We Know What You Will Ask: A Dialogue System for Multi-intent Switch and Prediction

Chen Shi[1(✉)], Qi Chen[2], Lei Sha[1], Hui Xue[2], Sujian Li[1], Lintao Zhang[2], and Houfeng Wang[1]

[1] Key Laboratory of Computational Linguistics, Peking University, Beijing, China
{shichen,shalei,lisujian,wanghf}@pku.edu.cn
[2] Microsoft Research Asia, Beijing, China
{cheqi,xuehui,lintaoz}@microsoft.com

Abstract. Existing task-oriented dialogue systems seldom emphasize multi-intent scenarios, which makes them hard to track complex intent switch in a multi-turn dialogue, and even harder to make proactive reactions for the user's next potential intent. In this paper, we formalize the multi-intent tracking task and introduce a complete set of intent switch modes. Then we propose ISwitch, a system that can handle complex multi-intent dialogue interactions. In this system, we design a gated controller to recognize the current intent, and a proactive mechanism to predict the next potential intent. Based on these, we use predefined patterns to generate proper responses. Experiments show that our model can achieve high intent recognition accuracy, and simplify the dialogue process. We also construct and release a new dataset for complex multi-turn multi-intent-switch dialogue.

1 Introduction

Task-oriented dialogue systems have applications in a broad variety of scenarios such as hotel reservation, airline ticket booking and customer servicing. A task-oriented dialogue system allows the users to interact with computers via natural language, which emancipates human labors from repetitive, redundant and boring tasks.

Dialogue systems are designed to satisfy the user intents. Here, intent means *a user's goal of the current utterance in a dialogue session*. In existing dialogue state tracking work [9,13,15,20,24], each dialogue session is either assumed to contain only a single (predetermined) intent, or rarely emphasized multi-intent scenario. *Multi-domain* dialogue systems [12,14,16,18,23] can handle queries from different domains, while they still consider intents are independent and process different intents separately. Since intents have finer granularity and sometimes they will interact with each other, these approaches cannot handle complex multi-intent switch situations. Since a user may switch intent during a dialogue session and greatly affect the following dialogue flow, it is crucial for a multi-turn dialogue system to recognize and track the intents of the interlocutors. Moreover, in reality, some of the user's intents are usually followed by some specific relevant intents. If the system can make a reasonable "guess" for these follow-up intents, and provide useful information before the user asks, it can save repetitive and

© Springer Nature Switzerland AG 2019
J. Tang et al. (Eds.): NLPCC 2019, LNAI 11838, pp. 93–104, 2019.
https://doi.org/10.1007/978-3-030-32233-5_8

redundant dialogue turns by borrowing information from other intents. For example, in a scheduling dialogue session, people often wish to book a meeting (1^{st} *intent*) and ask about the weather (2^{nd} *intent*), then eventually decide whether to change the schedule or not (*back to* 1^{st} *intent*). Existing dialogue systems sometimes is not able to model such interaction well, or only can simply respond to the user's questions, as shown in the *naïve response* in Fig. 1. However, when booking a meeting, the weather information is usually needed. If the system can "guess" the next intent might be *weather*, and provide *weather* information before the user asks, as shown in the *proactive response* in Fig. 1, it would improve the user experience and make the system seem "smarter".

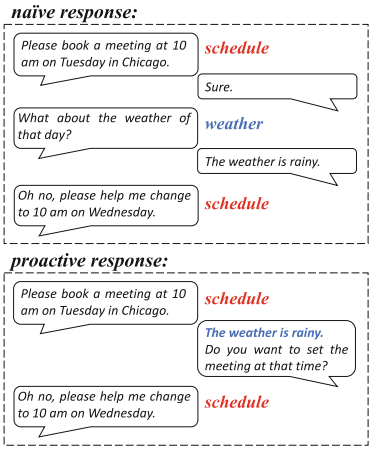

Fig. 1. Example of the *naïve* and the *proactive* responses. The proactive response predict the next intent of the user would be *weather*, and provide weather information before the user asks.

We can use two relatively straight forward approaches to adapt existing dialogue systems for multi-intent tracking. One trivial approach is to use a hierarchical recurrent encoder-decoder (HRED) framework [19] to form the sentence-level representation of each dialogue turn, and use this representation to perform the user intent classification. Since this approach does not take the slot values into consideration, it does not leverage all available information for intent tracking. The other approach is to directly use the slot information for intent tracking. [24] uses a belief tracker to help leverage all the slot value information to perform information extraction. Since their system is not designed for multi-intent interactive, they can not handle the complex intent switch scenario. Moreover, they are not able to share the overlap slot information between different intents. When facing intent switching, the system has to ask duplicated questions.

In this paper, we formalize the multi-intent tracking task, propose the ISwitch system which can handle complex multi-intent switch scenarios, including recognize current intent and predict next intent. The system is evaluated by the intent recognition accuracy and the intent switch accuracy. Experiment results show that our model can achieve high performance and simplify the dialogue process. We also release the MISD datasets for complex intent switch dialogue scenarios.

2 Model

In this paper, we treat a multi-turn dialogue as a sequence of N query-response pairs $D = \{Q_1, R_1, \cdots, Q_N, R_N\}$ between two interlocutors, in which *query* represents the user's utterance and *response* represents system's utterance. Given K potential intents and L kinds of slots, each query Q_t has an intent distribution $I_t \in \mathbb{R}^K$, which represents the interlocutor's purpose in the current utterance. The model consists of four parts, which are multi-intent tracking, proactive mechanism, information slot memory filling and response generation. In each dialogue turn t, the multi-intent tracking part leverages current query Q_t, last response R_{t-1} and current slot information $S_t \in \mathbb{R}^L$ into the a gated controller g_t to recognize the intent I_t of current turn. Then the proactive mechanism uses an intent transition matrix to predict the next potential intent I_{t+1}. If the next potential intent confidence exceeds the threshold and at least one corresponding slot of next potential intent is filled, we confirm I_{t+1} as the next intent. The slot information is obtained through sequence labeling methods, and is filled into a slot memory for global sharable. The system uses the current intent I_t and its corresponding slots to form a database query. The query results are filled into the corresponding patterns in the response generation process. The system is illustrated in detail in Fig. 2.

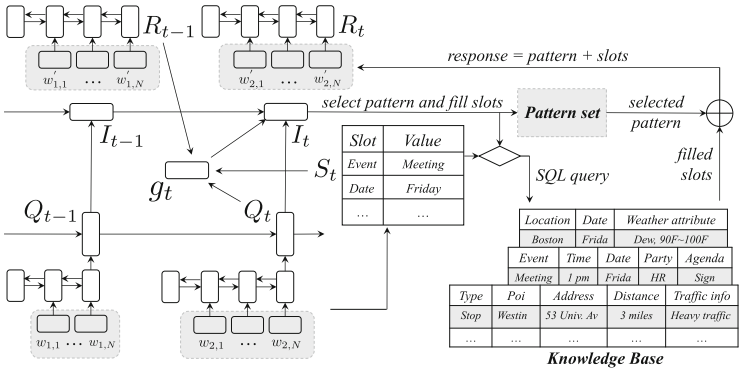

Fig. 2. An illustration of ISwitch model. Note that we first generate responses by pattern and slots. Then the R_t in the figure is calculated by reading the generated response via an LSTM.

2.1 Multi-intent Tracking

The multi-intent tracking part is the core of our ISwitch system, which can recognize the current intent. We first track the state of dialogue session in a distributed embedding representation. The dialogue session is modeled in two levels: word-level and utterance-level. We model the sequences with two RNNs: one at word-level and the other at utterance-level. The word-level RNN takes a query/response sentence as input and learns the embedding representation of it. While the utterance-level RNN takes the representation of each sentence as input, and outputs the session's states up to this turn.

For multi-intent tracking, we first distinguish three switching scenarios. We call them "modes" in the rest of the paper.

- **Mode *b* (Switch before finish)**: When the current intent is still in process, the user asks the system a question of another intent.
- **Mode *a* (Switch after finished)**: After the current intent is completed, the user starts to ask the question of another intent.
- **Mode *n* (No switch)**: The user continues to help the system solve the question of the current intent.

A brief illustration of the three intent switch modes is shown in Fig. 3.

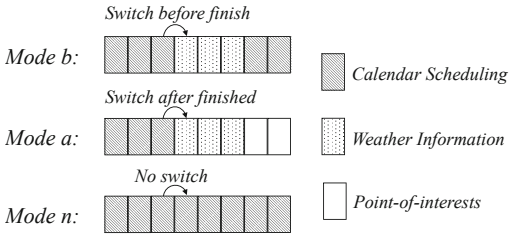

Fig. 3. An illustration of the 3 kinds of intent switch modes.

In our model, at each dialogue turn, we first decide the distribution of the modes, and use the results to form a gated switch process. More precisely, we can recognize hints of possible switch mode from the user's query in the current dialogue turn Q_t and the system's response in the previous turn R_{t-1}. In addition, the slot information S_t can also provide valuable hints for deciding intent switch mode. Therefore, we use a feed-forward layer to generate a distribution of the three modes.

$$\begin{aligned} g_t &= [g_t^b, g_t^a, g_t^n] \\ &= \text{softmax}(W_r R_{t-1} + W_q Q_t + W_s S_t) \end{aligned} \tag{1}$$

where $g_t \in \mathbb{R}^3$, W_r, W_q and W_s are trainable parameters. We leverage the softmax with temperature [10] to make the distribution "sharper".

If the system believes that I_{t-1} is going to switch without finishing (*mode b*), then I_t would be related to Q_t, R_{t-1}, S_t, and I_{t-1}. So we calculate the intent probability distribution of *mode b* as $f_b(I_{t-1}, Q_t, R_{t-1}, S_t)$, where f_b is a feed-forward layer.

If the system believes that I_{t-1} is completed (*mode a*), then we need to add a punishment to it since a user is not likely to fulfill a single task twice in one dialogue session. In this case, the distribution of I_t is calculated as $f_a(P(I_{t-1}), Q_t, R_{t-1}, S_t)$, where f_a is also a feed-forward layer. The punishment function P is implemented as follows:

$$P(I_{t-1}) = (1 - \text{softmax}(I_{t-1}))I_{t-1} \tag{2}$$

we leverage the softmax with temperature [10] to make the intent distribution "sharper".

If the system decides that the user is not switching intent (*mode n*), then the intent stays unchanged. In summary, the intent switch formula is as follows:

$$\begin{aligned} I_t = & g_t^b \cdot f_b(I_{t-1}, Q_t, R_{t-1}, S_t) \\ & + g_t^a \cdot f_a(P(I_{t-1}), Q_t, R_{t-1}, S_t) \\ & + g_t^n \cdot I_{t-1} \end{aligned} \tag{3}$$

Our training object is the cross entropy of the intents in D. Given $I_1, \cdots, I_N$, and the annotated one-hot intent vector $y_1, \cdots, y_N$, we have the loss of the intents:

$$\mathcal{L}_{\text{intent}}(D) = -\sum_{i}^{N} y_i \log I_i \tag{4}$$

2.2 Proactive Mechanism

The proactive mechanism can make a reasonable "guess" for the user's next potential intent. If the "guess" is confirmed, it will provide useful information before the user asks, to avoid repetitive dialogue turns. We use an intent transition matrix $\mathscr{T} \in \mathbb{R}^{K \times K}$ to model the switching of intents. An element $\mathscr{T}_{ij}$ is a real-valued score indicating the confidence of how likely the i-th intent will switch to the j-th intent. By using a transition matrix, we model the intent switching as a Markov chain. We use a quadratic form $I_{t-1}\mathscr{T}I_t^\top$ to represent the consistency between the intent transition matrix and the predicted probability. We link the cross entropy and the consistency function together via the predicted intents, which makes the intent switch information distilled to the intent transition matrix. Our final loss function is as follows:

$$\begin{aligned} & \mathcal{L}(D) = \mathcal{L}_{\text{intent}}(D) - \lambda_1 \sum_{t \in (1,N]} I_{t-1}\mathscr{T}I_t^\top \\ & s.t. \sum_j \mathscr{T}_{ij} = 1, \mathscr{T}_{ij} \geqslant 0, \mathscr{T}_{ii} = 0, \quad \text{for } i = 1, \cdots, K. \end{aligned} \tag{5}$$

The constraint is integrated into the loss function by the Lagrange function. Since all the components described above are differentiable, our model can be trained end-to-end by back propagation. We use Adam [11] for optimization. During inference, after I_t is predicted, we multiply it with $\mathscr{T}$ to obtain the probability distribution of the next intent: $I_{t+1} = I_t\mathscr{T}$. If the next potential intent confidence exceeds the threshold and at least one corresponding slot of next potential intent is filled, we confirm I_{t+1} as the next intent. Then, our system can react one step ahead.

2.3 Information Slot Memory Filling

For each Q_t, we need to extract the key information for the final response generation. Intuitively, the response made by the machine should be based on the information provided by Q_t, so we should record the information slots in each dialogue turn. Each dialogue session D has a slot-value list, which contains all the information slots required.

All the utterances in this session retain and updates this list during the dialogue process. Different slots are divided into informable slots and requestable slots [24]:

The informable slots are exacted information which is provided by the users to constrain the content of response. For example, as shown in Table 1, the value of an informable slot event can be extracted from the user's utterance as playing football. Then the content of response must be something related with playing football.

The requestable slots are unknown information, which are the slots the users tried to ask a value for, such as time and parties in Table 1. We also take the question words like "where" and "when" as requestable slots. The system needs to return the exact value of these slots in the next few dialogue turns.

Table 1. An example of an utterance with labels for each words. "I" and "R" means informable and requestable slots, "O" means others.

Context	I	need	the	time	and	parties	for	playing	football	please
Labels	O	O	O	R-time	O	R-party	O	I-sport	I-sport	O

We use sequence labeling methods to extract the slots. The labeling process takes an utterance as input, labels each word in the utterance as an informable slot, requestable slot, or others, and fills these slots into a global memory so that different intents can share overlapped slot values. An labeling example is shown in Table 1. The value of requestable slots cannot be directly extracted from the current utterance. The system needs to form a query for database to get the value of the requestable slots after labeling. The interaction process with database relies on several manually designed patterns, which will be introduced in detail in Sect. 2.4.

2.4 Response Generation

In each dialogue turn, after the current intent and slot-value list are ready, the system would generate a natural language response to the user by pattern and filled slots. The response of a task-oriented dialogue system requires accuracy more than diversity and fluency, which is relatively hard for language-model-based generation. Therefore, we do not use the widely-chosen sequence-to-sequence model [21] in the response generation. Instead, we use "pattern+slot" method to make responses. The generation patterns are manually built sentences with some empty slots. Certain slot values can not be directly extracted from the dialogue process, and need to be retrieved from database. After the requested information is obtained, the pattern with slot information filled will be used as the response. According to the situation of requestable slots in the query sentence, we decide whether to provide information or update the database. For each intent, we design five types of patterns. Note that we have a particular pattern (Pattern 5) for the situation that the next intent is confirmed by the proactive mechanism.

– Pattern 1: For the requestable slots, if there's only one possible result, we directly return it to the user. A simple example is shown as "Pattern 1" in Table 2.

- Pattern 2: For the requestable slots, if there are more than one possible results, we give the user all choices to choose from. In the example above, if the system found two possible restaurants, it will ask the user to choose one as is shown in the line "Pattern 2" in Table 2.
- Pattern 3: If the system cannot find any possible result for requestable slots from the database, it would ask the user to change the question. (Table 2 Pattern 3).
- Pattern 4: If the user did not provide any requestable slots, then update the database. (Table 2 Pattern 4).
- Pattern 5: (*proactive* pattern) If the next intent is confirmed by proactive mechanism, the system would provide extra useful information. (Table 2 Pattern 5).

Table 2. Examples of generated responses of five types of patterns.

Q	Where can I find a pizza restaurant?
Pattern 1	restaurant A serves delicious pizza, want to have a try?
Pattern 2	restaurant A and restaurant B both serves delicious pizza, which one would you choose?
Pattern 3	Sorry, I don't know, would you ask something else?
Q	Book a meeting at 10 am on Tuesday in Chicago office for me.
Pattern 4	OK, set up a meeting on that day.
Pattern 5	The weather of on Tuesday in Chicago is rainy with temperature 60F. Do you want to set the meeting at that time?

3 Experiments

In this section, we compare our model with the baseline systems in terms of the intent tracking metrics. We also provide the generation results in a real case for human evaluation of the proactive mechanism, and the slot labeling results.

3.1 MISD Dataset

The lack of appropriate training data is one of the main challenges for the dialogue community when building a multi-intent dialogue system. Existing well-known datasets like ATIS [6,17] and DSTC [7,8,25] are either single-turn, or not designed for multi-intent tracking. For the multi-intent switch scenario, we build a new dataset called *multi-intent switch dataset (MISD)*, which contains 6214 dialogue sessions with 22863 dialogue turns. One average, there are about 3 intent switches in a single dialogue session. The MISD dataset is based on the Stanford dataset which takes the real in-car assistant scenario, and is grounded through knowledge bases [5]. Since in the Stanford dataset, there's only one intent in each dialogue session, we manually relabel the dataset to include more complex intent switch cases that might happen in reality. We first define 14 kinds of intents, which have 48 kinds of corresponding slots. Since the intents and slots may change in one dialogue session, we manually relabel the slots and intents for each dialogue turn. We will release our MISD dataset for further research.

3.2 Baseline Systems

Since there is no existing systems especially designed for multi-intent tracking, we adapt two well-known and influential multi-turn dialogue systems as our baselines. The first one is the state-of-the-art single-intent dialogue system proposed by [24]. The second one is the most common approach for multi-turn dialogue structure (HRED) proposed by [19]. Since neither of them has the multi-intent tracking module, we extract the embedding before the generation part of those models, feed them into a classifier to detect the intent of the current utterance. Moreover, since our ISwitch system and the system proposed by [24] both leverage all the information – the query, response and slot information – in a dialogue process while the HRED system do not take them all into account, we also extend the HRED system. We feed all information above into the HRED structure, and use a general matrix to form the intent switch process, in which $I_t = W_{\mathrm{Q}} \circ Q_t + W_{\mathrm{R}} \circ R_{t-1} + W_{\mathrm{S}} \circ S_t$, where W_R, W_Q and W_S are trainable parameters.

For all the ISwitch models and baseline models, the hidden dimension of BiRNN structure is 50. All the dialogue sessions are padded to ten turns. The Adam learning rate and the dropout rate we used are 0.001 and 0.5, respectively.

3.3 Evaluation

Since the overall ISwitch system consists of multi-intent tracking, slot labeling, and proactive generation, the evaluation is also conducted on all these parts. For the multi-intent tracking, we leverage frequently-used quantitative metrics for evaluation. For the proactive generation, we provide the generation results in a real case for qualitative human evaluation. We also provide the slot labeling performance.

Table 3. The intent prediction accuracy, macro precision, recall, F-score, and intent switch accuracy for the proposed ISwitch model and other benchmark models.

Model	Accuracy	Macro-P	Macro-R	Macro-F	Switch-accuracy
Serban	91.64	93.03	83.48	88.00	90.53
Wen	92.16	92.04	85.57	88.69	91.85
Serban (Q+R+S)	92.34	94.77	85.70	90.01	91.19
ISwitch (Q)	92.60	92.32	90.07	91.18	92.17
ISwitch (R)	65.60	36.88	30.34	33.29	69.11
ISwitch (S)	91.60	93.21	88.77	90.93	91.27
ISwitch (Q+R)	93.73	93.42	90.43	91.91	93.00
ISwitch (Q+S)	93.77	94.28	90.72	92.47	93.13
ISwitch (R+S)	93.38	93.86	90.58	92.19	93.08
ISwitch (Q+R+S)	**94.27**	**94.63**	**91.18**	**92.87**	**93.57**

Muti-intent Tracking. For the multi-intent tracking evaluation, we leverage the intent recognition accuracy and the marco precision, recall and F-score as evaluation metrics. Moreover, since the gate mechanism of our proposed ISwitch system is mainly aimed at the intent switch detection, we specifically observe the switch position of the dialogue sessions, and leverage the switch accuracy to evaluate the ability of switch detection. Since our switch gate is generated according to Q, R, and S, we also conduct ablation tests on different combination of these components. The experimental results are shown in Table 3. From Table 3, we find that ISwitch(Q+R+S) can achieve better performance (acc 94.27) than all the baseline systems. Since ISwitch(Q+R+S) introduces more accurate information for intent tracking than the system modified from [19,24], we can achieve at least 2 points intent prediction accuracy improvement. For the switch accuracy, we also outperform baseline systems with about 3 points, which implies the effectiveness of our gate mechanism. From the ablation test results, we find that the information in query is more important than the information in slots, and is significantly more important than the information in last response. Since the query utterance contains almost all the key information including some slot information, it is crucial for multi-intent tracking. The slot information is also helpful for the intent tracking since some slots has obvious corresponding relationship with intents. On the contrary, the information in last response contains few useful information for current turn's intent

Proactive Response (3 turns):

Driver: Please book a meeting at 10 am on Tuesday in Chicago office for me.
Assistant: **The weather on Tuesday in Chicago is rainy with temperature 60F $\sim$ 80F.** Do you want to set the meeting at that time?
Driver: Oh no, please help me change to 10 am on Wednesday.
Assistant: **The weather on Wednesday in Chicago is sunny with temperature 80F $\sim$ 100F.** Do you want to set the meeting at that time?
Driver: Ok, thank you very much!
Assistant: You're welcome.

Non-proactive Response (6 turns):

Driver: Please book a meeting at 10 am on Tuesday in Chicago office for me.
Assistant: Sure.
Driver: What about the weather of that day?
Assistant: The weather on Tuesday in Chicago is rainy with temperature 60F $\sim$ 80F.
Driver: Oh no, please help me change to 10 am on Wednesday.
Assistant: Sure.
Driver: What about the weather on Wednesday?
Assistant: The weather on Wednesday in Chicago is sunny with temperature 80F $\sim$ 100F.
Driver: Ok, set up the meeting on that day.
Assistant: Sure.
Driver: Thank you very much!
Assistant: You're welcome.

Fig. 4. Case for response generated by the proactive mechanism and the non-proactive version. The words underlined are extracted slots by slot labeling part.

decision. We also find that if we combine components together, the result is better than we use the components separately.

Proactive Generation. The proactive mechanism provides extra useful information, which needs human to recognize, and then use to simplify the questions. So it can not be evaluated by a fixed "test set", and needs the human evaluation during the real interaction between human and the system demo. We tried 100 dialogue interactive sessions with our system, and find that each dialogue process has 2 less turns using proactive mechanism by average. We demonstrate a real example in which a user asks for booking a conference when the weather is unsuitable and then changes to another day. Figure 4 shows the proactive generated response and non-proactive version. Both responses are generated by ISwitch model, while the non-proactive version does not include the proactive module in Sect. 2.2. As shown in Fig. 4, when the driver asks to book a meeting on exact time and location, the proactive version can automatically predict the user's potential intent would be weather, and provide weather information before the user asks. This will save lots of repetitive and redundant dialogue process (3 less turns in this session). While the non-proactive can still follow the intent of the driver and provide relevant responses, but it takes more interactive turns.

Slot Labeling. As introduced in Sect. 2.3, we treat the slot information extraction for each utterance in dialogue as a sequence labeling task. For a given utterance, we first use the NLTK tokenizer and pos-tagger [1] to do the tokenization and pos tagging. Then we feed the tokenized query utterance, the pos-tag labels, chunk labels and slot labels of each word in the utterance into an open source CNN-BiRNN-CRF based sequence labeling toolkit NeuralNER[1] [3]. In the MISD dataset, we have 48 kinds of slots to label. The slot labeling accuracy, precision, recall and F-score are 94.40, 95.45, 88.11, 91.63, respectively. From the results, we can see that such kind of sequence labeling method can already provide good quality for slot information extraction.

4 Related Work

Methods. Existing task-oriented dialogue systems [9,13,15,20,26] are data-driven systems which leverage partially observable Markov Decision Process (POMDP) based dialogue managers. Recently, a relatively complete end-to-end task-oriented dialogue system is proposed by [24]. This system divided the dialogue processing procedure into four modules, which are *Intent Network*, *Belief Tracker*, *Database Operator*, and *Generation Network*. In this structure, the *Intent Network* is the same as the encoder in sequence-to-sequence framework [2,21], which encodes the input tokens, and get the representation at each dialogue turn. *Belief Tracker* (also called *Dialogue State Tracker*) is a discriminative model which tracks key information across the whole dialogue session. It is the most important component in the end-to-end task-oriented dialogue system. [9] first proposes *Belief Tracker* based on recurrent neural network, which takes advantage of the automatic speech recognition (ASR)'s output to update the belief state.

[1] https://github.com/Franck-Dernoncourt/NeuroNER.

Since then, many different belief tracking models has been proposed, such as rule-based system [8], statistical discriminative model [22]. There are also researches using latent neural embeddings for state tracking [5,15,27].

Corpora. One classical corpora for single-turn multi-intent classification task is the airline travel information system (ATIS) corpus [6,17], but it is specific for single turn dialogue. Classical corpora of multi-turn task-oriented dialogue include the well-known Dialogue State Tracking Challenge (DSTC) [7,8,25], which contains topics of bus schedule, booking restaurants and tourist information. More recently, Stanford proposed a dataset [5] which is grounded through underlying knowledge bases. Maluuba also releases a dataset [4] of hotel and travel-booking dialogues collected in Wizard-of-Oz Scheme. The limitation of the previous corpora is that they are either single-turn, or seldom emphasize the multi-intent scenario. The uniqueness of our proposed dataset is that our dataset is a multi-turn multi-intent switch dataset.

5 Conclusion

In this paper, we formalize the multi-intent tracking task and introduce a complete set of intent switch modes. Then we propose a task-oriented multi-turn dialogue system which can handle the complex multi-intent switch scenario. In this system, we design a gated controller and a proactive mechanism to track intents and guess the next potential intent, then use pre-defined patterns to generate proper responses. We evaluate our system on a multi-intent dialogue dataset made by ourselves. Experimental results show that our ISwitch system contributes to the intent recognition in terms of both intent prediction accuracy and intent switch accuracy, simplifies the dialogue process, provides high slot labeling results, and can make the generated responses more natural.

Acknowledgments. Our work is supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002101 and National Natural Science Foundation of China under GrantNo. 61433015.

References

1. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python. O'Reilly Media, Sebastopol (2009)
2. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP (2014)
3. Dernoncourt, F., Lee, J.Y., Szolovits, P.: Neuroner: an easy-to-use program for named-entity recognition based on neural networks. In: EMNLP (2017)
4. El Asri, L., et al.: Frames: a corpus for adding memory to goal-oriented dialogue systems. In: SIGdial, pp. 207–219 (2017)
5. Eric, M., Krishnan, L., Charette, F., Manning, C.D.: Key-value retrieval networks for task-oriented dialogue. In: SIGdial, pp. 37–49 (2017)
6. Hemphill, C.T., Godfrey, J.J., Doddington, G.R.: The ATIS spoken language systems pilot corpus. In: Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, 24–27 June 1990 (1990)

7. Henderson, M., Thomson, B., Williams, J.: Dialog state tracking challenge 2 and 3 (2013)
8. Henderson, M., Thomson, B., Williams, J.D.: The second dialog state tracking challenge. In: SIGDIAL, pp. 263–272 (2014)
9. Henderson, M., Thomson, B., Young, S.: Word-based dialog state tracking with recurrent neural networks. In: SIGDIAL (2014)
10. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
11. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015)
12. Lee, K., et al.: An assessment framework for dialport. In: Proceedings of the International Workshop on Spoken Dialogue Systems Technology (2017)
13. Lee, S., Eskenazi, M.: Recipe for building robust spoken dialog state trackers: dialog state tracking challenge system description. In: SIGDIAL (2013)
14. Lemon, O., Bracy, A., Gruenstein, A., Peters, S.: The WITAS multi-modal dialogue system I. In: Seventh European Conference on Speech Communication and Technology (2001)
15. Mrkšić, N., Séaghdha, D.Ó., Wen, T.H., Thomson, B., Young, S.: Neural belief tracker: data-driven dialogue state tracking. In: ACL, vol. 1, pp. 1777–1788 (2017)
16. Nakano, M., Sato, S., Komatani, K., Matsuyama, K., Funakoshi, K., Okuno, H.G.: A two-stage domain selection framework for extensible multi-domain spoken dialogue systems. In: SIGDIAL (2011)
17. Price, P.J.: Evaluation of spoken language systems: the ATIS domain. In: Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, 24–27 June 1990 (1990)
18. Rayner, M., Lewin, I., Gorrell, G., Boye, J.: Plug and play speech understanding. In: SIGdial Workshop (2001)
19. Serban, I.V., Sordoni, A., Bengio, Y., Courville, A., Pineau, J.: Building end-to-end dialogue systems using generative hierarchical neural network models. In: AAAI, pp. 3776–3783 (2016)
20. Sun, K., Chen, L., Zhu, S., Yu, K.: The SJTU system for dialog state tracking challenge 2. In: SIGDIAL (2014)
21. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS (2014)
22. Thomson, B., Young, S.: Bayesian update of dialogue state: a POMDP framework for spoken dialogue systems. Comput. Speech Lang. (2010)
23. Ultes, S., et al.: PyDial: a multi-domain statistical dialogue system toolkit. In: Proceedings of ACL 2017, System Demonstrations
24. Wen, T.H., et al.: A network-based end-to-end trainable task-oriented dialogue system. In: EACL, vol. 1, pp. 438–449 (2017)
25. Williams, J., Raux, A., Ramachandran, D., Black, A.: The dialog state tracking challenge. In: SIGDIAL, pp. 404–413 (2013)
26. Williams, J.D.: Web-style ranking and SLU combination for dialog state tracking. In: SIGDIAL (2014)
27. Williams, J.D., Asadi, K., Zweig, G.: Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In: ACL, vol. 1, pp. 665–677 (2017)