A Structured Semantic Reinforcement method for Task-Oriented Dialogue

Anonymous ACL submission

Abstract

Recently, many BERT based approaches have been proposed for task-oriented dialogue (TOD) task. Despite their impressive performance, the insufficient utilization of deep semantic information and long-distance context understanding makes it difficult for these methods to digest complex dialogue scenarios for they cannot obtain sufficient evidence from dialogue data to support dialogue decision-making. In this work, we propose a novel structured semantics reinforcement (SSR) method to handle these issues. SSR reorganized the end-to-end TOD structure, which mainly includes two key components: 1. The dialogue symbolic memory, which cache the objects mentioned in the dialogue and the structure under the semantic relationship. 2. semantic projection module, understanding module, based on the previous structured results, determines the source of the slot extraction required for the current task. And our approach achieves state-of-the-art results on dataset MultiWOZ 2.1, where we acquire a joint goal accuracy beyond 60% and also gains a significant effect on dataset DSTC8.

1 Introduction

Task-oriented dialogue, as a focus in the field of conversational AI, has attracted a surging interest from both academia and industry. In a dialogue system, dialogue state tracking (DST) is a sub-task that is defined to be recognizing the meaning and intent in a user utterance, and be able to keep and update this information during the process of the dialog (Young et al., 2010). DST is critical to a dialogue system since it determines the next action that the system can respond to a user utterance. Previous works on DST evolve from traditional approaches that operate on a fixed ontology (Mrkšić et al., 2017; Liu and Lane, 2017; Zhong et al., 2018) to approaches that can handle open vocabulary (Ren et al., 2018; Nouri and Hosseini-Asl, 2018).

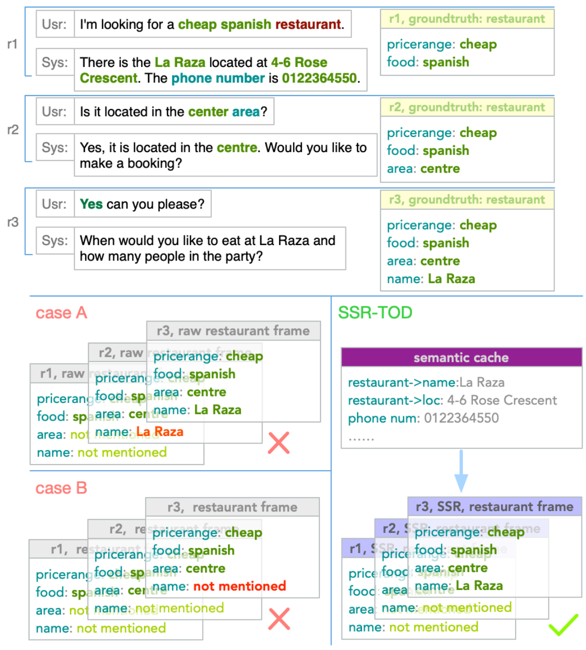

Figure 1: An example of a real conversation being parsed. **case A** shows a traditional approach that immediately recognizes *restaurant-name* when the slot appears but the user has not decided to use it. In **case B**, the method is to follow the user's will, and the slot is not recognized in the round *r2*, but the slot cannot be retrieved when it is needed. In **SSR-TOD**, unlike the result of using NLU directly in traditional DST, we have added symbolic memory here for structured parsing of dialogue description content and caching of information that the task frame needs to use immediately or later.

All these previous works focus on extraction from the user and/or system utterance in the current round, but ignore the fact that some desired results may come from the long-distance dialogue history. Figure 1 shows an illustrating example of such a phenomenon. In the example, the value for the restaurant name appears in the first round of the dialogue, but the dialogue system is not expected to put the restaurant name into a dialogue state since the user has not decided whether to book the recommended restaurant or not. Instead, the value is desired in the third round where a decision is confirmed by the user. This means we need to obtain the value from a long-distance history. Heck et al. (2020) adopt a copy mechanism to obtain slot-values from dialogue state history, but obviously, it cannot solve the problem shown in the example.

To tackle this problem, we propose a novel approach to dialog state tracking. The basic idea is to utilize an additional data structure named *symbolic memory* to store all the candidate slot-values that we can obtain during the conversations and use the memory as a dialogue history. Extensive experiments on two benchmarks demonstrate the effectiveness of the proposed approach.

2 Related Work

Early works on DST consider fixed ontology. Mrkšić et al. (2017) for the first time propose neural models to couple spoken language understanding (SLU) and state tracking. The basic idea is to rely on embedding representations instead of exact matching to retrieve correct answers. (Liu and Lane, 2017) focus on an end-to-end neural network model for task-oriented dialogue. Zhong et al. (2018) propose a globally-locally self-attentive approach to dialogue state tracking where global modules learn parameters shared among slots while local modules learn slot-specific parameters. All these works suffer from scalability and generalization issues. Nouri and Hosseini-Asl (2018) extend the approach proposed in Zhong et al. (2018) by using only one recurrent network with global conditioning instead of (1+# slots) recurrent networks. Ren et al. (2018) propose an approach named StateNet which is independent of the number of values, shares parameters across all slots, and uses pre-trained word vectors. Chao and Lane (2019) for the first time propose to use BERT (Devlin et al., 2019) as encoding dialogue context, including both current utterances and history. Heck et al. (2020) enhance previous BERT-DST with three copy mechanisms, which are used to obtain values from the user utterance, the system utterance, and previous dialogue states.

3 Our Approach

3.1 Problem Formulation

We define a dialogue as a sequence of T rounds of utterance pairs, denoted as as $X = (< U_1, M_1 >, \ldots, < U_T, M_T >)$ where U_t is the user utterance and M_t is the corresponding response from the system in the round t. In each round of dialogue, the goal of the dialogue system is to predict whether the slots under the current domain have been assigned values, and what corresponding values have been assigned. Therefore, we organize the output of each round in the form of domain-slot-pair $S = \{S_1, ..., S_N\}$, where N is the total number of all the slots under all domains.

To solve the above problem, we propose a framework as depicted in Figure 2. Briefly, the framework consists of the following modules.

- **Dialogue Symbolic Memory**. A data structure for storing all the slot-values that appear in the dialogue history.
- **Dialogue Context Encoder**. A BERT-based module that encodes user utterance and system utterance in the current round as well as history.

3.2 Dialogue Symbolic Memory

To build dialogue symbolic memory, we refer to CUED dialogue acts (Steve, 2009) to introduce task-independent ontology and manually establish the mapping from entities to task slots. Figure 3 presents an example. We implement the mapping between schema and entity to slots through simple rules. After the mapping, we get a batch of new entity attributes $P = \{P_1, ..., P_K\}$, where K is the total number of entity attributes introduced by the mapping relationship.

Dialogue symbolic memory is specifically designed to track the slots and properties recognized in the dialogue process. Here, we set a strong heuristic rule for filling slots: if a slot has a corresponding entity property, the slot will be filled first with the value of the entity property, and the slots in the DST will be handled by the semantic projection module.

Take the *venue* entity as an example. There are three types of *venue* entities in MultiWOZ: restaurant, hotel, and attraction. They have common properties: name and area. Therefore, the venue name, venue area, and venue type of the venue are introduced into P as entity properties. For further illustration, for example, in a scene where a restaurant is identified during the dialogue process, its name is “backstreetbistro” and its area is “center”. We recognize the hotel of the next scheduled task. When we need to find the hotel of "the same area

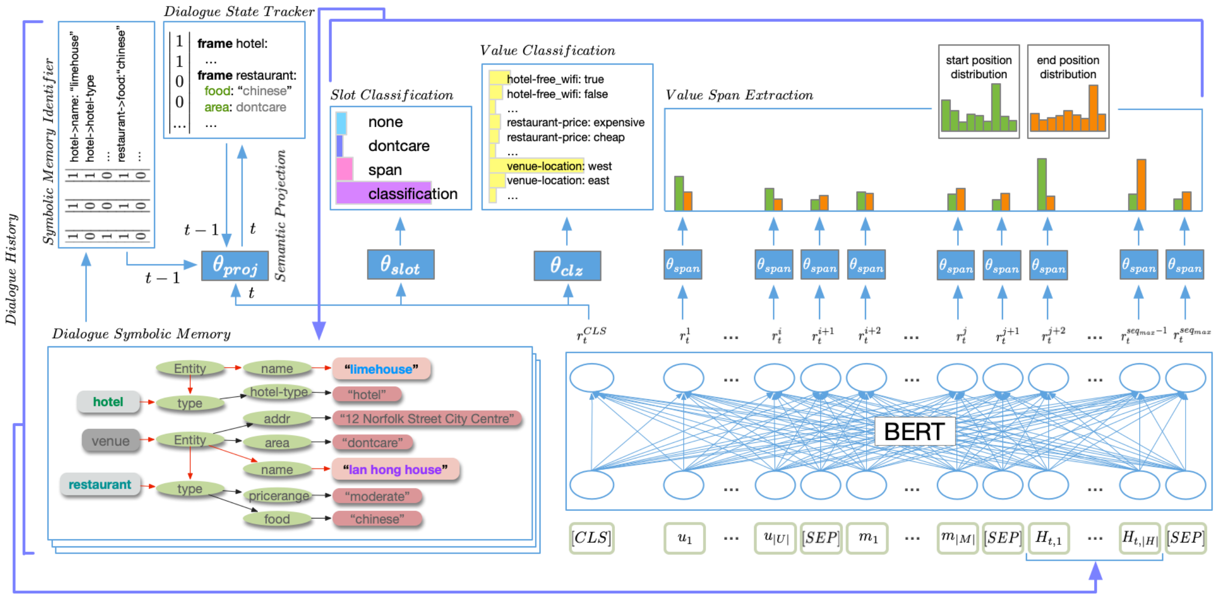

Figure 2: Architecture of the proposed approach.

of", we can display the area information of the previous event from the symbolic memory through the previous schema, instead of expecting our model to learn how to extract the area of the restaurant from the previous dialogue context according to the semantics of "the same area".

Through the introduction of this task-independent schema, we display the relationship (the main relationships include: entity to property, and the same property of different entities under the same type) between many task slots hidden behind the dialogue in the dialogue process to the model. At the same time, because it is task-independent, slots between different domains can share entities and attributes in this way, so that the dialogue task can make full use of the logical relationship behind the dialogue, not just the identified semantic information.

In addition, for the following semantic projection module to learn how DST can get values from the symbolic memory, the symbolic memory identifier

$$i_t^{cache} \in \{0,1\}^{(N+K+1)\times Q} \tag{1}$$

is introduced to mark whether the symbolic memory is filled at each position in round T, the Q here is a hyperparameters. As the number of layers of symbolic memory, with the more layers, and the more historical semantic information can be used. We selected three layers artificially.

3.3 Dialogue Context Encoder

The module is an adaption from the BERT encoding structure in (Chao and Lane, 2019; Heck et al., 2020), as depicted in Figure 1. The input the encoder consists of three parts:

$$R_t = BERT([CLS] \oplus U_t \oplus [SEP] \oplus M_t \oplus \\ [SEP] \oplus H_t \oplus [SEP]), \tag{2}$$

where U_t is the user utterance in the round t, M_t is the corresponding system utterance, and H_t is structured dialogue history from the symbolic memory. Note that the definition of H_t is different from the definition in (Heck et al., 2020) where $H_t = (U_{t-1}, M_{t-1}), \ldots, (U_1, M_1)$ is the history of dialogue up to and excluding round t. The output is used as the encode of the dialogue content in the round t, it consists of two parts: one is r_t^{CLS}, which as the representation of dialogue context sentence-level, will be directly used by the following slot classification module and value classification module to predict the slot type and enumeration value. And each of following

$$R_t = [r_t^{CLS}, r_t^1, ..., r_t^{seq_{max}}] \tag{3}$$

, as the representation of dialogue context token-level, are all corresponding to a token at the corresponding position in the input. Then, this part will be input into the value span extraction module for span-extraction of non-enumerated slot-values.

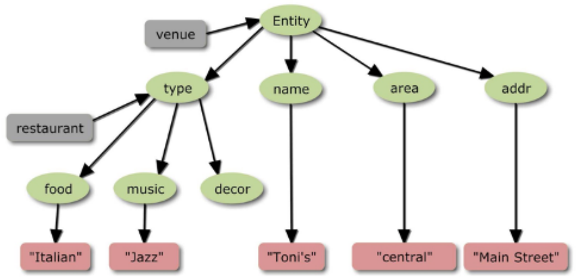

Figure 3: An illustrating example for dialogue symbolic memory.

3.4 Dialogue State Tracker

This part is the same as the DST in the traditional end-to-end method to record the slot filling of each task during the entire dialogue. As before, in order to let the semantic projection module learn how to value, introduce the DST identifier

$$i_t^{DST} \in \{0,1\}^{N+1} \tag{4}$$

to record in each round t, whether each domain-slot-pair is filled.

3.5 Semantic Projection Module

This module is used to learn how DST gets values from symbolic memory:

$$p_{t,s}^{proj}(a_t^{sem} \oplus i_{t-1}^{DST}) = softmax(W_s^{proj}(a_t^{sem} \oplus i_{t-1}^{DST}) + b_s^{proj}) \in \mathbb{R}^{N+1} \tag{5}$$

where

$$a_t^{sem} = r_t^{CLS} \oplus i_{t-1}^{cache} \tag{6}$$

The input consists of three parts: 1. The structured semantic information extracted from the current round T, and because the symbolic memory is a multi-layer structure, it also contains a certain length of historical structured semantic information. 2. For the dialogue status of the previous round, we learn the dialogue status of the current round of t through the historical $T-1$ and the DST status of previous rounds. 3. Semantic representation of sentence-level extracted by BERT.

3.6 Slot Classification Module

This module uses to the sentence-level representation r_t^{CLS} and i_t^{cache} of the previous Dialogue context as input. Also, this module is used to learn if the slot has value (*none* or *dontcare*) and how to get value form the corresponding module value (*span* or *classification*).

$$p_{t,e}^{slot}(a_t^{sem}) = softmax(W_e^{slot}(a_t^{sem}) + b_e^{slot}) \in \mathbb{R}^4, \quad e \in \{S_1, ..., S_N, P_1, ..., P_K\} \tag{7}$$

Span represents the value obtained from the value span extraction module, and class represents the value obtained from the value classification module. In the input part, the i_t^{cache} in here acts as structured historical information. Since the values in the symbolic memory are filled or not, this part of structured information is obviously helpful to the sub-task of the current slot classification.

3.7 Value Classification Module

This module is used to learn enumerated slot value, for example, "true" of if has wifi, "expensive" of restaurant's price range, "north" of location, "5" of star rating, we define the enumeration of slot value as:

$$M = \{M_1, ..., M_L\} \tag{8}$$

The enumerated slot value is L, so we define the module:

$$p_{t,e}^{clz}(a_t^{sem} \oplus p_{t,e}^{proj}(a_t^{sem} \oplus i_{t-1}^{DST})) = W_{t,e}^{clz}(a_t^{sem} \oplus p_{t,e}^{proj}(a_t^{sem} \oplus i_{t-1}^{DST})) + b_e^{clz} = [none, M_1, ..., M_L] \in \mathbb{R}^{L+1} \tag{9}$$

The input here includes the semantic information of BERT, the structured semantics and history of the symbolic memory, and corresponding type of the previous value is added to constrain the learning of the value classification.

3.8 Value Span Extraction Module

This module uses the dialogue context representation $r_t^i, i \in [1, seq_{max}]$ of token-level form previous BERT to predict the start and end position of each slot in the origin utterance. The $\alpha_{t,e}^i$ and $\beta_{t,e}^i$ are corresponding to the start position and end position of each slot e in the symbolic memory in round t.

$$[\alpha_{t,e}^i, \beta_{t,e}^i] = W_e^{span} r_t^i + b_e^{span} \in \mathbb{R}^2, \forall 1 \leq i \leq n \tag{10}$$

$$p_{t,e}^{start} = softmax(\alpha_{t,e}) \tag{11}$$

$$p_{t,e}^{end} = softmax(\beta_{t,e}) \tag{12}$$

$$start_{t,e} = argmax(p_{t,e}^{start}) \tag{13}$$

$$end_{t,e} = argmax(p_{t,e}^{end}) \tag{14}$$

3.9 Dialogue History

When we get current round t and all the proprieties slot e from prediction, and we use the information we get to organize a structured dialogue history to support the next round of predictions.

$$H_t = i_{t-1}^{cache} \otimes (schema_{t-1,e} \oplus token_{t-1,e}) \tag{15}$$

$$schema_e \in \{1, ..., N + K + 1\} \tag{16}$$

The formula above is the unique digital id representation of the schema of each prediction result e, $token_{t-1,e}$ is the normalized word token of the value at e. In all locations where i_{t-1}^{cache} has prediction results, there are corresponding $schema_e \oplus embedding_{t-1,e}$ to structure the representation, pairs separated by $[CLS]$.

4 Experiments

4.1 Experimental Settings

Datasets. We evaluate our approach on two widely used benchmarks: MultiWOZ 2.1 (Eric et al., 2020) and Schema-Guided Dialogue (SGD) (Rastogi et al., 2019). MultiWOZ 2.1 is a very challenging dataset for the task of DST. It contains more than 10,000 multi-domain dialogues defined over a fairly large ontology. The dialogues belong to 5 domains (*train*, *restaurant*, *hotel*, *taxi*, *attraction*) with 30 domain-slot pairs that appear in all portions of the data. SGD consists of over 20k annotated multi-domain, task-oriented conversations between a human and a virtual assistant. These conversations involve interactions with services and APIs spanning 20 domains, such as banks, events, media, calendar, travel, and weather. SGD shares the same ontology with MultiWOZ 2.1 in many domains

DST Models	DSTC8 SGD-All	MulitiWOZ 2.1
BERT-DST (Chao and Lane, 2019)	38.30%	43.40%
TripPy (Heck et al., 2020)	-	55.30%
SimpleTOD (Hosseini-Asl et al., 2020)	-	56.45%
ConvBERT-DG (Jiang et al., 2020)	41.94%	58.70%
TripPy+CoCoAug (Li et al., 2020)	-	60.53%
This Work	**50.70%**	**67.89%**

Table 1: Comparison results of our approach and previous works on SGD-ALL and MultiWOZ 2.1.

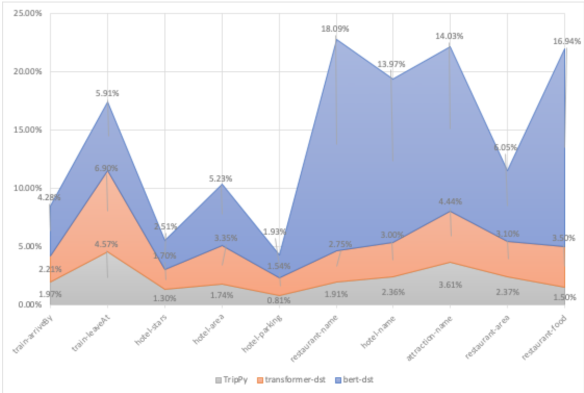

Figure 4: for each slot, measure the JSA difference rate of each scheme (TripPy, transformer-dst, bert-dst) compare to SSR-TOD (from table 2). compare between ontology covered slots and uncovered : right 5 slot is covered by ontology schema, left 5 slot is uncovered. the figure shows that the 5 slots covered by ontology achieve more JSA increment than the 5 slots uncovered among all three schemes JSA difference rate comparison. furthermore, among the 5 slots uncovered, *train-arriveBy* and *train-leaveAt* achieve more JSA than the other 3 slots, for they gain benefits from venues which covered by ontology.

Evaluation. We adopt Joint Goal Accuracy (JGA) as the evaluation metric to measure the overall performance of the models. JGA is defined as the average of prediction accuracies obtained in each round of dialogue. Only if all domains, slots, and values in the dialogue state are predicted correctly, the dialog state prediction is considered correct.

Training Details. Our model is initialized with a pre-trained BERT model that has 12 layers of 768 hidden units and 12 self-attention heads. We set the learning rate and warmup proportion to 1e-4 and 0.1 respectively, and we set the maximum sequence length of BERT input to 256. We use a batch size of 16. The model is trained on a P100 GPU device for 50 epochs.

In the data augmentation phase, we enhanced the original data with the method in (Li et al., 2020) by 8 times and obtained more significant effects.

4.2 Main Results

Table 1 depicts the results of our approach and previous works on MultiWOZ 2.1 and SGD-ALL. From the results we can see that out approach has

slot name	is covered	type	BERT-DST	Transformer-DST	TripPy	SSR-TOD-Aug	BERT-DST diff	Transformer-DST diff	TripPy diff
train-arriveBy	false	span	94.87%	96.79%	97.02%	98.93%	4.28%	2.21%	1.97%
train-leaveAt	false	span	93.14%	92.28%	94.33%	98.64%	5.91%	6.90%	4.57%
hotel-stars	false	classification	96.83%	97.59%	97.98%	99.25%	2.51%	1.70%	1.30%
hotel-area	false	classification	92.61%	94.29%	95.78%	97.45%	5.23%	3.35%	1.74%
hotel-parking	false	classification	95.75%	96.12%	96.81%	97.60%	1.93%	1.54%	0.81%
restaurant-name	true	span	82.98%	95.37%	96.15%	97.99%	18.09%	2.75%	1.91%
hotel-name	true	span	86.61%	95.84%	96.44%	98.71%	13.97%	3.00%	2.36%
attraction-name	true	span	86.32%	94.24%	94.99%	98.43%	14.03%	4.44%	3.61%
restaurant-area	true	classification	93.43%	96.11%	96.80%	99.09%	6.05%	3.10%	2.37%
restaurant-food	true	classification	85.00%	96.04%	97.94%	99.40%	16.94%	3.50%	1.50%

Table 2: joint slot accuracy comparison and analysis: we extracted ten representative slots. The second column indicates whether the slots in the first column are covered by our external knowledge. The first five rows show five uncovered slots, and the last five are covered slots. The third column type indicates how the slot is extracted in our task, span means that the slot is extracted by the value span extraction module, and the classification means that the slot is extracted by the value classification module. The next three columns BERT-DST, Transformer-DST (Zeng and Nie, 2020), TripPy, and SSR-TOD-Aug respectively correspond to the JGA performance of the slot in the above four schemes. The last three columns show the improvement ratio of the slot in the SSR-TOD-Aug scheme compared to the JGA under the other three schemes.

dst model	**with dontcare**	**ignore dontcare**
TripPy	53.83%	55.39%
SSR-TOD base	55.67%	57.90%
SSR-TOD-Aug	61.87%	67.89%

Table 3: different standard and version

obtained performance that greatly surpasses previous systems on both DSTC8 and MultiWOZ 2.1 data sets. The improvements are attributed to two possible reasons. One is the structured knowledge which is incorporated in the previous part. The other is that the network structure, the symbolic memory can carry more structured dialogue history than the traditional models.

4.3 Discussion

4.3.1 Impact of Ontology Knowledge

In our experiment, to facilitate analysis and comparison, we did not cover all the slots with an external ontology schema, as shown in table 2, we selected five covered and uncovered slots, and in these 10 slots, and we also extracted 5 span type and 5 classification type. Table 2 shows the JGA performance of those slots on the original baseline scheme and SSR-TOD-Aug respectively.

It can be seen that in these slots, the JSA of SSR-TOD-aug is better than the previous three schemes in an all-round way. Furthermore, we show the JSA difference rate of SSR-TOD-aug compared to the aforementioned schemes in Figure 4 separately, which can be obviously seen that the JSA difference rate of the left five slots that are not covered by external knowledge is much lower than that of the covered slots, which directly illustrates the importance of external knowledge.

External knowledge is introduced as a schema, and we found why is it so effective when analyzing data case by case. The reason is that the model does not have basic common sense like us, imagine a scene of booking a hotel, the system said "a star rating of 4", the human can realize only hotel has a star rating and guesthouse does not, so the booking type must be a hotel, therefore, the lack of common sense information often leads to ambiguity and interruption of reasonable dialogue.

The introduction of this kind of knowledge information, on the one hand, complements the background information for the dialogue; on the other hand, it externalizes the knowledge contained in the semantics and dialogue flow. Complementary information makes the model, as a learning object without any background knowledge, have the possibility to fully understand the dialogue. And the knowledge externalization allows the model to learn how we make dialogue decisions through explicit features.

In addition to the benefits of introducing knowledge, there is also a very important point is type constraint. From the data point of view, which type of slot is suitable for span or classification has obvious characteristics. For example, there are only a few fixed dishes provided by the restaurant, so classification should be used in slot *restaurant-food*, also, *car-parking* also has two opinions which are true or false. The various place names corresponding to *restaurant-name* cannot be predicted in advance and should be extracted by the span method. Compared with BERT-DST which is a classic method of using span to solve all slot fill-

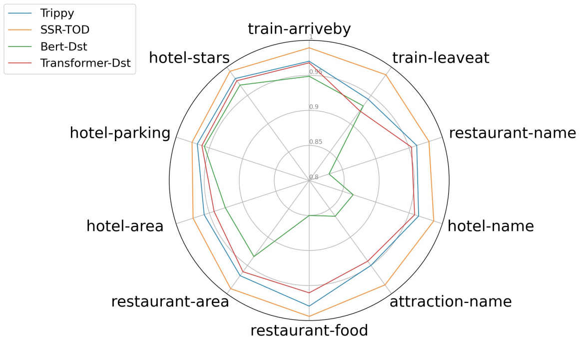

Figure 5: joint slot accuracy compare between TripPy, BERT-DST, Transformer-DST and SSR-TOD-Aug

```
user: Could you help me find a moderately priced chinese food restaurant?
sys:  there are 4. which side of town do you prefer?
restaurant:
    "restaurant-food": "chinese",
    "restaurant-pricerange": "moderate"

user: It doesnt matter what side of town.
sys:  In that case, I would suggest this one lan hong house.
restaurant:
    "restaurant-food": "chinese",
    "restaurant-pricerange": "moderate"
    "area": "dontcare"

user: Okay, can I please have the postcode and address?
sys:  Here's you are: cb12lf, 12 Norfolk Street City Centre.
restaurant:
    "restaurant-food": "chinese",
    "restaurant-pricerange": "moderate"
    "name": "lan hong house"
    "area": "dontcare"
```

Figure 6: example dialogue and slot groundtruth

ing. In hotel-parking extraction, there are only simple scenarios of "yes" and "no", so there is not much difference with SSR-TOD, while restaurant-food has "British", "European", and "gastropub" in many cases, if still use the span method, the effect is reduced compared with the SSR-TOD using classification.

4.3.2 Impact of Semantic Knowledge

The semantic knowledge was represented by symbolic memory and semantic projection module, has two core functions here. semantic projection module is to process the mapping from schema to slot; symbolic memory is to cache semantic information so that the model can handle more complex scenarios, such as semantics across multiple rounds, and multiple-choice cases.

And the mapping in the semantic projection module is based on simple rules. The key point here is that we did not completely cover the entire task with a set of schema, and the reason is, on the one hand, in the implementation of the algorithm, the closer to the details, the more obvious the inherent characteristics of the data set. It is difficult to achieve unbiased coverage by introducing common knowledge and forced coverage will lead to a lot of extra proofreading and annotation work. On the other hand, ontology knowledge can be regarded as our consistent static understanding of common concepts, and it is more aimed at the description of the concept itself. In the dialogue scenario, ontology knowledge is the common sense of the dialogue participants. and much of the information will not be used by the dialogue participants to make dialogue decisions. Therefore, we only extract and simplify the part that is often described semantically in statistics. Also, to verifying the validity of external knowledge, our experiments did not cover all domains.

Figure 6 shows the effect of the semantic knowledge, and it shows the dialogue in figure one and the groud truth of each round of the slot. It can be seen from figure 6 that in the second round of dialogue, the system recommended "lan hong house", but at this time the user did not confirm the reservation of this hotel, so the name of this hotel should not be in the restaurant frame of DST when the user confirmed the reservation in the next round, the name of the restaurant can be "lan hong house". In the traditional scheme (BERT-DST and Transformer-DST), this situation will not be handled, so the slot must be predicted when it is not needed, resulting in errors.

And TripPy has a trick mechanism, which is also one of the three copy mechanisms. It caches the content on the system side and solves part of the problem to some extent. However, the situation after a round still cannot be handled. As far as the problem itself is concerned, the traditional solutions have not touched the fundamentals, in essence, the assumptions of traditional DST are simple and straightforward, making the task frame not capable of handling this kind of cross-round semantics. And symbolic memory can deal with this situation well. Before the task frame, the information of the restaurant name was stored in symbolic memory, and the name information is passed to the downstream module as a slot during the task frame. At the same time, when the situation of "book a taxi from the hotel to the restaurant" occurs, the hotel and restaurant mentioned above can also be found in the symbolic memory according to the schema structure without ambiguity. This is why SSR-TOD's ability to understand dialogue is stronger than traditional solutions.

5 Conclusion

We display a new perspective for dealing with the TOD problem, compared with the traditional method of sending NLU results directly to DST to process dialogue tasks. In addition to the traditional NLU information extraction, we have additionally added a mechanism for organizing and caching knowledge under the new assumptions. So that DST can focus on the processing of specific dialogue tasks, thereby expanding the boundaries and capabilities of the end-to-end solution that can handle dialogue tasks. We also proved this from the performance of the experiment.

This is our attempt under the new dialogue hypothesis. SSR-TOD is a simple attempt. Later we will further look for a more reasonable model structure to realize the new hypothesis.

At the same time, due to the increase of the semantic layer, it is possible to unify various tasks in the dialogue interaction scenario.

References

Guan-Lin Chao and Ian Lane. 2019. BERT-DST: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from Transformer. In *Proceedings of Interspeech*, pages 1468 – 1472.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyag Gao, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: Multi-domain dialogue ¨ state corrections and state tracking baselines. In *Proceedings of LREC*, pages 422–428.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gašić. 2020. TripPy: A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of SIGdial*, pages 35–44.

Ehsan Hosseini-Asl, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Proceedings of NeurIPS*.

Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2020. Convbert: Improving bert with span-based dynamic convolution. *arXiv preprint arXiv:2008.02496*.

Shiyang Li, Semih Yavuz, Kazuma Hashimoto, Jia Li, Tong Niu, Nazneen Rajani, Xifeng Yan, Yingbo Zhou, and Caiming Xiong. 2020. Coco: Controllable counterfactuals for evaluating dialogue state trackers. *arXiv preprint arXiv:2010.12850*.

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In *Proceedings of Interspeech*, pages 1777–1788.

Nikola Mrkšić, Diarmui Ó Séaghdha, Tsung-Hsien Wen, Blaise Thoms, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of ACL*, pages 1777–1788.

Elnaz Nouri and Ehsan Hosseini-Asl. 2018. Toward scalable neural dialogue state tracking model. In *arXiv preprint arXiv:1812.00899*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *arXiv preprint arXiv:1909.05855*, pages 422–428.

Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. 2018. Towards universal dialogue state tracking. In *Proceedings of EMNLP*, pages 2780–2786.

Young Steve. 2009. CUED standard dialogue acts. In *Report Cambridge University Engineering Department*.

Steve Young, Milica Gašić, Simon Keizer, Francois Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech Language*, 24(2):150–174.

Yan Zeng and Jian-Yun Nie. 2020. Jointly optimizing state operation prediction and value generation for dialogue state tracking. *arXiv preprint arXiv:2010.14061*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive dialogue state tracker. In *Proceedings of ACL*, pages 1458–1467.