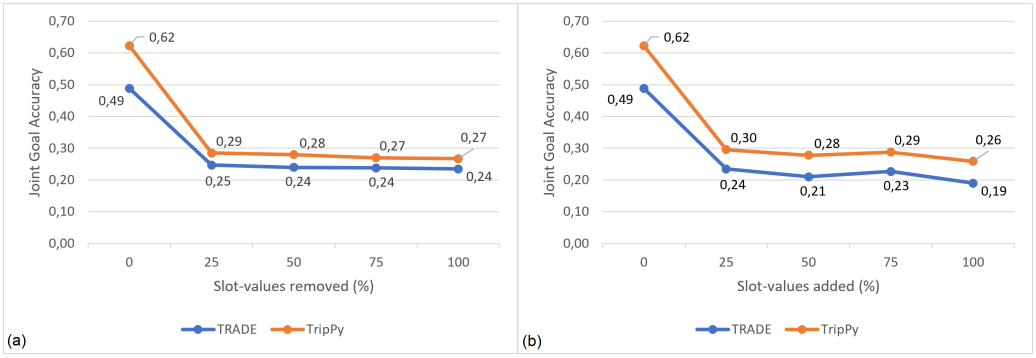

Figure 2: Impact on DST. TRADE and TripPy JGA when trained on D_0_train and tested on adapted D_1_test, with (a) increasing percentages of slot-values removed, and (b) increasing percentages of slot-values added.

a simple algorithm that, given the intent and the slot-values predicted for a user's message, queries the KB and returns the best action based on the query results (e.g. if no entities are found in the KB, it returns the action NO-RESULTS).

5.2 Impact of Slot-value Changes

In the following experiments a DST model is first trained on D_0_train dialogues, and then tested on adapted D_1_test dialogues, in order to assess the impact of slot-value changes. D_1_test dialogues for both Experiment1 and Experiment2 are automatically produced through a dialogue adaptation procedure based on a large pre-trained language model.

Dialogue adaptation procedure. We have adopted a dialogue adaptation strategy based on the method proposed in (Labruna and Magnini, 2022). There are three steps:

- (i) Modify KB_0 introducing a specified amount of changes, i.e., introducing or removing slot-values and instances.
- (ii) From the resulting KB_1, using a set of manually defined patterns, we extract a corpus of about 100K textual sentences; this corpus is used to fine-tune a pre-trained language model (we use BERT (Devlin et al., 2019)).
- (iii) Once BERT has been fine-tuned on KB_1, we use the resulting model (i.e., $BERT_{KB_1}$) to predict slot-values to be substituted in D_0_test test dialogues. We prompt $BERT_{KB_1}$ masking all the slot-values in D_0_test, and, in order to maximize the slot-value language variability, we randomly select from the top ten predictions returned by the model.

The resulting D_1_test has exactly the same amount of dialogues of the original D_0_test provided by MultiWOZ 2.3. More specific details of this procedure are presented in the next paragraphs. As an example, the following user utterance from D_0 dialogues: "I'm looking for an **Indian** restaurant in the **north** part of town" will first be masked as follows:
"I'm looking for an **[MASK]** restaurant in the **[MASK]** part of town"
and finally, we will ask $BERT_{KB_1}$ to predict the substitutions to the masks, which will produce something like:
"I'm looking for an **English** restaurant in the **northwest** part of town"
which populates D_1_test dialogues. Note that the substitutions are produced sequentially from left to right, therefore the first prediction will condition the subsequent ones.

Generating the fine-tuning patterns. In order to fine-tune BERT on our domain, we select a number of utterances - both from user and system - randomly taken from the original MultiWOZ dataset. Then, we mechanically substitute the slot-values in the utterance with all the possible slot-values from KB_1 (the target domain) that have the same slot as the original one. For example, in the utterance "I'm looking for an **Indian** restaurant" the value INDIAN is substituted with all values with the slot RESTAURANT-FOOD in KB_1.

Experiment1: Removing slot-values. Here the goal is to quantify the impact of removing existing slot-values on a DST model. We have defined four versions of modified KB_1 (25%, 50%, 70%, 100%), where we have removed increasing amounts of slot-values from KB_0, randomly substituting them with values that are not removed.

Percentages refer to the proportion between the slot-values that are removed and those kept (e.g., 100% means that the same number of slot-values are removed and kept). In order to choose which slot-values are to be removed, we used an algorithm that minimizes the difference between the actual percentage of removed values and the desired percentage. Note that, after removing, the number of instances in KB_0 and KB_1 is exactly the same, although the slot-value distribution is changed. As for evaluation, we use adapted D_1_test dialogues generated by the dialogue adaptation procedure described in this section, applied on the four versions of KB_1.

Experiment 2: Introducing new slot-values. Here the goal is to quantify the impact of introducing new slot-values (unseen in KB_0) on a DST model. We have defined four versions of modified KB_1 (25%, 50%, 70%, 100%), where we have added an increasing amount of slot-values from KB_0. The percentages refer to the proportion between the new slot-values and the old ones (e.g. 100% means that the number of the new values is the same as the old ones). The new slot-values where taken from a number of different databases, taking care to preserve the domain affinity with respect to each slot. After the addition, the number of instances in KB_0 and KB_1 is exactly the same, although the slot-value distribution is changed. As for the evaluation, we use adapted D_0_test dialogues, generated by the dialogue adaptation procedure described in this section, on the four versions of KB_1.

5.3 Assessing the impact of Instance Changes

We now aim at assessing the impact on the dialogue manager (DM) component caused by introducing or removing domain instances. We start from the MultiWOZ KB_0 and randomly simulated variations both increasing and reducing the number of instances, to obtain KB_1. We consider only the RESTAURANT domain since it is the most complete and well-representative of all domains. The dialogue manager is evaluated checking whether its responses on the D_0_test MultiWOZ dialogues are consistent with the modified KB_1. The intuition is that as new instances are added or removed, the DM capacity to correctly predict the next action would correspondingly decrease. As evaluation metric we used dialog manager correctness (DMC), introduced in Section 2.3.

Experiment 3: Increasing the number of instances. In the increasing setting, we considered five incremental percentages of instance addition (10, 20, 30, 40 and 50). Each new instance was created by selecting random slot-values from those already in KB_0 (no new slot-value is added).

Experiment 4: Decreasing the number of instances. In the reduction setting, we used the same percentages for deciding how many instances have to be randomly removed at each variation of KB_0. Each of the modified KBs was then compared to the original MultiWOZ dialogue and the corresponding DMC correctness is assessed.

5.4 Training on Adapted Dialogues

In this experiment we apply dialogue adaptation on D_0_train dialogues, build a DST model on top of such adapted D_1_train dialogues, and evaluate the resulting model against adapted D_1_test test dialogues. The goal is to investigate whether automatic dialogue adaptation on D_0_train can reduce the decrease in performance of the DST model.

Experiment 5: Training on adapted slot-values. This is similar to Experiment 2 in Section 5.2, introducing new slot-values, with the difference that now, instead of training DST on D_0_train, it is trained on adapted D_1_train dialogues. To produce D_1_train we apply the same procedure used in Experiment 2 to produce D_1_test and defined in this Section. Note that, although the same dialogue adaptation procedure is applied to both training and test data, being independently run on D_0_train and D_0_test, the resulting adaptations may still differ, and slot-values that are present in D_1_train may not appear in D_1_test.

6 Results and Discussion

In this section we present the results obtained on the five experiments introduced in Section 5.

6.1 Impact of Slot-value Changes

As for Experiment 1, Figure 2(a) plots the variation of TRADE and TripPy global joint accuracy at different slot-value removing rates. We observe that removing slot-values in the dialogues brings a massive impact on the degradation of the DST models. Both models show a similar degradation pattern, with TripPy having better scores in general. In both cases, however, the JGA shows a decrease

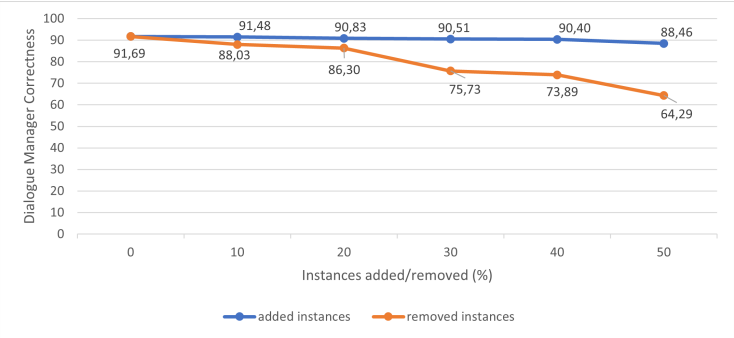

Figure 3: Impact on DM. Correctness when different amounts of instances are added or removed.

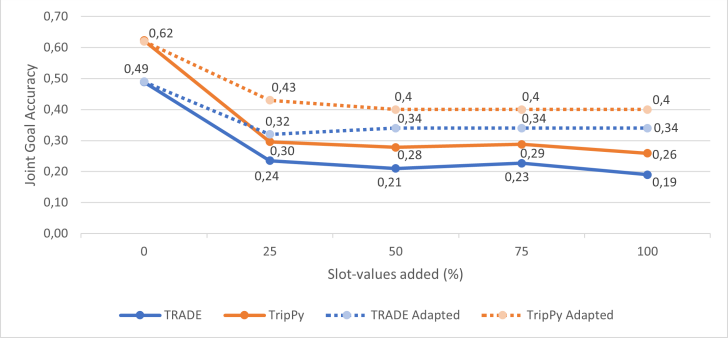

Figure 4: Performances (JGA) of DST models when trained on non-adapted dialogues (continuous line) and adapted dialogues (dotted line) for incremental additions of slot-values.

of about 50% with respect to the zero-change situation.

As for Experiment 2, Figure 2(b) plots the variation of TRADE and TripPy when we incrementally introduce the new slot-values. We note that, again, TripPy scores slightly better than Trade, with both models showing similar degradation rates. As for Experiment 1, the JGA decreases of around 50% with respect to the zero-change situation, showing that the two DST models are poorly robust to the domain changes we have introduced.

For both experiments, the JGA difference among increasing changes is minimal, with a loss of a couple of points in the adding situation, and about 5 points in the removing situation. This can be explained because the current dialogue adaptation procedure does not guarantee that all the changes in KB_0 are actually reflected in D_1_test after BERT fine-tuning.

6.2 Impact of Instance Changes

Figure 3 plots the variation of DM correctness (DMC) when we incrementally add new instances, or reduce them (Experiments 3 and 4). Here, reducing instances comes with a more significant degradation, while adding instances brings the score down by only 3 points. This is due to the fact that cutting off pieces of knowledge, until halving it, has a much stronger impact on the system responses in the dialogues, rather than adding new knowledge, which leaves all previous information untouched. For instance, if the system provides information on a specific restaurant, DMC is not affected by the number of new restaurants that have been introduced; on the other hand, a system presenting to the user information on a restaurant that is not present anymore in the KB, will lead to a failure situation. This is very clear in the plot, with the DMC decreasing by up to 27 points when the reduction rate is set to 50%.

6.3 Training on Adapted Slot-values

As for Experiment 5, Figure 4 compares the DST models when trained without any dialogue adaptation (i.e., using the original MULTIWOZ training) and with dialogue adaptation. We see that the automatic adaptation results in a significant improvement with respect to the no-adaptation situation. For instance, adding 100% of the slot values, TripPy gains 35% JGA (from .26 to .40) when trained on automatically adapted D_1_train. On the other side, the performance degradation from

the initial no-change situation is about 51% for both models. This is because, while the original dataset was collected manually, the adapted dataset uses values automatically generated by the fine-tuned BERT, which still introduces noise values.

7 Related Work

This section presents relevant work related to dialogue adaptation. Although the idea of adapting dialogues to reflect domain changes is, to the best of our knowledge, original, there have been numerous attempts to modify or extend training data in order to make models more robust to unseen dialogue phenomena.

Delexicalization. The most similar approach to dialogue adaptation is delexicalization, which consists of substituting the slot-values in the dialogue with their corresponding slots (e.g., "*I'm looking for Italian food*" with "*I'm looking for* FOOD-TYPE *food*"), or other placeholders (Wen et al., 2016; Wang et al., 2022). Although the idea is that the dialogue model can generalize enough for recognising different slots for similar KBs, delexicalization, in practice, does not achieve good performance. On the MultiWOZ data-set, a Trade DST model trained on delexicalized slot-values has a Joint accuracy of 0.014, with a significant decrease in performance.

Data augmentation. The idea behind data augmentation in dialogue modeling (Louvan and Magnini, 2020b,c) is to automatically create new training data by applying changes to existing data, without altering their fundamental characteristics. In the case of a conversational data-set, new utterances are created substituting every slot-value with a different value taken from the values for the same slot (e.g., from the utterance *I want to go to the north*, we can create new utterances substituting "south", "west" and "east" to "north").

8 Conclusion

Dialogue adaptation is useful when collecting and annotating new dialogues for certain domain changes becomes too costly, and a cheap solution is preferable. The idea is that domain changes (e.g., new slot-values, new instances) are reflected in training dialogues through corresponding automatic adaptations. We have provided empirical evidence that current dialogue models, both dialogue state tracking and dialogue manager, are strongly affected by domain changes, with a significant decrease in performance. We discussed a number of issues that make automatic dialogue adaptation a challenging task. As for future work, the main goal is to improve dialogue adaptation techniques. While the use of pre-trained generative language models fine-tuned to the specific domain changes is promising, the amount of generated noise is still high, and more work is necessary to better constraint the slot-value generation to achieve human-like performance.

References

Vevake Balaraman and Bernardo Magnini. 2021. Domain-aware dialogue state tracker for multi-domain dialogue systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:866–873.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *ICLR*. OpenReview.net.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Layla El Asri and Adam Trischler. 2017. Safe evaluation of dialogue management. In *Proceedings of WiNLP*.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 422–428, Marseille, France. European Language Resources Association.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. 2020. Trippy: A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGdial 2020, 1st virtual meeting, July 1-3, 2020*, pages 35–44. Association for Computational Linguistics.

John F. Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Trans. Inf. Syst.*, 2(1):26–41.

Adarsh Kumar, Peter Ku, Anuj Goyal, Angeliki Metallinou, and Dilek Hakkani-Tur. 2020. Ma-dst: Multi-attention-based scalable dialog state tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8107–8114.

Wai-Chung Kwan, Hongru Wang, Huimin Wang, and Kam-Fai Wong. 2022. A survey on recent advances and challenges in reinforcement learningmethods for task-oriented dialogue policy learning. *arXiv preprint arXiv:2202.13675*.

Tiziano Labruna and Bernardo Magnini. 2021a. Addressing slot-value changes in task-oriented dialogue systems through dialogue domain adaptation. In *Proceedings of RANLP 2021*.

Tiziano Labruna and Bernardo Magnini. 2021b. From cambridge to pisa: A journey into cross-lingual dialogue domain adaptation for conversational agents. *CLiC-it 2021*.

Tiziano Labruna and Bernardo Magnini. 2022. Fine-tuning bert for generative dialogue domain adaptation. In *Text, Speech, and Dialogue*, pages 490–501.

Changye Li, David Knopman, Weizhe Xu, Trevor Cohen, and Serguei Pakhomov. 2022. GPT-D: Inducing dementia-related linguistic anomalies by deliberate degradation of artificial neural language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1866–1877, Dublin, Ireland. Association for Computational Linguistics.

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 482–489. IEEE.

Samuel Louvan and Bernardo Magnini. 2020a. Recent neural methods on slot filling and intent classification for task-oriented dialogue systems: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 480–496, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Samuel Louvan and Bernardo Magnini. 2020b. Simple data augmentation for multilingual nlu in task oriented dialogue systems.

Samuel Louvan and Bernardo Magnini. 2020c. Simple is better! lightweight data augmentation for low resource slot filling and intent classification. *arXiv preprint arXiv:2009.03695*.

Michaael McTear. 2020. *Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots*. Morgan and Claypool Publishers.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788. Association for Computational Linguistics.

Alexandros Papangelis, Vangelis Karkaletsis, and Fillia Makedon. 2012. Evaluation of online dialogue policy learning techniques. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1410–1415.

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

Weizhi Wang, Zhirui Zhang, Junliang Guo, Yinpei Dai, Boxing Chen, and Weihua Luo. 2022. Task-oriented dialogue system as natural language generation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2698–2703.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. *arXiv preprint arXiv:1603.01232*.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, Florence, Italy. Association for Computational Linguistics.