

Modeling Multi-Action Policy for Task-Oriented Dialogues

Lei Shu, Hu Xu, Bing Liu
University of Illinois at Chicago
{lshu3, hxu48, liub}@uic.edu

Piero Molino
Uber AI
piero@uber.com

Abstract

Dialogue management (DM) plays a key role in the quality of the interaction with the user in a task-oriented dialogue system. In most existing approaches, the agent predicts only one DM policy action per turn. This significantly limits the expressive power of the conversational agent and introduces unwanted turns of interactions that may challenge users' patience. Longer conversations also lead to more errors and the system needs to be more robust to handle them. In this paper, we compare the performance of several models on the task of predicting multiple acts for each turn. A novel policy model is proposed based on a recurrent cell called gated Continue-Act-Slots (gCAS) that overcomes the limitations of the existing models. Experimental results show that gCAS¹ outperforms other approaches.²

1 Introduction

In a task-oriented dialogue system, the dialogue manager policy module predicts actions usually in terms of dialogue acts and domain specific slots. It is a crucial component that influences the efficiency (e.g., the conciseness and smoothness) of the communication between the user and the agent. Both supervised learning (SL) (Stent, 2002; Williams et al., 2017a; Williams and Zweig, 2016; Henderson et al., 2005, 2008) and reinforcement learning (RL) approaches (Walker, 2000; Young et al., 2007; Gasic and Young, 2014; Williams et al., 2017b; Su et al., 2017) have been adopted

user msg	Hi! I'm looking for good thriller. Are there any playing right now?
agent msg	Yes, there are! The Witch, The Other Side of the Door, and The Boy are all thrillers. Would you like to find tickets for a showing for any of them?
agent acts	inform(moviename=The Witch, The Other Side of the Door, The Boy; genre=thriller) multiple_choice(moviename)

Table 1: Dialogue example.

to learn policies. SL learns a policy to predict acts given the dialogue state. Recent work (Wen et al., 2017; Liu and Lane, 2018) also used SL as pre-training for RL to mitigate the sample inefficiency of RL approaches and to reduce the number of interactions. Sequence2Sequence (Seq2Seq) (Sutskever et al., 2014) approaches have also been adopted in user simulators to produce user acts (Gur et al., 2018). These approaches typically assume that the agent can only produce one act per turn through classification. Generating only one act per turn significantly limits what an agent can do in a turn and leads to lengthy dialogues, making tracking of state and context throughout the dialogue harder. An example in Table 1 shows how the agent can produce both an *inform* and a *multiple_choice* act, reducing the need for additional turns.

The use of multiple actions has previously been used in interaction managers that keep track of the floor (who is speaking right now) (Raux and Eskénazi, 2007; Khouzaimi et al., 2015; Hastie et al., 2013, among others), but the option of generating multiple acts simultaneously at each turn for dialogue policy has been largely ignored, and only explored in simulated scenarios without real data (Chandramohan and Pietquin, 2010).

This task can be cast as a multi-label classification problem (if the sequential dependency among the acts is ignored) or as a sequence generation one as shown in Table 2.

In this paper, we introduce a novel policy model to output multiple actions per turn (called multi-act), generating a sequence of tuples and expand-

¹The code is available at <https://leishu02.github.io/>

²To appear in EMNLP 2019

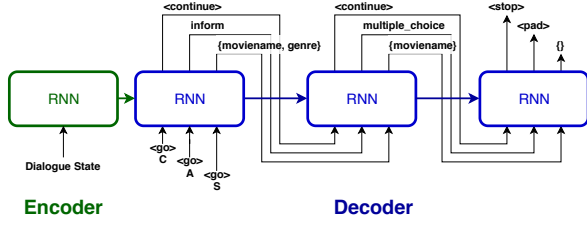


Figure 1: CAS decoder: at each step, a tuple of (continue, act, slots) is produced. The KB vector k regarding the queried result from knowledge base is not shown for brevity.

ing agents’ expressive power. Each tuple is defined as $(continue, act, slots)$, where *continue* indicates whether to continue or stop producing new acts, *act* is an *act type* (e.g., *inform* or *request*), and *slots* is a set of slots (names) associated with the current *act type*. Correspondingly, a novel decoder (Figure 1) is proposed to produce such sequences. Each tuple is generated by a cell called gated Continue Act Slots (gCAS, as in Figure 2), which is composed of three sequentially connected gated units handling the three components of the tuple. This decoder can generate multi-acts in a double recurrent manner (Tay et al., 2018). We compare this model with baseline classifiers and sequence generation models and show that it consistently outperforms them.

2 Methodology

The proposed policy network adopts an encoder-decoder architecture (Figure 1). The input to the encoder is the current-turn dialogue state, which follows Li et al. (2018)’s definition. It contains policy actions from the previous turn, user dialogue acts from the current turn, user requested slots, the user informed slots, the agent requested slots and agent proposed slots. We treat the dialogue state as a sequence and adopt a GRU (Cho et al., 2014) to encode it. The encoded dialogue state is a sequence of vectors $\mathbf{E} = (e_0, \dots, e_l)$ and the last hidden state is h^E . The CAS decoder recurrently generates tuples at each step. It takes h^E as initial hidden state h_0 . At each decoding step, the input contains the previous (continue, act, slots) tuple $(c_{t-1}, a_{t-1}, s_{t-1})$. An additional vector k containing the number of results from the knowledge base (KB) query and the current turn number is given as input. The output of the decoder at each step is a tuple (c, a, s) , where $c \in \{\langle \text{continue} \rangle, \langle \text{stop} \rangle, \langle \text{pad} \rangle\}$, $a \in A$ (one act

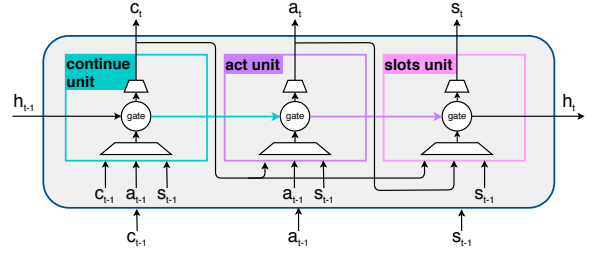


Figure 2: The gated CAS recurrent cell contains three units: continue unit, act unit and slots unit. The three units use a gating mechanism and are sequentially connected. The KB vector k is not shown for brevity.

from the act set), and $s \subset S$ (a subset from the slot set).

2.1 gCAS Cell

As shown in Figure 2, the gated CAS cell contains three sequentially connected units for outputting continue, act, and slots respectively.

The **Continue unit** maps the previous tuple $(c_{t-1}, a_{t-1}, s_{t-1})$ and the KB vector k into x_t^c . The hidden state from the previous step h_{t-1} and x_t^c are inputs to a GRU^c unit that produces output g_t^c and hidden state h_t^c . Finally, g_t^c is used to predict c_t through a linear projection and a softmax.

$$\begin{aligned} x_t^c &= W_x^c[c_{t-1}, a_{t-1}, s_{t-1}, k] + b_x^c, \\ g_t^c, h_t^c &= \text{GRU}^c(x_t^c, h_{t-1}), \\ P(c_t) &= \text{softmax}(W_g^c g_t^c + b_g^c), \\ \mathcal{L}^c &= - \sum_t \log P(c_t). \end{aligned} \quad (1)$$

The **Act unit** maps the tuple (c_t, a_{t-1}, s_{t-1}) and the KB vector k into x_t^a . The hidden state from the continue cell h_t^c and x_t^a are inputs to a GRU^a unit that produces output g_t^a and hidden state h_t^a . Finally, g_t^a is used to predict a_t through a linear projection and a softmax.

$$\begin{aligned} x_t^a &= W_x^a[c_t, a_{t-1}, s_{t-1}, k] + b_x^a, \\ g_t^a, h_t^a &= \text{GRU}^a(x_t^a, h_t^c), \\ P(a_t) &= \text{softmax}(W_g^a g_t^a + b_g^a), \\ \mathcal{L}^a &= - \sum_t \log P(a_t). \end{aligned} \quad (2)$$

The **Slots unit** maps the tuple (c_t, a_t, s_{t-1}) and the KB vector k into x_t^s . The hidden state from the act cell h_t^a and x_t^s are inputs to a GRU^s unit that produces output g_t^s and hidden state h_t^s . Finally, g_t^s is used to predict s_t through a linear projection and

annotation	inform(moviename=The Witch, The Other Side of the Door, The Boy; genre=thriller) multiple_choice(moviename)
classification	inform+moviename, inform+genre, multiple_choice+moviename
sequence	'inform' '(' 'moviename' '=' ';' 'genre' '=' ') 'multiple_choice' '(' 'moviename' ')' 'eos'
cas sequence	((continue), inform, {moviename, genre}) ((continue), multiple_choice, {moviename}) ((stop), {pad}, {})

Table 2: Multiple dialogue act format in different architectures.

domain	total	train	valid	test	acts	slots	pairs
movie	2888	1445	433	1010	11	29	90
taxi	3093	1548	463	1082	11	23	63
restaurant	4101	2051	615	1435	11	31	91

Table 3: Dataset: train, validation and test split, and the count of distinct acts, slots and act-slot pairs.

domain & speaker	1 act	2 acts	3 acts	4 acts
movie user	9130	1275	106	11
movie agent	5078	4982	427	33
taxi user	10544	762	50	8
taxi agent	7855	3301	200	8
restaurant user	12726	1672	100	3
restaurant agent	10333	3755	403	10

Table 4: Dialogue act counts by turn.

a sigmoid. Let z_t^i be the i -th slot’s ground truth.

$$\begin{aligned}
x_t^s &= W_x^s [c_t, a_t, s_{t-1}, k] + b_x^s, \\
g_t^s, h_t^s &= \text{GRU}^s(x_t^s, h_t^a), \\
s_t &= \text{sigmoid}(W_g^s g_t^s + b_g^s), \\
\mathcal{L}^s &= - \sum_t \sum_{i=0}^{|S|} z_t^i \log s_t^i + (1 - z_t^i) \log (1 - s_t^i).
\end{aligned} \tag{3}$$

The overall loss is the sum of the losses of the three units: $\mathcal{L} = \mathcal{L}^c + \mathcal{L}^a + \mathcal{L}^s$

3 Experiments

The experiment dataset comes from Microsoft Research (MSR)³. It contains three domains: movie, taxi, and restaurant. The total count of dialogues per domain and train/valid/test split is reported in Table 3. At every turn both user and agent acts are annotated, we use only the agent side as targets in our experiment. The acts are ordered in the dataset (each output sentence aligns with one act). The size of the sets of acts, slots, and act-slot pairs are also listed in Table 3. Table 4 shows the count of turns with multiple act annotations, which amounts to 23% of the dataset. We use MSR’s dialogue management code and knowledge base to obtain the state at each turn and use it as input to every model.

³https://github.com/xiul-msr/e2e_dialog_challenge

	Entity F ₁			Success F ₁		
	movie	taxi	restaurant	movie	taxi	restaurant
Classification	34.02	49.71	28.23	70.41	84.45	39.97
Seq2Seq	39.95	63.12	60.21	77.82	75.09	55.70
Copy Seq2Seq	28.04	62.95	59.14	77.59	74.58	58.74
CAS	48.02	59.16	54.70	76.81	78.89	65.18
gCAS	50.86	64.00	60.35	77.95	81.17	71.52

Table 5: Entity F₁ and Success F₁ at dialogue level.

3.1 Evaluation Metrics

We evaluate the performance at the act, frame and task completion level. For a frame to be correct, both the act and all the slots should match the ground truth. We report precision, recall, F₁ score of turn-level acts and frames. For task completion evaluation, **Entity F₁** score and **Success F₁** score (Lei et al., 2018) are reported. The Entity F₁ score, differently from the entity match rate in state tracking, compares the slots requested by the agent with the slots the user informed about and that were used to perform the KB query. We use it to measure agent performance in requesting information. The Success F₁ score compares the slots provided by the agent with the slots requested by the user. We use it to measure the agent performance in providing information.

Critical slots and Non-critical slots: By ‘non-critical’, we mean slots that the user informs the system about by providing their values and thus it is not critical for the system to provide them in the output. Table 1 shows an example, with the genre slot provided by the user and the system repeating it in its answer. Critical slots refers to slots that the system must provide like moviename in the Table 1 example. Although non-critical slots do not impact task completion directly, they may influence the output quality by enriching the dialogue state and helping users understand the system’s utterance correctly. Furthermore, given the same dialog state, utterances offering non-critical slots or not offering them can both be present in the dataset, as they are optional. This makes the prediction of those slots more challenging for the system. To provide a more detailed analysis, we report the precision, recall, F₁ score of turn-level

method	Act									Frame								
	movie			taxi			restaurant			movie			taxi			restaurant		
	\mathcal{P}	\mathcal{R}	\mathcal{F}_1	\mathcal{P}	\mathcal{R}	\mathcal{F}_1	\mathcal{P}	\mathcal{R}	\mathcal{F}_1	\mathcal{P}	\mathcal{R}	\mathcal{F}_1	\mathcal{P}	\mathcal{R}	\mathcal{F}_1	\mathcal{P}	\mathcal{R}	\mathcal{F}_1
classification	84.19	50.24	62.93	92.20	55.48	69.27	79.71	33.94	47.60	63.91	18.39	28.56	65.87	44.31	52.98	49.63	12.32	19.74
Seq2Seq	73.44	73.62	73.53	77.52	69.29	73.17	65.66	66.01	65.83	42.88	24.81	31.43	57.12	50.32	53.51	39.97	25.40	31.06
Copy Seq2Seq	67.56	73.61	70.46	73.99	69.21	71.52	64.93	65.69	65.31	41.90	23.12	29.80	51.66	50.23	50.93	36.96	27.22	31.35
CAS	70.46	76.08	73.16	79.85	72.54	76.02	65.40	72.43	68.73	43.12	31.60	36.47	51.66	54.29	52.94	33.72	25.45	29.01
gCAS	73.08	75.78	74.41	79.47	75.39	77.37	68.30	74.39	71.22	42.24	35.50	38.58	53.77	56.24	54.98	36.86	32.41	34.49

Table 6: Precision (\mathcal{P}), Recall (\mathcal{R}) and \mathcal{F}_1 score (\mathcal{F}_1) of turn-level acts and frames.

	example 1	example 2
groundtruth	request(date; starttime)	inform(restaurantname=; starttime =) multiple_choice(restaurantname)
classification	request+date	[]
Seq2Seq	'request' '(' 'date' ';' 'starttime' ')'	'inform' '(' 'restaurantname' '=' ')' 'multiple_choice' '=' 'restaurantname' ')'
Copy Seq2Seq	'request' '(' 'date' '=' ')'	'inform' '(' 'restaurantname' '=' ';' ';' ';' '=' ';' 'starttime' '=' ')'
CAS	request {}	inform {restaurantname}
gCAS	request {date; starttime}	inform {restaurantname} multiple_choice {restaurantname}

Table 7: Examples of predicted dialogue acts in the restaurant domain.

for all slots, critical slots and non-critical slots of the *inform* act.

3.2 Baseline

We compare five methods on the multi-act task.

Classification replicates the MSR challenge (Li et al., 2018) policy network architecture: two fully connected layers. We replace the last activation from softmax to sigmoid in order to predict probabilities for each act-slot pair. It is equivalent to binary classification for each act-slot pair and the loss is the sum of the binary cross-entropy of all of them.

Seq2Seq (Sutskever et al., 2014) encodes the dialogue state as a sequence, and decodes agent acts as a sequence with attention (Bahdanau et al., 2015).

Copy Seq2Seq (Gu et al., 2016) adds a copy mechanism to Seq2Seq, which allows copying words from the encoder input.

CAS adopts a single GRU (Cho et al., 2014) for decoding and uses three different fully connected layers for mapping the output of the GRU to continue, act and slots. For each step in the sequence of CAS tuples, given the output of the GRU, continue, act and slot predictions are obtained by separate heads, each with one fully connected layer. The hidden state of the GRU and the predictions at the previous step are passed to the cell at the next step connecting them sequentially.

gCAS uses our proposed recurrent cell which contains separate continue, act and slots unit that are sequentially connected.

The classification architecture has two fully connected layers of size 128, and the remaining models have a hidden size of 64 and a teacher-

forcing rate of 0.5. Seq2Seq and Copy Seq2Seq use a beam search with beam size 10 during inference. CAS and gCAS do not adopt a beam search since their inference steps are much less than Seq2Seq methods. All models use Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001.

3.3 Result and Error Analysis

As shown in Table 5, gCAS outperforms all other methods on Entity \mathcal{F}_1 in all three domains. Compared to Seq2Seq, the performance advantage of gCAS in the taxi and restaurant domains is small, while it is more evident in the movie domain. The reason is that in the movie domain the proportion of turns with multiple acts is higher (52%), while in the other two domains it is lower (30%). gCAS also outperforms all other models in terms of Success \mathcal{F}_1 in the movie and restaurant domain but is outperformed by the classification model in the taxi domain. The reason is that in the taxi domain, the agent usually informs the user at the last turn, while in all previous turns the agent usually requests information from the user. It is easy for the classification model to overfit this pattern. The advantage of gCAS in the restaurant domain is much more evident: the agent’s *inform* act usually has multiple slots (see example 2 in Table 7) and this makes classification and sequence generation harder, but gCAS multi-label slots decoder handles it easily.

Table 6 shows the turn-level acts and frame prediction performance. CAS and gCAS outperform all other models in acts prediction in terms of \mathcal{F}_1 score. The main reason is that CAS and gCAS output a tuple at each recurrent step, which makes for

method	All Slots									Non-critical slots								
	movie			taxi			restaurant			movie			taxi			restaurant		
	\mathcal{P}	\mathcal{R}	\mathcal{F}_1	\mathcal{P}	\mathcal{R}	\mathcal{F}_1	\mathcal{P}	\mathcal{R}	\mathcal{F}_1	\mathcal{P}	\mathcal{R}	\mathcal{F}_1	\mathcal{P}	\mathcal{R}	\mathcal{F}_1	\mathcal{P}	\mathcal{R}	\mathcal{F}_1
classification	67.90	21.48	32.64	73.52	72.66	73.08	45.16	12.71	19.84	62.98	13.39	22.08	43.91	60.03	50.72	33.61	11.15	16.75
Seq2Seq	53.25	29.54	38.00	64.09	74.32	68.83	42.36	17.73	25.00	47.90	13.95	21.61	64.15	48.45	55.20	35.28	12.95	18.94
Copy Seq2Seq	52.78	28.43	36.95	56.92	74.06	64.37	38.15	22.38	28.21	40.45	12.48	19.07	45.95	55.46	50.26	34.90	19.11	24.70
CAS	63.61	33.16	43.59	61.90	80.39	69.94	51.12	22.57	31.31	56.21	26.96	36.44	43.03	68.03	52.72	37.31	15.87	22.27
gCAS	54.75	38.70	45.35	62.31	79.76	69.96	44.20	29.65	35.49	48.23	36.68	41.67	44.35	62.15	51.77	31.26	29.60	30.41

Table 8: \mathcal{P} , \mathcal{R} and \mathcal{F}_1 of turn-level *inform* all slots and non-critical slots.

method	Critical Slots								
	movie			taxi			restaurant		
	\mathcal{P}	\mathcal{R}	\mathcal{F}_1	\mathcal{P}	\mathcal{R}	\mathcal{F}_1	\mathcal{P}	\mathcal{R}	\mathcal{F}_1
Classification	70.29	29.13	41.19	85.18	75.90	80.27	55.66	13.76	22.07
Seq2Seq	55.08	44.26	49.08	64.08	80.97	71.54	46.24	20.97	28.86
Copy Seq2Seq	57.54	43.49	49.54	59.49	78.83	67.81	40.11	24.59	30.49
CAS	69.59	39.02	50.00	68.15	83.57	75.08	59.93	27.10	37.32
gCAS	61.89	40.62	49.04	67.48	84.28	74.95	61.35	29.69	40.01

Table 9: \mathcal{P} , \mathcal{R} and \mathcal{F}_1 of turn-level *inform* critical slots.

shorter sequences that are easier to generate compared to the long sequences of Seq2Seq (example 2 in Table 7). The classification method has a good precision score, but a lower recall score, suggesting it has problems making granular decisions (example 2 in Table 7). At the frame level, gCAS still outperforms all other methods. The performance difference between CAS and gCAS on frames becomes much more evident, suggesting that gCAS is more capable of predicting slots that are consistent with the act. This finding is also consistent with their Entity \mathcal{F}_1 and Success \mathcal{F}_1 performance.

However, gCAS’s act-slot pair performance is far from perfect. The most common failure case is on non-critical slots (like ‘genre’ in the example in Table 2): gCAS does not predict them, while it predicts the critical ones (like ‘moviename’ in the example in Table 2).

Table 7 shows predictions of all methods from two emblematic examples. Example 1 is a frequent single-act multi-slots agent act. Example 2 is a complex multi-act example. The baseline classification method can predict frequent pairs in the dataset, but cannot predict any act in the complex example. The generated sequences of Copy Seq2Seq and Seq2Seq show that both models struggle in following the syntax. CAS cannot predict slots correctly even if the act is common in the dataset. gCAS returns a correct prediction for Example 1, but for Example 2 gCAS cannot predict ‘starttime’, which is a non-critical slot.

Tables 8 and 9 show the results of all slots, critical slots and non-critical slots under the *inform* act. gCAS performs better than the other methods on all slots in the movie and restaurant domains. The

reason why classification performs the best here in the taxi domain is the same as the Success \mathcal{F}_1 . In the taxi domain, the agent usually informs the user at the last turn. The non-critical slots are also repeated frequently in the taxi domain, which makes their prediction easier. gCAS’s performance is close to other methods on critical-slots. The reason is that the *inform* act is mostly the first act in multi-act and critical slots are usually frequent in the data. All methods can predict them well.

In the movie and restaurant domains, the *inform* act usually appears during the dialogue and there are many optional non-critical slots that can appear (see Table 3, movie and restaurant domains have more slots and pairs than the taxi domain). gCAS can better predict the non-critical slots than other methods. However, the overall performance on non-critical slots is much worse than critical slots since their appearances are optional and inconsistent in the data.

4 Conclusion and Future Work

In this paper, we introduced a multi-act dialogue policy model motivated by the need for a richer interaction between users and conversation agents. We studied classification and sequence generation methods for this task, and proposed a novel recurrent cell, gated CAS, which allows the decoder to output a tuple at each step. Experimental results showed that gCAS is the best performing model for multi-act prediction. The CAS decoder and the gCAS cell can also be used in a user simulator and gCAS can be applied in the encoder. A few directions for improvement have also been identified: 1) improving the performance on non-critical slots, 2) tuning the decoder with RL, 3) text generation from gCAS. We leave them as future work.

Acknowledgments

We would like to express our special thanks to Alexandros Papangelis and Gokhan Tur for their support and contribution. We also would like to thank Xiujun Li for his help on dataset preparation

and Jane Hung for her valuable comments. Bing Liu is partially supported by the NSF grant IIS-1910424 and a research gift from Northrop Grumman.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations, San Diego, California, USA*.
- Senthilkumar Chandramohan and Olivier Pietquin. 2010. User and noise adaptive dialogue management using hybrid system actions. In *Spoken Dialogue Systems for Ambient Environments*, Springer, pages 13–24.
- Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP. ACL*, pages 1724–1734.
- Milica Gasic and Steve J. Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22:28–40.
- Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL (1)*. The Association for Computer Linguistics.
- Izzeddin Gur, Dilek Z. Hakkani-Tur, Gokhan Tur, and Pararth Shah. 2018. User modeling for task oriented dialogues. *2018 IEEE Spoken Language Technology Workshop (SLT)* pages 900–906.
- Helen F. Hastie, Marie-Aude Aufaure, Panos Alexopoulos, Heriberto Cuayáhuitl, Nina Dethlefs, Milica Gasic, James Henderson, Oliver Lemon, Xingkun Liu, Peter Mika, Nesrine Ben Mustapha, Verena Rieser, Blaise Thomson, Pirros Tsiakoulis, and Yves Vanrompay. 2013. Demonstration of the parlance system: a data-driven incremental, spoken dialogue system for interactive search. In *SIGDIAL Conference*.
- James Henderson, Oliver Lemon, and Kallirroi Georgila. 2005. Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In *IJCAI workshop on knowledge and reasoning in practical dialogue systems*. Citeseer, pages 68–75.
- James Henderson, Oliver Lemon, and Kallirroi Georgila. 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics* 34(4):487–511.
- Hatim Khouzaimi, Romain Laroche, and Fabrice Lefèvre. 2015. Turn-taking phenomena in incremental dialogue systems. In *EMNLP*.
- Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, San Diego, California, USA*.
- Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *ACL*.
- Xiujun Li, Sarah Panda, Jingjing Liu, and Jianfeng Gao. 2018. Microsoft dialogue challenge: Building end-to-end task-completion dialogue systems. volume abs/1807.11125.
- Bing Liu and Ian Lane. 2018. End-to-end learning of task-oriented dialogs. In *Proceedings of the NAACL-HLT*.
- Antoine Raux and Maxine Eskénazi. 2007. A multi-layer architecture for semi-synchronous event-driven dialogue management. *2007 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* pages 514–519.
- Amanda J Stent. 2002. A conversation acts model for generating spoken dialogue contributions. *Computer Speech & Language* 16(3-4):313–352.
- Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve J. Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. In Kristiina Jokinen, Manfred Stede, David DeVault, and Annie Louis, editors, *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*. Association for Computational Linguistics, pages 147–157.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*. pages 3104–3112.
- Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Recurrently controlled recurrent networks. In *Advances in Neural Information Processing Systems*. pages 4736–4748.
- Marilyn A. Walker. 2000. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *J. Artif. Intell. Res.* 12:387–416.
- Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve J. Young. 2017. Latent intention dialogue models. In *ICML. PMLR*, volume 70 of *Proceedings of Machine Learning Research*, pages 3732–3741.

- Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017a. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *ACL (1)*. Association for Computational Linguistics, pages 665–677.
- Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017b. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *ACL*.
- Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*.
- Steve J. Young, Jost Schatzmann, Karl Weilhammer, and Hui Ye. 2007. The hidden information state approach to dialog management. *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07* 4:IV–149–IV–152.