1. INTRODUCTION

Semantic frame parsing is an important research topic in spoken language understanding (SLU). The main target of semantic frame parsing in SLU is to extract meaningful slots from the query and assign them correct slot tags, i.e., slot filling. Traditionally, semantic frame parsing can be achieved by a variety of techniques, including conditional random fields (CRFs) \cite{1,2}, hidden Markov chains (HMMs) \cite{3} and support vector machines (SVMs) \cite{4}. Recent works on semantic frame parsing have sought to leverage recurrent neural network (RNN) models for sequence prediction \cite{1,5,6,7,8,9,10,11,12,13}.

Many models demonstrate decent performance on different benchmark SLU datasets, such as ATIS \cite{14} and SNIPS \cite{15}. Most of these models are only able to perform semantic frame parsing based on single-round question answering (QA) between a user and a machine. The system cannot correct the slot labels based on a user’s feedback if its first round response is wrong, let alone take extra information from a user’s response if his/her first-round query is incomplete.

Currently, people use dialogue state tracking (DST) models to handle multiturn responses in a dialogue system \cite{16,17,18,19}. These models, however, mostly focus on handling topic changing, multidomain adaption and improving goal accuracy, which are very different from the target of multiturn semantic frame parsing. The main differences between multiturn semantic frame parsing and DST are the following:

1. Normally, there is no topic change in a multiturn semantic frame parsing task in comparison to that in a DST task.
2. The system’s feedback is not necessarily the text, as it is in a DST task. More importantly, it is common that the system’s feedback is not part of our training data since we only care about the user’s responses most of the time.
3. Normally, there is no domain adaptation in a multiturn semantic frame parsing task in comparison to in a DST task.

To handle these multiturn scenarios by understanding continuous feedback from users, in this paper, we propose a human-computer Multi-Step SLU (MS2LU) model, which can learn a robust reward function through human demonstration using inverse reinforcement learning (IRL), such that multiturn semantic frame parsing (or slot filling) can be achieved.

Figure 1 demonstrates a flight booking example generated from the ATIS dataset by using our MS2LU model. The result at each step is generated by sending our extracted slot information to the flight booking API. During his/her interaction with our system, a user provides several pieces of additional information by adding a leaving/returning date and flight type in Rounds 2 and 3, respectively. In Round 4, the user even changes his/her returning date again to another date. It can be observed that our system can handle all these changes together with new information robustly, update and extract the corresponding slot tags accurately, and finally fetch the results from flight API correctly. All these features benefit from the MS2LU system introduced in this paper.

![Fig. 1: A flight booking example using the MS2LU model on the ATIS dataset](image-url)
using a multi-step learning technique to achieve multiround slot filling based on user demonstration and feedback.

2. We evaluate our model on the benchmark ATIS SLU dataset. Our system generates real flight information retrieved from Google Flight API and allows for one round of user feedback to correct if the answer is wrong during training and testing. This achieves the state-of-the-art performance on the test dataset.

3. We design and perform a multiround human flight booking task via Amazon Mechanical Turk. The MS task with one round of user feedback, and the other is a multiround flight booking task via Amazon Mechanical Turk.

2. MULTI-STEP SLU (MS2LU) SYSTEM

The MS2LU system is a reinforcement learning-based semantic frame parsing framework that can leverage the user’s response step by step to improve its system performance. The system contains several components—the feature generator, the slot extraction model, the adversarial discriminator and the reward estimator—to generate rewards using IRL. In Section 3, we conduct two experiments on the ATIS dataset: one is an SLU task with one round of user feedback, and the other is a multiround SLU flight booking task via Amazon Mechanical Turk.

2.1. Feature Generator

The feature generator extracts important semantic features from the origin query and user feedback at each round. These features are used to estimate reward $R$ and to generate policy $\pi$ and state $s_t$ in the slot extraction model.

Specifically, both the origin query and user feedback are encoded by the attention bidirectional RNN structure, as given in [21]. The encoded query feature is denoted as $c^q$, which is the final output of the query encoder $E_{query}$. Comparatively, the encoded user feedback feature $c^f_t$ is different at each time step (or round) $t$ because it only considers the user’s feedback feature generated by the feedback encoder $E_{feedback}$ in round $t$. The two feature embeddings are then concatenated together as $[c^q, c^f_t]$ to be consumed by the reward estimator and slot extraction model.

2.2. Slot Extraction Model

The slot extraction model contains two submodules: a reinforcement learning (RL) policy model and an RNN tagging model.

The RL policy model generates policy $\pi_\beta$ and the masking rules, i.e., state $s_t \in \mathbb{R}^{1 \times k}$, to identify the slot values to be maintained. Each of $s_t$’s entry is a binary value of either 0 or 1, representing whether the $i^{th}$ label $l_i$ should be presented (1) or not (0). The state update can be represented mathematically as follows:

$$s_t = f_{\pi_\beta}(g(c^q, c^f_t), s_{t-1})$$

where $f(\cdot)$ is the long short-term memory (LSTM) unit [22], and $g(\cdot)$ is a multilayer perceptron (MLP) for origin query and user feedback matching. $\pi_\beta$ is the estimated RL policy with parameter $\beta$. The update law of $\beta$ is handled by the adversarial discriminator to be discussed.

The RNN tagging model extracts the slot candidate matrix $C_{t}^{slot} \in \mathbb{R}^{k \times m}$ from the concatenated queries $[c^q, c^f_t]$. The $i^{th}$ row of $C_{t}^{slot}$ represents the average sum of the labeled token embeddings under the $i^{th}$ label $l_i$, $k$ is the total number of label types, and $m$ is the word embedding dimension. If there is no token under a label, then that label’s row is padded by zeros.

The final output of the slot extraction model is the masked slot matrix $M_t$ defined as follows:

$$M_t = diag(s_t) \cdot C_{t}^{slot}$$

where $diag(s_t) \in \mathbb{R}^{k \times k}$ is a diagonal matrix with its diagonal element $diag(s_t)(i, i) = s_t(i)$, and all other elements are zeros. Therefore, $M_t \in \mathbb{R}^{k \times m}$ is the masked slot candidate matrix by only keeping the rows where $s_t$ has nonzero values (and leaving the other rows with zeros).

2.3. Reward Estimator

The reward estimator $R_\theta(M_t)$ is a nonlinear function $\phi$, which takes the current round masked slot matrix $M_t$ and user feedback feature vector $c^f_t$ as its input.

Inspired by the reward design in different IRL applications [23] [24] [25] [26], the reward estimator function $R_\theta(M_t)$ is defined as follows:

$$R_\theta(M_t) = \phi(W_f(f(M_t) + M_t c^f_t) + b_l)$$

where $\phi$ is a nonlinear projection function, and $W_f$ and $b_l$ denote the weight and bias in the output layer, respectively. $M_t c^f_t$ stands for the projection of the feedback feature $c^f_t$ to the slot filling representation space $M_t$, and $f(\cdot)$ is an LSTM structure.

Remarks: It is worth noting that both the slot candidate matrix $C_{t}^{slot}$ and the masked matrix $M_t$ depend only on the origin query and the current-round $t$’s user feedback. The state $s_t$ contains previous-round user feedback information and hence can help filter out the unnecessary slot values and only keep the useful information.

2.4. Adversarial Discriminator

To train the nonlinear estimated reward function $R_\theta(M_t) = \phi(\cdot)$, we leverage the adversarial discriminator to associate the generated masked slot candidate matrix with the reward function. Similar to [25], we use the reward Boltzman distribution to approximate the data distribution:

$$p_{\theta}(M_t) = \frac{e^{R_\theta(M_t)}}{\sum_{M_t} e^{R_\theta(M_t)}}$$
where $M_t$ denotes the empirical sample at time step $i$. The optimal reward function $R^* (M)$ is achieved when the reward Boltzmann distribution $p_\theta (M_t)$ is equal to that in the "real" data distribution $p^* (M)$.

The objective function $J$ of the adversarial discriminator is a min-max function that maximizes $p_\theta$'s similarity with the empirical distribution of the training data $p_e$ and minimizes the similarity between them and the data generated by our slot extraction policy $\pi_\beta$, which can be mathematically represented as follows:

$$J = \arg \max_{\beta} \arg \min_{\theta} KL(p_e \parallel p_\theta) - KL(\pi_\beta \parallel p_\theta) \quad (5)$$

where $KL(\cdot)$ represents the KL divergence. Following a similar derivation in [25], our SGD learning law for the policy model network parameter $\beta$ and the reward estimator network parameter $\theta$ can be written as follows:

$$\frac{\partial J}{\partial \beta} = E_{M \sim p_e (M)} (\frac{\partial R_\theta}{\partial \theta}) - E_{M \sim \pi_\beta (M)} (\frac{\partial R_\theta}{\partial \theta})$$

$$\frac{\partial J}{\partial \beta} = E_{M \sim \pi_\beta (M)} (R_\theta (M) - \log \pi_\beta (M) - b) \times \frac{\log \pi_\beta (M)}{\partial \beta} \quad (6)$$

2.5. Slot Filling Table

As shown in Figure 2, the slot extraction model sends its output $M_t$ to the slot filling table $T$ to update the corresponding slot entries. The slot filling table at round $t$ is represented by $T_{t-1}$, containing $k$ rows, where $k$ is the number of slot types. At each round $t$, the table value $T_i$ is updated as follows:

$$T_i = T_{i-1} \cup M_t \quad (7)$$

by adding the new slot entries from $M_t$ to table $T_{i-1}$. If there already exist some values for some specific slot types in $T_{i-1}$, then we will update them correspondingly using those in $M_t$ at round $t$.

The slot values in the slot filling table are combined into a formatted query by using predefined templates and then sent to the Google Flight API to fetch the results.

3. EXPERIMENT

We conduct two experiments to demonstrate how the new system works and evaluate its performance. The first experiment is an SLU task with one round of user feedback to correct the result if the output of the origin query is wrong. The second experiment is a multiround flight booking task with help via Amazon Mechanical Turk.

3.1. Dataset

We use the ATIS dataset in both experiments and follow the train/test split in [5, 27, 1, 9], which contains 4978 utterances in the training set and 893 utterances in the test set; the total number of slot tags is 127. In the second experiment, we ask Amazon Mechanical Turks to help expand the original ATIS dataset to a multiround flight booking QA dataset. For each single query in ATIS, we ask Turks to generate 1 to 4 rounds of feedback in two categories: the first type is to ask for extra information, as in Rounds 2 and 3 in Figure 1, and the second type is to update/correct the previously stated information, as in Round 4. Round 1 uses the same feedback as that we collected for experiment 1, with one round of user feedback (to be mentioned in 3.3). The Turks are allowed to choose either type of feedback at each round by themselves (except for Round 1, which is inherited from experiment 1), and they need to note the slot tags in their feedback as ground-truth labels. The average feedback rounds include 3.2 feedback units per query.
3.2. Model Setup

For the RL policy model in the slot extraction model, we use Adam [28] as the optimizer, with an initial learning rate 10^{-5}, and we choose \( \alpha = 0.5 \) and \( \lambda = 1.0 \). The RNN structures used in the tagging model follow the same setup as in \([5]\) and \([8]\). The nonlinear function \( \phi(\cdot) \) in the reward function \( R_\theta(M_t) \) is chosen as the softsign function, i.e., \( \phi(x) = \frac{x}{1 + |x|} \). The embedding size \( k \) is set as 200.

3.3. Experiment 1: Flight booking with one round of user feedback

In the first experiment, we allow for our Turks to provide one round of correction feedback during both training and inference. We pretrain an RNN-based slot tagging model using the attention bi-RNN model given in \([5]\) and the slot-gated bi-RNN model given in \([8]\). These models are used to generate the slot tags for the origin query and the slot candidates \( C_{\text{slot}} \), as in Figure 2 and are also used as baseline models. When training the new MS2LU system, our Turks first check whether the template-based result generated by the slot extracted from the original user query can fetch the correct result from the Google Flight API. If the result is wrong, then our Turks will provide feedback to correct the mistake specifically. For example, if we want to query a destination “Cincinnati”, but the result displays the incorrect destination of “Philadelphia”, then a Turk should reply as follows: “I want to go to Cincinnati actually”. Similarly, we also allow Turks to have at most one round of interaction during inference for correction purposes. Table 1 shows a comparison of the results by using the baseline models and those with our MS2LU system. We can observe that our MS2LU system with one round of feedback can improve slot F1 by more than 2.5% on both RNN tagging model structures.

Table 1: Experiment 1: flight booking with one-round user feedback

| Model | Slot F1 % | Sentence Acc % |
|-------|-----------|----------------|
| Attention bi-RNN | 94.2 | 78.9 |
| Attention bi-RNN + MS2LU | 96.8 | 83.6 |
| Slot-gated bi-RNN | 95.2 | 82.6 |
| Slot-gated bi-RNN+ MS2LU | 97.7 | 85.7 |

3.4. Experiment 2: Multiround flight booking task

The second experiment is a multiround flight booking task using the expanded ATIS dataset, as described in the data section. Again, we compare the results by using the baseline RNN models and those with our MS2LU system. In each round, the input to the attention RNN model is the concatenation of the raw user query and the user’s feedback at round \( t \). In Table 2 we show the test result by using the slot F1 scores at each round.

Based on the experimental results, we can observe that our MS2LU model performs better than the baseline RNN models in all rounds. The advantage gaps of the MS2LU model over baselines become larger when the number of feedback rounds increases. One main reason is that the new system is able to remember user feedback history better with an RL structure and hence can make a better decision as to whether to keep, remove or update the slot values.

Furthermore, we also compare the best performing slot-gated bi-RNN+MS2LU model with two other state-of-the-art DST models, MA-DST [16] and TRADE [17], to test how these DST systems perform in our multiround SLU task. The results are shown in Table 3.

Table 2: Experiment 2.1: Comparison of MS2LU models on a multiround flight booking task

| Model/slot F1(%) | Round 1 | Round 2 | Round 3 | Round 4 |
|------------------|---------|---------|---------|---------|
| Attention bi-RNN | 94.2    | 93.1    | 92.7    | 89.2    |
| Attention bi-RNN + MS2LU | 96.8 | 95.3 | 95.8 | 96.3 |
| Slot-gated bi-RNN | 95.2 | 93.4 | 91.8 | 90.1 |
| Slot-gated bi-RNN+ MS2LU | 97.7 | 96.5 | 96.1 | 97.1 |

Table 3: Experiment 2.2: Comparison between the MS2LU model and DST models on a multiround flight booking task

| Model/slot F1(%) | Round 1 | Round 2 | Round 3 | Round 4 |
|------------------|---------|---------|---------|---------|
| TRADE | 83.6 | 82.2 | 82.5 | 79.8 |
| MA-DST | 85.5 | 83.2 | 81.6 | 80.3 |
| Slot-gated bi-RNN+ MS2LU | 97.7 | 96.5 | 96.1 | 97.1 |

It can be observed that even the state-of-the-art DST models do not perform very well on the multiround SLU task, the main reasons for which are as follows:
1. We do not have any text in the system’s response during training, but most of the DST systems require system response texts as one of their inputs.
2. Furthermore, there is no domain changing or topic changing in our multiround SLU problem, so the DST models cannot exhibit their advantages in handling the domain/topic changing scenarios as in other DST tasks.

4. CONCLUSIONS

In this paper, we introduce a novel multi-step SLU system that can leverage the multiround user’s feedback to update slot values in a semantic frame parsing task. We test our model with two experiments on the ATIS dataset by using single-round and multiround feedback from users. By comparing with baseline tagging models, we show that our MS2LU system can greatly improve the tagging model’s performance by leveraging user feedback, and the advantage becomes greater when the number of feedback rounds increases.

5. REFERENCES

[1] P. Xu and R. Sarikaya, “Convolutional neural network based triangular crf for joint intent detection and slot filling,” in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 78–83.
[2] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao, “Recurrent conditional random field for language un-
understanding,” in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014, pp. 4077–4081.

[3] Y. He and S. Young, “Hidden vector state model for hierarchical semantic parsing,” in 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP’03)., vol. 1. IEEE, 2003, pp. I–I.

[4] C. Raymond and G. Riccardi, “Generative and discriminative algorithms for spoken language understanding,” in Eighth Annual Conference of the International Speech Communication Association, 2007.

[5] B. Liu and I. Lane, “Attention-based recurrent neural network models for joint intent detection and slot filling,” Interspeech 2016, pp. 685–689, 2016.

[6] Y. Wang, A. Patel, and H. Jin, “A new concept of deep reinforcement learning based augmented general tagging system,” in Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 1683–1693.

[7] Y. Wang, A. Patel, Y. Shen, and H. Jin, “A deep reinforcement learning based multimodal coaching model (dcm) for slot filling in spoken language understanding (slu),” Proc. Interspeech 2018, pp. 3444–3448, 2018.

[8] C.-W. Goo, G. Gao, Y.-K. Hsu, C.-L. Huo, T.-C. Chen, K.-W. Hsu, and Y.-N. Chen, “Slot-gated modeling for joint slot filling and intent prediction,” in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), 2018, pp. 753–757.

[9] Y. Wang, Y. Shen, and H. Jin, “A bi-model based rnn semantic frame parsing model for intent detection and slot filling,” in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), 2018, pp. 309–314.

[10] Y. Wang, Y. Shen, Y. Deng, and H. Jin, “Multi-model structures for classification and intent determination,” Oct. 22 2020, uS Patent App. 16/728,987.

[11] Q. Chen, Z. Zhuo, and W. Wang, “Bert for joint intent classification and slot filling,” arXiv preprint arXiv:1902.10909, 2019.

[12] C. Zhang, Y. Li, N. Du, W. Fan, and P. S. Yu, “Joint slot filling and intent detection via capsule neural networks,” arXiv preprint arXiv:1812.09471, 2018.

[13] Y. Wang, Y. Shen, Y. Deng, and H. Jin, “A new concept of multiple neural networks structure using convex combination,” IEEE transactions on neural networks and learning systems, vol. 31, no. 11, pp. 4968–4979, 2020.

[14] P. I. Price, “Evaluation of spoken language systems: The atis domain,” in Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990, 1990.

[15] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril et al., “Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces,” arXiv preprint arXiv:1805.10190, 2018.

[16] A. Kumar, P. Ku, A. K. Goyal, A. Metallinou, and D. Hakkani-Tur, “Ma-dst: Multi-attention based scalable dialog state tracking,” arXiv preprint arXiv:2002.08898, 2020.

[17] C.-S. Wu, A. Madotto, E. Hosseini-Asl, C. Xiong, R. Socher, and P. Fung, “Transferable multi-domain state generator for task-oriented dialogue systems,” in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 808–819.

[18] Y. Wang and H. Jin, “A deep reinforcement learning based multi-step coarse to fine question answering (mscqa) system,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, 2019, pp. 7224–7232.

[19] Y. Wang, Y. Shen, and H. Jin, “An interactive adversarial reward learning-based spoken language understanding system,” Proc. Interspeech 2020, pp. 1565–1569, 2020.

[20] Y. Wang and H. Jin, “A boosting-based deep neural networks algorithm for reinforcement learning,” in 2018 Annual American Control Conference (ACC). IEEE, 2018, pp. 1065–1071.

[21] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” arXiv preprint arXiv:1409.0473, 2014.

[22] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] L. Li, W. Chu, J. Langford, and R. E. Schapire, “A contextual-bandit approach to personalized news article recommendation,” in Proceedings of the 19th international conference on World wide web. ACM, 2010, pp. 661–670.

[24] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragani, “Cooperative inverse reinforcement learning,” in Advances in neural information processing systems, 2016, pp. 3909–3917.

[25] X. Wang, W. Chen, Y.-F. Wang, and W. Y. Wang, “No metrics are perfect: Adversarial reward learning for visual storytelling,” in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018, pp. 899–909.

[26] Z. Li, J. Kiseleva, and M. de Rijke, “Dialogue generation: From imitation learning to inverse reinforcement learning,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 6722–6729.

[27] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu et al., “Using recurrent neural networks for slot filling in spoken language understanding,” IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), vol. 23, no. 5, pp. 530–539, 2015.

[28] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” ICLR 2014, 2014.