MultiWOZ 2.2 : A Dialogue Dataset with Additional Annotation Corrections and State Tracking Baselines

Xiaoxue Zang[1], Abhinav Rastogi[1], Srinivas Sunkara[1], Raghav Gupta[1], Jianguo Zhang[2], Jindong Chen[1]
[1]Google Research, [2]University of Illinois at Chicago
[1]{xiaoxuez, abhirast, srinivasksun, raghavgupta}@google.com,
[2]jzhan51@uic.edu, [1]jdchen@google.com

Abstract

MultiWOZ (Budzianowski et al., 2018) is a well-known task-oriented dialogue dataset containing over 10,000 annotated dialogues spanning 8 domains. It is extensively used as a benchmark for dialogue state tracking. However, recent works have reported presence of substantial noise in the dialogue state annotations. MultiWOZ 2.1 (Eric et al., 2019) identified and fixed many of these erroneous annotations and user utterances, resulting in an improved version of this dataset. This work introduces MultiWOZ 2.2, which is a yet another improved version of this dataset. Firstly, we identify and fix dialogue state annotation errors across 17.3% of the utterances on top of MultiWOZ 2.1. Secondly, we redefine the ontology by disallowing vocabularies of slots with a large number of possible values (e.g., restaurant name, time of booking). In addition, we introduce slot span annotations for these slots to standardize them across recent models, which previously used custom string matching heuristics to generate them. We also benchmark a few state of the art dialogue state tracking models on the corrected dataset to facilitate comparison for future work. In the end, we discuss best practices for dialogue data collection that can help avoid annotation errors.

1 Introduction

Task-oriented dialogue systems have become very popular in the recent years. Such systems assist the users in accomplishing different tasks by helping them interact with APIs using natural language. Dialogue systems consist of multiple modules which work together to facilitate such interactions. Most architectures have a natural language understanding and dialogue state tracking module to generate a structured representation of user's preferences from the dialogue history. This structured representation is used to make API calls and as a signal for other modules. Then, the dialogue policy module determines the next actions to be taken by the dialogue system. This is followed by the natural language generation module, which converts the generated actions to a natural language utterance, which is surfaced to the user.

Recently, data-driven techniques have achieved state-of-the-art performance for the different dialogue systems modules (Wen et al., 2017b; Ren et al., 2018; Zhang et al., 2019; Chao and Lane, 2019). However, collecting high quality annotated dialogue datasets remains a challenge for the researchers because of the extensive annotation required for training the modules mentioned above. Many public datasets like DSTC2 (Henderson et al., 2014), WOZ (Wen et al., 2017a), SimulatedDialogue (Shah et al., 2018), MultiWOZ (Budzianowski et al., 2018), TaskMaster (Byrne et al., 2019), SGD (Rastogi et al., 2019), etc. have been very useful to facilitate research in this area. Among these datasets, MultiWOZ is the most widely used benchmark for dialogue state tracking. It contains over 10,000 dialogues spanning 8 domains, namely - Restaurant, Hotel, Attraction, Taxi, Train, Hospital, Bus, and Police.

Since its inception, the MultiWOZ dataset has undergone a few updates. Lee et al. (2019) introduced user dialogue actions providing a structured semantic representation for user utterances. Eric et al. (2019) fixed 32% of dialogue state annotations across 40% of the turns and introduced slot descriptions, culminating in MultiWOZ 2.1, a new version of the dataset. Despite the large scale of corrections introduced in MultiWOZ 2.1, there are still many unaddressed annotation errors (Zhang et al., 2019). Furthermore, several approaches to dialogue state tracking use span annotations identifying the locations in the user and system utterances where slot values have been mentioned, to make the system efficient and generalizable to new slot values (Rastogi et al., 2017; Wu et al., 2019; Zhang et al., 2019; Rastogi et al., 2019; Xu and Hu, 2018; Zhou and Small, 2019; Gao et al., 2019). Because of the absence of these span annotations in MultiWOZ, these approaches resort to generating them using custom string matching heuristics, making their comparison difficult.

To address these limitations, we introduce MultiWOZ 2.2[1], an updated version of the MultiWOZ dataset. Our contributions are threefold.

1. We identify the annotation errors, inconsistencies, and ontology issues in MultiWOZ 2.1, and publish its improved version.

2. We add slot span annotations for user and system utterances to standardize them across future models. We also annotate the active user intents and requested slots for each user utterance.

3. We benchmark a few state-of-the-art dialogue state tracking models on the corrected dataset to facilitate comparison for future work.

The paper is organized as follows. First we describe the different types of annotation errors and inconsistencies we observed in MultiWOZ 2.1 (Section 2). Then, we outline the redefinition of ontology (Section 3), followed by the description of correction procedure (Section 4) and new annotations we introduce (Section 5). Finally, in Section 6, we present the performance of a few recent dialogue state tracking models on MultiWOZ 2.2.

2 Annotation Errors

The MultiWOZ dataset was collected using a Wizard-of-Oz setup (Kelley, 1984). In this setup, two crowd-workers are paired together, one acting as a user and the other as the dialogue agent. Each dialogue is driven by a unique set of instructions specifying the user goal, which are shared with the crowd-worker playing the role of the user. After every user turn, the crowd-worker playing the role of the dialogue agent (wizard) annotates the updated dialogue state. After updating the state, the tool shows the set of entities matching the dialogue state to the wizard, who then uses it to generate a response which is sent to the user. Remaining annotations such as the system actions are collected using a second annotation task.

[1]The dataset is available at `https://github.com/budzianowski/multiwoz`.

The Wizard-of-Oz setup is widely considered to produce natural conversations, as there is no turn level intervention guiding the flow of the dialogue. However, because of its heavy reliance on humans for generating the correct annotations, the procedure is prone to noisy annotations. We identified two major classes of errors outlined below, which were not corrected in MultiWOZ 2.1.

2.1 Hallucinated Values

Hallucinated values are present in dialogue state without being specified in the dialogue history. We observed four different types of such errors, which are shown in Figure 1 and described below.

1. **Early Markups:** These values have been mentioned by the agent in a future utterance. Since the user has not accepted them yet, they should be excluded from the dialogue state.

2. **Annotations from Database:** These values are not mentioned in the dialogue at all, even in the future utterances. They appear to be incorrectly added by the wizard based on results of the database call.

3. **Typos:** These values cannot be found in the dialogue history because of a typographical error. These errors occur since slot values are entered as free-form text in the annotation interface.

4. **Implicit Time Processing:** This specifically relates to slots taking time as a value. Sometimes, the value is obtained by adding or subtracting some pre-determined duration from the time specified in dialogue history (Figure 1). In other cases, it is implicitly rounded off to closest quarter (Dialogue 1 in Figure 2). This further burdens models with learning temporal arithmetic.

We observed that the errors mentioned above are quite frequent. In total we found that hallucinated values appear in 3128 turns across 948 dialogues in the MultiWOZ 2.1 dataset.

2.2 Inconsistent State Updates

We also encountered annotations in MultiWOZ 2.1 that are semantically correct, but don't follow consistent annotation guidelines. Inconsistencies arise in the dialogue state because of three main reasons:

1. **Multiple Sources:** A slot value may be introduced in the dialogue state through various sources. It may either be mentioned by the user,

Example Dialogue Segment	**MultiWOZ 2.1**	**MultiWOZ 2.2**
1. Early Markup		
User: Help me find a moderate priced british food place please. Sys: restaurant one seven is a nice place. Do you want to book?	r-food=british, r-pricerange=moderate, r-name=one seven	r-food=british, r-pricerange=moderate
2. Annotation from Database		
User: Can you give me the address to the hospital in Cambridge? Sys: The address is Hills Rd, Cambridge Postcode: CB20QQ	hospital-department=acute medical assessment unit	-no update-
3. Typo		
Sys: Okay, I can help with that. What day and time would you like to dine and how many people should I have the reservation for? User: On Thursday at 5:00. I also need a hotel in the same area. No need to have free parking.	r-bookday=thursday, r-booktime=15:00, hotel-area=west	r-bookday=thursday, r-booktime=5:00, hotel-area=west
4. Implicit Time Processing		
User: Can I get the postcode for that? I also need to book a taxi to the Golden Wok.	r-name=Golden Wok, r-bookday=friday, r-booktime=11:00, taxi-leaveAt=friday, taxi-destination=Golden Wok	r-name=Golden Wok, r-bookday=friday, r-booktime=11:00, taxi-destination=Golden Wok
Sys: The postcode is cb21tt. Are you looking for a taxi from Old Schools to the Golden Wok?		
User: Yes I do. I'd like to make sure I arrive at the restaurant by the booked time. Can you check?	r-name=Golden Wok, r-bookday=friday, r-booktime=11:00, taxi-leaveAt=friday, taxi-arriveby=10:45	r-name=Golden Wok, r-bookday=friday, r-booktime=11:00, taxi-arriveby=11:00

Figure 1: Examples of hallucinated values in MultiWOZ 2.1 and the corrections in MultiWOZ 2.2. Please note that we omit state annotations unrelated to the extracted utterances. "r" used in the slot name in the right two columns is an abbreviation of restaurant.

offered by the system, carried over from another slot in the dialogue state of a different domain, or be a part of the ontology.

2. **Value Paraphrasing:** The same slot value can be mentioned in many different ways, often within the same dialogue e.g. the value "18:00" for the slot time may be mentioned as "6 pm", "1800", "0600 pm", "evening at 6" etc.

3. **Inconsistent tracking strategy:** Crowd-workers have inconsistent opinions on which slot values should be tracked in the same dialogue context. For example, some workers track all slot values that the user agrees with while others only track user-specified slot values.

Table 1 shows dialogue state update from three different sources for similar slots from different dialogues in MultiWOZ 2.1. In the first case, the value "08:00" for slot *train-arriveby* comes from the ontology, despite the presence of an equivalent value "8:00" in the user utterance. On the other hand, in the second example, the slot value in the dialogue state comes from the user utterance despite the ontology listing "17:45" as a value for the slot *train-leaveat*. In the third example, the value of *train-leaveat* is not derived from any of the sources mentioned above, but is generated by incorporating the semantics. The slot value can be mentioned in multiple ways, but in order to evaluate a dialogue system fairly, it's necessary to either maintain a consistent rule for deciding how the value is picked among all the mentions or consider all the mentions as the correct answer. MultiWOZ 2.1 gives one unique correct answer for each dialogue state but lacks an explicit rule on how it is determined. This inconsistency confuses the model during training and unfairly penalizes it during evaluation if it

Source	User utterance	Dialogue state update
Ontology	I need to arrive by 8:00.	train-arriveby=08:00
Dialogue history	Sometime after 5:45 PM would be great.	train-leaveat=5:45pm
None	I plan on getting lunch first, so sometime after then I'd like to leave.	train-leaveat=after lunch

Table 1: Example of slot values annotated using different strategies in "PMUL0897.json", 'MUL0681.json'", and "PMUL3200.json" in MultiWOZ 2.1.

outputs a slot value which is different but equivalent to the one listed in ground truth.

Figure 2 shows another example where dialogue states are updated differently in similar scenarios. In both dialogues, the system offers an instance that fulfills the user's requirement, but the dialogue states are updated differently after user shows an intent to book the ticket. Specifically, in dialogue 1 the value for *train-arriveby* provided by the system is tracked in the dialogue state while not in dialogue 2. Dialogue 1 also showcases the implicit time processing issue discussed in Section 2.1, where the time "12:08" has been rounded to "12:15" in the dialogue state.

3 Ontology Issues

Although MultiWOZ 2.0 provides a predefined ontology which is claimed to enumerate all slots and the possible values for every slot, it has been reported to be incomplete. As a result, many researchers have built their own ontology to achieve a better performance (Wu et al., 2019; Goel et al., 2019). To fix the problem of incompleteness, MultiWOZ 2.1 rebuilt the ontology by listing all values present in dialogue states across the dataset, but it still has some unaddressed issues.

First, for some slots, multiple values sharing the same semantics are listed. Some examples are "8pm" and "20:00", "a and b guesthouse" and "a and b guest house", "cheap|moderate" and "moderate|cheap" for the slots *restaurant-book-time*, *hotel-semi-name* and *hotel-semi-pricerange* respectively. We find that 51% of the values for the slot *hotel-name* are not semantically unique, and similar figures for the *restaurant-name* and *attraction-name* slots. Such duplicate values make evaluation hard since MultiWOZ 2.1 only assumes one correct value for each slot in the dialogue state.

Second, we observe multiple slot values in the ontology that can't be associated with any entities in the database. Values like "free" for slot *attraction-name*; "cam", "dif", and "no" for slot *restaurant-name* are some examples. Such values could be introduced in the ontology because of typographical errors in the utterances or annotation errors. Our investigation shows that 21.0% of the slot values in the ontology can't be directly mapped back to the values in the database through exact string matching. We also observed a few logical expressions like "cheap|moderate", "NOT(hamilton lodge)" etc. in the ontology. We believe that these expressions, although semantically correct, add noise during training. The ontology should either omit such expressions altogether or include all possible expressions to enable generalization to cases not observed in the training data.

4 Correction Procedure

To avoid the issues described above, we advocate the definition of ontology prior to data collection. This not only serves as a guideline for annotators, but also prevents annotation inconsistencies in the dataset and corruption of the ontology from typographical and annotation errors. This section describes our definition of the new ontology, which we call *schema*, followed by the corrections made to the state and action annotations. Lastly, we also show the statistics of our modifications.

4.1 Schema Definition

It is not realistic for the ontology to enumerate all the possible values for some slots like *restaurant-name* and *restaurant-booktime*, which can take a very large set of values. With addition or removal of entities in the database, the set of possible values also keeps changing continuously. Rastogi et al. (2019) proposed a representation of ontology, called *schema*, to facilitate building a scalable dialogue system that is capable of handling such slots. A *schema* divides the different slots into two categories - *non-categorical* and *categorical*. Slots with a large or dynamic set of possible values

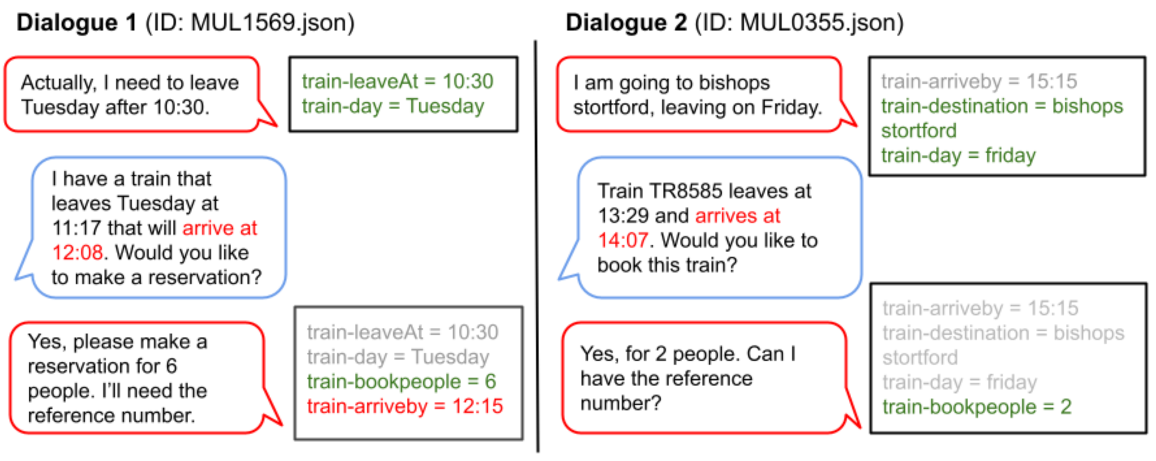

Dialogue 1 (ID: MUL1569.json)
Actually, I need to leave Tuesday after 10:30.
train-leaveAt = 10:30
train-day = Tuesday
I have a train that leaves Tuesday at 11:17 that will arrive at 12:08. Would you like to make a reservation?
Yes, please make a reservation for 6 people. I'll need the reference number.
train-leaveAt = 10:30
train-day = Tuesday
train-bookpeople = 6
train-arriveby = 12:15
Dialogue 2 (ID: MUL0355.json)
I am going to bishops stortford, leaving on Friday.
train-arriveby = 15:15
train-destination = bishops stortford
train-day = friday
Train TR8585 leaves at 13:29 and arrives at 14:07. Would you like to book this train?
Yes, for 2 people. Can I have the reference number?
train-arriveby = 15:15
train-destination = bishops stortford
train-day = friday
train-bookpeople = 2

Figure 2: Example of dialogues states being updated differently in similar scenarios. In both dialogues, user accepts a train offered by the system. In dialogue 1, *train-arriveby* is annotated in the dialogue state after user's agreement, but not in dialogue 2. Dialogue 1 also shows implicit time processing, where the value 12:08 in the system utterance is rewritten to 12:15 in the subsequent dialogue state.

Domain	**Categorical slots**	**Non-categorical slots**
Restaurant	pricerange, area, bookday, bookpeople	food, name, booktime
Attraction	area, type	name
Hotel	pricerange, parking, internet, stars, area, type, bookpeople, bookday, bookstay	name
Taxi	-	destination, departure, arriveby, leaveat
Train	destination, departure, day, bookpeople	arriveby, leaveat
Bus	day	departure, destination, leaveat
Hospital	-	department
Police	-	name

Table 2: Categorical and non-categorical slots defined for 8 domains in MultiWOZ 2.2.

are called *non-categorical*. Unlike ontology, the schema doesn't provide a pre-defined list of values for such slots. Their value is extracted from the dialogue history instead.

On the other hand, slots like *hotel-pricerange* or *hotel-type*, which naturally take a small finite set of values are called *categorical*. Similar to the ontology, the schema lists all possible values for such slots. Furthermore, during annotation, the values of these slots in the dialogue state and user or system actions must be selected from a pre-defined candidate list defined in the schema. This helps achieve sanity and consistency in annotations.

We define categorical and non-categorical slots for each domain as shown in Table 2. The idea of splitting the slots in MultiWOZ into categorical and non-categorical is not new. Many models have used the number of possible slot values as the classification criterion (Zhang et al., 2019). Similarly, we classify slots with fewer than 50 different slot values in the training set as categorical, and the others as non-categorical.

Note that since the Bus and Police domains have very few dialogues in the training set (5 and 145 respectively), the number of possible slot values in this domain does not reflect the true attributes of the slots. Thus, we classify them by referring to similar slots in different domains instead of following the threshold rule.

4.2 Categorical Slots

The list of all possible values for categorical slots is built from the corresponding database provided with MultiWOZ 2.1. In addition, we allow "dontcare" as a special value, which is used when user doesn't have a preference. We also observe cases where the mentioned value is outside the scope of the database, such as the example below, where MultiWOZ 2.1 specifies "$100" as the value for *hotel-pricerange* in the dialogue state.

User: *Well,I want it cheaper than AIRBNB,so how about $100 a night?*

System: *Unfortunately, we do not have such specific price ranges, but our options are divided into 3 categories: cheap, moderate or expensive. Which would you prefer?*

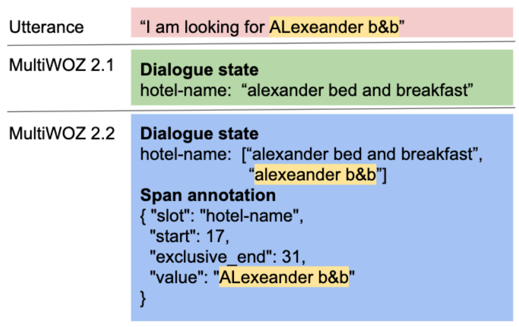
Since "$100" is not a part of the schema, we use "unknown" as the slot value in the dialogue state to express that the requirement specified by the user can not be fulfilled by the schema.

4.3 Non-categorical Slots

Values of non-categorical slots are extracted from the dialogue history. Due to the typographical errors and slot value paraphrasing, the exact value can not be found in many cases. Some examples are "el shaddia guest house" being written as "'el shaddai" or "18:00" being written as "6pm" in the dialogue utterances. Since in practice, typographical errors are inevitable and the same value can be mentioned in variable ways, we try to not modify the utterance to keep the dialogue natural. We also allow the presence of more than one value in the dialogue state. During evaluation, a prediction listing either of the listed values is considered correct.

We use a customized string matching method that takes into consideration the possible typos and alternative expressions to locate all values semantically similar to the annotation. If there are multiple matches, we select the most recently mentioned value and annotate its span. We also add this value to the dialogue state, while preserving the original value. Figure 3 shows the differences between the annotations in MultiWOZ 2.1 and MultiWOZ 2.2. The former only assumes a single value for each slot, even though the slot values can be mentioned in multiple ways and predicting any one of these variants should be considered correct. Thus, in MultiWOZ 2.2, the dialogue state can contain a list of values for a slot: predicting any value in this list is considered correct.

In some cases, slot value is carried over from other slots without being explicitly mentioned in the dialogue. For instance, in the utterance "I need to book a taxi from the museum to the restaurant", the slot value for *taxi-destination* is copied from the value for *restaurant-name* populated earlier. Instead of annotating the span for *taxi-destination*, we note down the original slot that *taxi-destination* copies its value from. The span annotation for such slots can be obtained by tracing back the copy chain. We posit that this information can be beneficial for state tracking models utilizing a copy mechanism.



Utterance "I am looking for ALexeander b&b"

MultiWOZ 2.1 **Dialogue state**
hotel-name: "alexander bed and breakfast"

MultiWOZ 2.2 **Dialogue state**
hotel-name: ["alexander bed and breakfast",
"alexeander b&b"]
Span annotation
```
{ "slot": "hotel-name",
  "start": 17,
  "exclusive_end": 31,
  "value": "ALexeander b&b"
}
```

Figure 3: Example of the difference between dialogue state annotation in MultiWOZ 2.1 and MultiWOZ 2.2 and span annotations in MultiWOZ 2.2.

4.4 User and System Actions

The user and system action annotations provide a semantic representation of the respective utterances. These annotations were not part of the original MultiWOZ 2.0 release. They were created by Lee et al. (2019) and were subsequently added to MultiWOZ 2.1. However, around 5.82% of turns have missing action annotations. We use crowdsourcing to obtain annotations for these 8,333 dialogue turns (7,339 user and 994 system). The slot names used in dialogue acts are slightly different from the ones used in dialogue state annotations. We rename the slots in the dialogue acts to remove this mismatch.

MultiWOZ 2.1 uses domain-specific prefixes to associate actions with a certain domain. A few dialogue acts also have the "Booking" prefix, which is used in a few domains including Restaurant, Hotel and Train whenever a reservation is involved. In these cases, it is difficult to identify the domain corresponding to the action since the same prefix is used across many domains. We eliminate the domain and "Booking" prefixes from the dialogue acts, so that a uniform representation of actions can be used across all domains. To retain the association with the domain, actions for the same domain are grouped together into *frames*, following the representation used by Rastogi et al. (2019).

4.5 Statistics

Table 3 contains statistics on the corrections in the training, dev, and test sets. We observe that the errors are relatively uniformly distributed across the three splits. Combining all the aforementioned procedures, we modify dialogue states in 17.3% of

Dataset	% of state	% of dialogues
train	17.3	27.9
dev	17.3	28.7
test	17.6	29.5

Table 3: The ratio of the modified dialogue states (same as the number of user utterances) and modified dialogues in the training, dev, and test sets.

the user utterances across 28.2% of all dialogues. Out of the total modified 12,375 utterance annotations, a majority of the corrections fix the state update inconsistencies described in Section 2.2 by listing all the different ways in which a value has been mentioned over the dialogue context in the dialogue state. Of these state updates, 1497, or just over 12% involved corrections for two or more slots. Missing action annotations were added in a total of 8,333 utterances, whereas pre-existing actions in MultiWOZ 2.1 were verified and fixed for around 10% of the utterances.

5 Additional annotations

Besides the span annotations, we also add active user intents and requested slots for every user turn. Predicting active user intents and requested slots are two new sub-tasks that can be used to evaluate model performance and facilitate dialogue state tracking. Prediction of active intents or APIs is also essential for efficiency in large-scale dialogue systems which support hundreds of APIs.

- **Active intents:** It specifies all the intents expressed in the user utterance. Note that utterances may have multiple active intents. For example, in "can i get the college's phone number. i am also looking for a train to birmingham new street and should depart from cambridge looking for a train", the user exhibits the intent both to know more about an attraction and to search for a train.

  Based on the action and state annotations, we define a single search intent for the Attraction, Bus, Hotel, and Police domains and a single booking intent for Taxi domain, whereas for the Restaurant, Hotel, and Train domains, both search and booking intents are defined.

- **Requested slots:** It specifies the slots that the user requests information about from the system. This field is generated based on the user actions in each turn. These annotations find direct applicability in developing dialogue policy models, since requesting additional information about entities is very common in task-oriented dialogue.

6 Dialogue State Tracking Benchmarks

Recent data-driven dialogue state tracking models that achieve state-of-the-art performance mainly adopt two classes of methods: span-based and candidate-based. Span-based methods extract values from dialogue history and are suitable for tracking states of non-categorical slots, while candidate-based methods that perform classification on pre-defined candidate lists to extract values are better-suited for categorical slots. To test models' performance on both categorical and non-categorical slots, we selected three dialogue state tracking models that use a mixture of both methods to benchmark the performance on the updated dataset: SGD-baseline (Rastogi et al., 2019), TRADE (Wu et al., 2019), and DS-DST (Zhang et al., 2019).

TRADE considers each slot as a mixture of categorical and non-categorical slot. It uses a pointer generator architecture to either generate the slot value from a pre-defined vocabulary or tokens in the dialogue history. On the contrary, SGD-baseline has separate tracking strategies for categorical and non-categorical slots. It first uses a shared pretrained BERT (Devlin et al., 2018) to encode a context embedding for each user turn, a slot embedding for each slot, and a slot value embedding for each slot value in the candidate list of the categorical slots. Then, it utilizes linear networks to perform classification for the categorical slot and to find start and end span indices for non-categorical slots. DS-DST is a recently proposed model achieving state-of-the-art performance on MultiWOZ 2.1 using pre-trained BERT. The main difference between DS-DST and SGD-baseline is that the context embedding used in DS-DST is conditioned on the domain-slot information while it is not in SGD-baseline.

We use joint goal accuracy as our metric to evaluate the models' performance. The joint goal accuracy is defined as the average accuracy of predicting all the slot values for a turn correctly. The performance of different models is shown in Table 4. In general, we observe similar performance on MultiWOZ 2.1 and MultiWOZ 2.2 across the three models. Table 5 compares the joint goal accuracy over only the categorical slots (cat-joint-acc) and only the non-categorical slots (noncat-joint-acc) across all the models. It shows that TRADE

Model	Multi-WOZ 2.0	Multi-WOZ 2.1	Multi-WOZ 2.2
TRADE	0.486	0.460	0.454
SGD-baseline	-	0.434	0.420
DS-DST	0.522	0.512	0.517

Table 4: Joint goal accuracy of TRADE, SGD-baseline and DS-DST models on MultiWOZ 2.0, MultiWOZ 2.1 and MultiWOZ 2.2 datasets.

Model	Cat-joint-acc	Noncat-joint-acc
TRADE	0.628	0.666
SGD-baseline	0.570	0.661
DS-DST	0.706	0.701

Table 5: Performance of TRADE, SGD-baseline, and DS-DST models on predicting categorical and non-categorical slots. Cat-joint-acc and noncat-joint-acc denote joint goal accuracy on categorical and non-categorical slots respectively.

and SGD-baseline demonstrate considerably higher performance on non-categorical slots than categorical slots. We infer that it may be caused by the corrections ensuring that the value in the dialogue state is also present in the dialogue history for all non-categorical slots.

7 Discussion

The Wizard-of-Oz paradigm is a very powerful technique to collect natural dialogues. However, the process of annotating these dialogues is prone to noise. In this section, we discuss some of the best practices to follow during task-oriented dialogue data collection so as to minimize annotation errors.

It is important to define an ontology or schema before data collection, listing the interface of all the domains and APIs. The schema should identify categorical slots, which have a fixed set of possible values, and the annotation interface should enforce the correctness of these slots. In particular, the interface should only allow the annotator to pick one of the values specified in the schema. For non-categorical slots, the interface should only allow values which have been mentioned in the dialogue history, and display an error otherwise. These simple checks help avoid typographical errors and value paraphrasing issues, discussed in Section 2. The annotation task can be followed by simple validation checks to identify erroneous annotations, which can be fixed by a follow-up crowd-sourcing task. For instance, listing the set of all possible values for every slot in the dataset helped us quickly identify instances listing “thursday” as the value for a time slot or “no” as the name of a hotel.

We also observed a few annotations utilizing logical expressions to represent the dialogue state. For instance, some dialogue state annotations utilize string “cheap>moderate” to mean that cheap is preferred over moderate, and “cinema|entertainment|museum|theatre” to mean that all values separated by “|”are acceptable. However, such values are disproportionately rare in the dataset (<1% of dialogues), thus making it difficult for models to handle such cases. It brings into question how to define a more expressive representation which can support such complex annotations and how we should design the model capable of handling such cases. We hope that as the state tracking technology advances, there will be more focus on this direction. On the other hand, it is important to ensure that such complex constraints are proportionately represented in the dataset if the system is intended to support them.

8 Conclusion

MultiWOZ 2.1 (Eric et al., 2019) is an improved version of the MultiWOZ 2.0 dataset, which is extensively used as a benchmark for dialogue state tracking. We identify annotation errors, inconsistencies and ontology related issues which were left unaddressed in MultiWOZ 2.1, and publish a corrected version – MultiWOZ 2.2. We added a new schema, standardized slot values, corrected annotation errors and standardized span annotations. Furthermore, we annotated active intents and requested slots for each user turn, and added missing user and system actions besides fixing existing ones. We benchmark a few state-of-the-art models on the new dataset: experimental results show that the models’ performance is similar between MultiWOZ 2.1 and MultiWOZ 2.2. We hope the cleaned dataset helps make fairer comparisons among models and facilitate research in this field.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 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.

Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Ben Goodrich, Daniel Duckworth, Semih Yavuz, Amit Dubey, Kyu-Young Kim, and Andy Cedilnik. 2019. Taskmaster-1: Toward a realistic and diverse dialog dataset. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4517.

Guan-Lin Chao and Ian Lane. 2019. Bert-dst: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. *arXiv preprint arXiv:1907.03040*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyag Gao, and Dilek Hakkani-Tur. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tur. 2019. Dialog state tracking: A neural reading comprehension approach. *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*.

Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. 2019. Hyst: A hybrid approach for flexible and accurate dialogue state tracking. *Interspeech 2019*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue*, pages 263–272.

J. F. Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Trans. Inf. Syst.*, page 26–41.

Sungjin Lee, Qi Zhu, Ryuichi Takanobu, Zheng Zhang, Yaoqin Zhang, Xiang Li, Jinchao Li, Baolin Peng, Xiujun Li, Minlie Huang, et al. 2019. Convlab: Multi-domain end-to-end dialog system platform. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 64–69.

Abhinav Rastogi, Dilek Hakkani-Tür, and Larry Heck. 2017. Scalable multi-domain dialogue state tracking. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop*, pages 561–568. IEEE.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*.

Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. 2018. Towards universal dialogue state tracking. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017a. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 438–449.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017b. A network-based end-to-end trainable task-oriented dialogue system. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

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*.

Puyang Xu and Qi Hu. 2018. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1448–1457.

Jian-Guo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wan, Philip S Yu, Richard Socher, and Caiming Xiong. 2019. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. *arXiv preprint arXiv:1910.03544*.

Li Zhou and Kevin Small. 2019. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *arXiv preprint arXiv:1911.06192*.