A Task-Oriented Dialogue Architecture via Transformer Neural Language Models and Symbolic Injection

Oscar J. Romero, Antian Wang, John Zimmerman, Aaron Steinfeld, Anthony Tomasic
Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, 15213, USA
`{oscarr, antianw, johnz, as7s, tomasic}@andrew.cmu.edu`

Abstract

Recently, transformer language models have been applied to build both task- and non-task-oriented dialogue systems. Although transformers perform well on most of the NLP tasks, they perform poorly on context retrieval and symbolic reasoning. Our work aims to address this limitation by embedding the model in an operational loop that blends both natural language generation and symbolic injection. We evaluated our system on the multi-domain DSTC8 data set and reported joint goal accuracy of 75.8% (ranked among the first half positions), intent accuracy of 97.4% (which is higher than the reported literature), and a 15% improvement for success rate compared to a baseline with no symbolic injection. These promising results suggest that transformer language models can not only generate proper system responses but also symbolic representations that can further be used to enhance the overall quality of the dialogue management as well as serving as scaffolding for complex conversational reasoning.

1 Introduction

Building task-oriented dialogue systems using a conventional pipeline approach, where modules are optimized separately, increases the fine control for dialogue management, but it does not necessarily improve overall performance (Madotto et al., 2018; Liu and Lane, 2018). In contrast, end-to-end neural models employ a straightforward training approach to generating system responses; however, this approach is impractical for goal-oriented dialogues where the system needs to interact with external systems or generate an explanation that supports its decisions (Ham et al., 2020).

Recently, the use of transformer models for building end-to-end dialogue systems has attracted considerable attention (Budzianowski and Vulić, 2019; Yang et al., 2020); however, as far as we know, current approaches operate solely at the text (word) level. We extend this approach to utilize transformer model's versatility to generate more complex constructs such as symbol representations.

In this paper, we propose a hybrid approach. We first apply a fine-tuned, end-to-end transformer model for multi-domain task-oriented dialogue. Then, during inference, we decouple the execution into expert modules that collaboratively process the content of a common knowledge base (resembling the blackboard architecture (Erman et al., 1980)).

In our experiments, we empirically demonstrate that the transformer model can be fine-tuned to generate not only text from a given input but also symbolic representations (e.g., utterances → dialogue states), manipulate those symbolic representations to generate new ones (e.g., dialogue states → system actions), and generate natural language from symbols (e.g., system actions → system response).

This work led us to a new generic reasoning architecture that leverages the ability of a transformer model to effectively manipulate representations that are mixtures of natural and symbolic language. The result is a simple architecture that uses a uniform representation to blend together dialogue aspects of interpretation, language understanding and generation, and behavior.

2 Method

Architecture: Our system resembles a blackboard architecture (Erman et al., 1980) (Figure 1) with a central memory blackboard and seven modules that implement different steps of the dialogue.

Blackboard: It provides a global memory where pieces of knowledge (history, user's intents and goals, system actions, etc.) are continuously updated by modules to maintain the dialogue context.

Forget: This module shortens sequence inputs that surpass the maximum limit of tokens that a transformer model can process at a time (in our case, 1,024 tokens for GPT-2). Additional input tokens beyond this limit are truncated, potentially

Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 438–444
July 29–31, 2021. ©2021 Association for Computational Linguistics

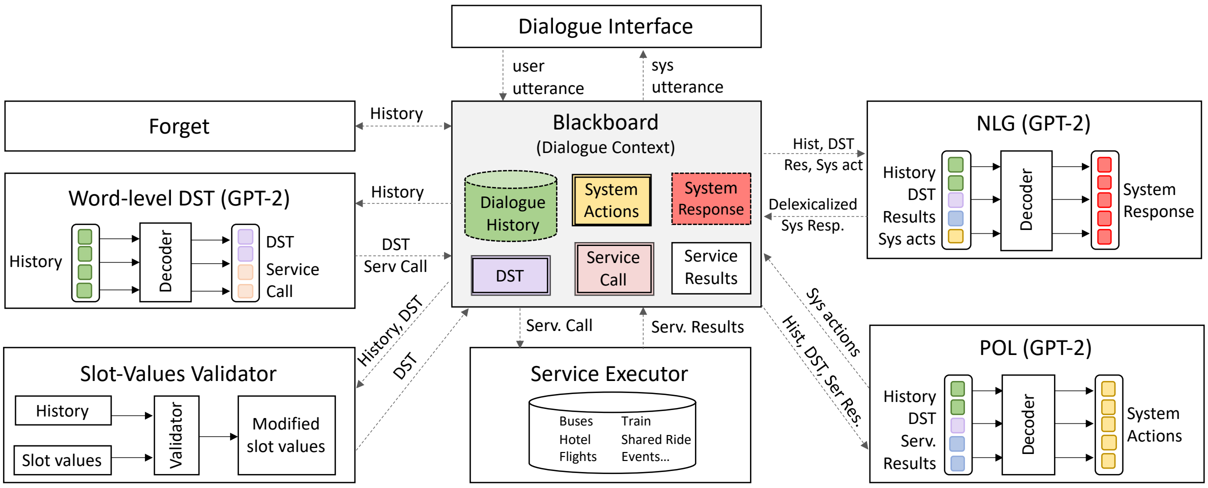


Figure 1: Dialogue System Architecture. Arrows illustrate retrieving/updating information from/to the blackboard. Boxes labeled *GPT-2* (DST, POL, NLG) represent the same neural module which is invoked multiple times using different aggregated inputs. Dotted boxes contain symbols and double-line boxes contain natural language.

discarding relevant symbols needed for dialogue processing. Instead, this module discards only the oldest (non-symbolic) elements in dialogue history to keep the input token size within the limit. A more sophisticated component that is more selective in discarding irrelevant information, chunking information, etc., is left as future work.

Word-level Dialogue State Tracking (DST): The transformer model takes the dialogue history as input and generates a symbolic dialogue state as output. Since the dialogue state's symbols (i.e., intent, service, and slot values) can be directly mapped into a service call, the model also outputs a call signature when all the required slots are met. Then, generated symbols are injected into the blackboard.

Slot-Values Validator: It checks whether the dialogue state's symbols were correctly predicted and, if so, they are injected back into the blackboard.

Service Executor: given a generated service call, the service executor queries the database and publishes the results on the blackboard.

Dialogue Policy (POL): Based on the current context, this module uses the transformer model to generate the next system actions (symbolic constructs that contain acts, slots, and values).

Natural Language Generation (NLG): Taking the current context as input, this module uses the transformer model to generate a natural language system response.

Finally, we implemented a control component that orchestrates modules' activation, allowing them to manipulate back and forth the content of the blackboard (a mixture of multi-domain, multi-intent symbols and natural language – see Alg. 1).

Fine-tuning: During training, we fine-tuned a pre-trained GPT-2 transformer model using 16,548 dialogues from the DSTC8 dataset described in section 3. To this purpose, we first pre-processed the data by encoding dialogue annotations into sequences of symbolic representation segments (for convenience, we used a Prolog-like syntax), intermixed with natural language. Then, we encoded a set of 9 special tokens, added them to our vocabulary for delimiters and segment indicators, and concatenated the segments as follows:

```
<bos><usr>...<sys>...<usr>...<dst>...
<svc>...<svr>...<sac>...<sut>...<eos>
```

Where `<bos>` and `<eos>` demarcate the beginning and end of an example; `<usr>` and `<sys>` represent the history of both user and system utterances; `<dst>` is the symbolic segment for the dialogue state tracker; `<svc>` and `<svr>` correspond to the service call and service result segments, respectively; `<sac>` are the system actions; and `<sut>` is the system utterance. Then, the forget module truncated each example as we describe before.

Although we fine-tuned the neural model end-to-end, during the test phase, we broke down the generation process into 3 main steps resembling the execution of a pipelined dialogue system (except that inputs to each module are composite structures of symbols and natural language that are assembled incrementally), as we described above (i.e., DST, POL, and NLG). This architectural breakdown allowed us to add new experts that intercept each module's outputs, manipulate the corresponding symbols, and inject the updates into the context maintained by the blackboard.

Example: consider the following snippet of a dialogue from the DSTC8 data set (in json format):

```
"turns": [
{
  "speaker": "USER",
  "utterance": "I need to take a bus from
            Las Vegas to San Francisco",
  "user_acts": ...
  ...
  "state": {
    "active_intent": "FindBus",
    "requested_slots": [],
    "slot_values": {
      "from_location": [
        "Las Vegas"
      ]
      ...
    }}
},
{
  "speaker": "SYSTEM",
  "utterance": "sure, I found 3 buses.
    One departs tomorrow at 10am...",
  "sys_acts": ...
  ...
  "service_call": {
    "method": "FindBus",
    "parameters": {
      "from_location": "Las Vegas",
        ...
      }
    },
    "service_results": [
      {
        "category": "direct",
        "departure_date": "2019-03-13",
        ...
      }]}
  ...
]
```

While user/system utterances do not require any change before being encoded as part of the fine-tuning data set (e.g., `<usr>I need to take...`), annotations for dialogue state, service calls and results, and system actions are encoded as Prolog-like compound terms (atoms followed by a comma-separated list of argument terms with variable arity). For instance, the dialogue state contains argument terms for the type of service, user's intent, and the slot values provided by the user:

```
<dst>
  has(
    state,[
      service(Buses),
      intent(FindBus),
      slot_values(
        from_location,[``Las Vegas''],
        ...
  )])
```

Likewise, the symbolic representation of the service call contains argument terms that correspond to mappings between active intent and service method, and slot values and call parameters:

```
<svc>
  call(
    Buses, [
      method(FindBus),
      parameters(
        from_location,[``Las Vegas''],
        ...
  )])
```

The service results are encoded as a list of compound terms, as follows:

```
<svr>
  results([
    idx1(slots,[
      category(``direct''),
      ...
    ]),
    idx2(slots,[
    ...]
  ])
```

Finally, the system action contains argument terms for the the type of dialogue act, the slots, and their corresponding values:

```
<sac>
  action(
    act(INFORM),
    slot(departure_time),
    value(10am)])
  ...)
```

3 Experiment Framework

We used the open-source implementation of GPT-2-small transformer model[1] with values for Adam learning rate (5.75e-5), epsilon for Adam optimizer (1e-8), and batch size (4). To generate more coherent text as proposed by Welleck et al. (2020), we set parameters top-p nucleus sampling (0.95) and top-k sampling (50) using grid search.

We utilized the Schema-Guided data set proposed at the Dialogue System Technology Challenge DSTC8-Task4[2]. We chose this data set due to: 1) its rich annotations across the whole dialogue pipeline; 2) its size that exceeds the existing dialogue corpora in scale (with over 20K multi-domain, task-oriented dialogues spanning 45 APIs over 20 domains); and 3) it contains a significant amount of dialogues for the transportation domain[3]. In this work, we only tested dialogues containing domains/services shown during training although unseen slot values were allowed (the evaluation of unseen domains is left as future work).

---

[1] https://github.com/huggingface/transformers

[2] github.com/google-research-datasets/dstc8-schema-guided-dialogue

[3] Our long-term goal is to explore the limits of the proposed hybrid approach in the context of mitigating accessibility barriers when accessing transportation information, e.g., (National Council on Disability, 2015; Steinfeld et al., 2017)

Since the DSTC8 challenge does not provide SQL scripts or equivalent, we reverse-engineered the database results from the data set and implemented our own services database.

We carried out automatic evaluation of our system on 2,361 dialogues using diverse metrics. For DST, we used the metrics provided with the data set, measuring: *average goal accuracy* (accuracy of predicting the value of a slot correctly), *joint goal accuracy* (average accuracy of predicting all slot assignments for a turn correctly), *active intent accuracy* (a fraction of user turns for which the intent was rightly predicted), and *requested slot F1* (the macro-averaged F1 score for requested slots over all eligible user turns). In addition, we extended these metrics to measure system actions in a similar way: *service call accuracy*, *joint parameter accuracy*, and *joint system action accuracy*. Also, we used *success rate* for system performance, and *BLEU* for fluency of the generated response. Human evaluation is left as future work.

Finally, *post hoc*, we ran an error analysis that let us identify the kind of error that affected system performance the most, allowing us to build a simple heuristic-based expert that focused on measuring particular kinds of modeling errors to identify areas for improving overall performance.

4 Results

We ran 3 different experiments as follows:
Exp$_o$: in order to ensure a fair comparison between the results reported in Rastogi et al. (2020) and our system's performance results, this experiment uses the ground truth values (*oracle*) of both user and system utterances. This experiment uses DSTC8 metrics and data, so our results can be compared directly to published results (26 approaches).
Exp$_g$: history is composed of gold user utterances and system utterances *generated* by our system. As opposed to the oracle experiment above, this experiment captures cascading errors that propagate from earlier steps to later steps in a dialogue.
Exp$_v$: a heuristic-based slot-value *validator* is added to Exp$_o$ to improve performance. For illustrative purposes only, this experiment measured the impact of mitigating the most critical errors (from error analysis) by manipulating symbols generated by GPT-2 (Exp$_g$). These results precisely identify weaknesses in the current model.

Approach	JGA	AGA	IA	RSF1
Team 9	**0.924**	**0.979**	0.957	0.993
Team 10	0.920	0.978	0.956	0.847
Exp$_v$	0.917	0.956	**0.974**	0.985
Team 8	0.910	0.970	N.A.	0.847
Team 14	0.900	0.960	0.957	**0.996**
Team 5	0.893	0.966	0.959	0.992
Exp$_o$	0.758	0.939	**0.974**	0.985
Exp$_g$	0.639	0.892	0.935	0.974
Baseline	0.412	0.677	0.950	0.995

Table 1: Overall results of DST evaluation. JGA: joint goal accuracy, AGA: average goal accuracy, IA: intent accuracy, and RSF1: requested slot F1 score. Due to space constraints, we only include the top-5 results reported in Rastogi et al. (2020).

DST Evaluation Results: overall, when compared to the *seen-services* results reported in Rastogi et al. (2020), our system outperformed other models on intent accuracy (see Table 1). Correctly predicting the intent demonstrates the ability of our system to track user's intentions and effectively detect domain switches. In addition, if we consider the 26 teams who participated in DSTC8-T4, our system ranks among the first half positions in Exp$_o$ and the first 2/3 positions in Exp$_g$. Finally, Exp$_v$ made an improvement in JGA of 21% and 43% over Exp$_o$ and Exp$_g$, respectively. Details of the heuristic-based module are described in the next section.

Error Analysis: from all the reported metrics, we focused on the results obtained for the Joint Goal Accuracy (JGA) for two reasons: 1) JGA is the primary evaluation metric used for ranking approaches submitted to DSTC8-Task4; and 2) this metric got the lowest scores for Exp$_o$ and Exp$_g$ among all the evaluation metrics (see Table 1).

From the error analysis, we found 3 main kinds of errors that affect JGA: 1) slot names were correctly predicted but slot values were not (10.5% of errors); 2) slot names that appeared in the gold DST but were not predicted by the system (29.1%); and 3) slot names that were predicted but did not appear in the gold DST (60.4%).

Given its significant presence, we focus on the third kind of error. The main causes for this error to occur are: 1) the slot value is predicted but not mentioned by either the user or the system in the dialogue history (over-fitting); and 2) the slot value is mentioned/offered by the system but not accepted by the user (e.g., the system says "There is a direct bus that departs at 9:50 am and costs $36.", where the slot `trip_fare` was unsolicited by the user, and then the user says "hmm any buses departing in the

Appr.	SCA	JPA	JSA	SR	BLEU
Exp$_v$	0.927	0.891	0.786	82.21%	2.14
Exp$_o$	0.927	0.828	0.786	80.45%	2.05
Exp$_g$	0.873	0.703	0.748	71.25%	1.63

Table 2: Overall results of the System Actions evaluation. SCA: service call accuracy, JPA: joint service call's parameter accuracy, JSA: joint system action accuracy, and SR: success rate.

afternoon?" only confirming `departure_time`).

We implemented a heuristic-based *slot-value validator* to mitigate the error above. First, we extracted and classified all the slot values from the training data set and store them in a dictionary. Then, a set of heuristic rules based on fuzzy string matching determine whether a slot value is present in the dialogue history by calculating its similarity with the values in the dictionary, fixing the first cause of the error. Next, if the slot value is mentioned only by the system, the value is retained only if the system offered the value at any prior turn (i.e., `sys_act: ‘‘OFFER’’`) and the user accepted the offered slot value in the next turn (i.e., `user_act: ‘‘SELECT’’ | ‘‘AFFIRM’’`).

System Actions and Performance Results: From Table 2, Exp$_o$ mainly improved JPA, SR, and BLEU over Exp$_g$ by 18%, 13%, and 26%, respectively. Clearly, some down-stream error propagation occurs. On the other hand, Exp$_v$ slightly improved JPA (8%) over Exp$_o$ due to fixing some of the DST issues also improved the quality of predicting service parameters. Finally, although BLEU scores are low (due to there was available only one single reference value per turn), they are paired with high success rates – in fact, a manual inspection of system utterances indicates an overall high quality of language generation (see Figure 2).

5 Related Work

In comparison with traditional pipelined dialogue architectures (Chen et al., 2017; Bohus and Rudnicky, 2009) where NLU (Lee et al., 2019), DST (Williams et al., 2013), and POL/NLG (Wen et al., 2015) modules are optimized separately; our architecture is simpler and less prone to cascading failures due to the folding of multiple NLP tasks into a single transformer model and the exposure of symbolic representation directly to the model.

More recently, pre-trained language models similar to GPT-2 have been used for building end-to-end dialogue systems. Our approach is similar in nature to the work proposed by (Hosseini-Asl et al., 2020; Peng et al., 2020) in that we use a single causal language model to generate all outputs given a dialogue context. However, unlike these approaches, our model not only encodes DST and database results (which shows a labeling cost reduction) but also encodes dialogue policy and service call templates, allowing the system to be able to monitor errors and manipulate symbolic representations at different stages of turn processing.

Similar to other transformer dialog systems (Wolf et al., 2019; Ramadan et al., 2018), our model learns from text; however, our model also learns and generates complex structures that intermix natural and symbolic language. In particular, the work described by Budzianowski and Vulić (2019) and Yang et al. (2020) encodes both belief state and knowledge base constructs into simple text representations and generates text-only outputs. In contrast, our approach encodes, manipulates, and generates more sophisticated knowledge representations, roughly first-order logic constant terms that are implicitly learned and which could be used to communicate with external sources and expert components such as a symbolic reasoner.

Dialogue systems have many similarities to conversational workflow systems. The Virtual Information Officer (Tomasic et al., 2007) required more than thirty individual models performing task classification, entity resolution, and information extraction. Moreover, (Romero et al., 2019) discuss the challenges found when directly translating natural language inputs into symbolic API calls in a service composition system. Both systems would benefit from the architecture and method presented here.

6 Conclusions

In this paper, we empirically demonstrated that several capabilities of transformer language models can be leveraged to construct a new dialogue architecture that is more flexible and simpler (resulting in much lower engineering costs) and extensible (allowing symbolic injection and manipulation), while retaining reasonable performance.

7 Acknowledgements

The contents of this paper were developed under grants from the National Institute on Disability, Independent Living, and Rehabilitation Research (NIDILRR grant numbers 90DPGE0003 and 90REGE0007).

References

Dan Bohus and Alexander I. Rudnicky. 2009. The ravenclaw dialog management framework: Architecture and systems. *Comput. Speech Lang.*, 23(3):332–361.

Paweł Budzianowski and Ivan Vulić. 2019. Hello, It's GPT-2-How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 15–22.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *SIGKDD Explor. Newsl.*, 19(2):25–35.

Lee D Erman, Frederick Hayes-Roth, Victor R Lesser, and D Raj Reddy. 1980. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys (CSUR)*, 12(2):213–253.

Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592. Association for Computational Linguistics.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191. Curran Associates, Inc.

Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. SUMBT: Slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5478–5483, Florence, Italy. Association for Computational Linguistics.

Bing Liu and Ian Lane. 2018. End-to-end learning of task-oriented dialogs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 67–73, New Orleans, Louisiana, USA. Association for Computational Linguistics.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. *CoRR*, abs/1804.08217.

National Council on Disability. 2015. Transportation update: Where we've gone and what we've learned.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020. SOLOIST: few-shot task-oriented dialog with A single pre-trained auto-regressive model. *CoRR*, abs/2005.05298.

Osman Ramadan, Paweł Budzianowski, and Milica Gasic. 2018. Large-scale multi-domain belief tracking with knowledge sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 432–437.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Schema-Guided Dialogue State Tracking Task at DSTC8. *arXiv preprint arXiv:2002.01359*.

Oscar J. Romero, A. Dangi, and S. A. Akoju. 2019. NLSC: Unrestricted Natural Language-Based Service Composition through Sentence Embeddings. In *2019 IEEE International Conference on Services Computing (SCC)*, pages 126–135.

Aaron Steinfeld, Jordana L Maisel, and Edward Steinfeld. 2017. *Accessible Public Transportation: Designing Service for Riders with Disabilities*. Routledge.

Anthony Tomasic, Isaac Simmons, and John Zimmerman. 2007. Learning information intent via observation. In *Proceedings of the 16th international conference on World Wide Web*, pages 51–60.

Sean Welleck, Ilia Kulikov, Jaedeok Kim, Richard Yuanzhe Pang, and Kyunghyun Cho. 2020. Consistency of a recurrent language model with respect to incomplete decoding. *arXiv preprint arXiv:2002.02492*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

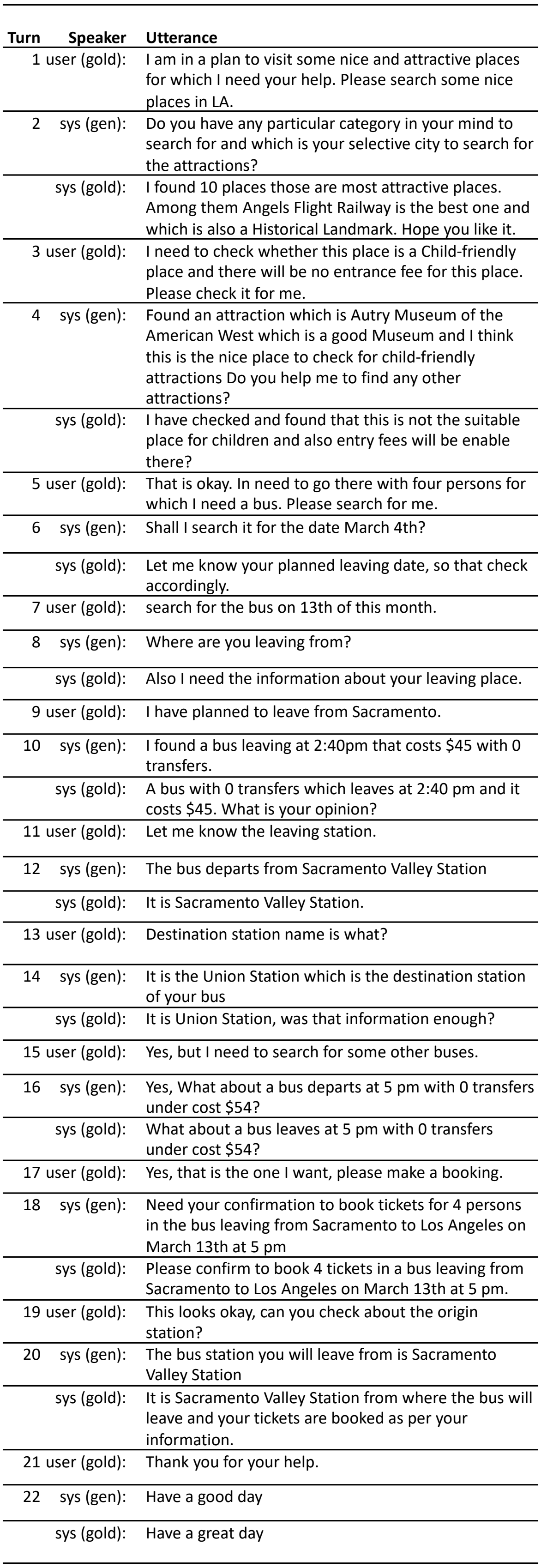
Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, Metz, France. Association for Computational Linguistics.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2020. Ubar: Towards fully end-to-end task-oriented dialog systems with gpt-2. *arXiv preprint arXiv:2012.03539*.

Algorithm 1 Main Flow of Control

```
dialogue_control():
Input: $signal = start$
while $signal$ != $end$ do
    $usr\_utt$ ← get_user_input().
    $sys\_utt$ ← process_response($usr\_utt$).
    return $sys\_utt$.
end while
process_response():
Input: $usr\_utt$
Output: $sys\_utt$
$bb$ ← update_blackboard($usr\_utt$).
$dst$, $serv\_call$ ← exec_gpt2("DST", $bb$).
$bb$ ← update_blackboard($dst$, $serv\_call$).
$dst$ ← validate_slots($bb$).
$bb$ ← update_blackboard($dst$).
$results$ ← exec_serv_call($serv\_call$).
$bb$ ← update_blackboard($results$).
$sys\_acts$ ← exec_gpt2("POL", $bb$).
$bb$ ← update_blackboard($sys\_acts$).
$sys\_utt$ ← exec_gpt2("NLG", $bb$).
$bb$ ← update_blackboard($sys\_utt$).
return $sys\_utt$.
exec_gpt2():
Input: $id, bb$ # bb: blackboard
Output: $outputs$
if $id$ = "DST" then
    $input$ ← encode($bb.hist$).
else if $id$ = "POL" then
    $input$ ← encode($bb.hist$, $bb.dst$, $bb.results$).
else if $id$ = "NLG" then
    $input$ ← encode($bb.hist$, $bb.dst$, $bb.results$, $bb.sys\_acts$).
end if
if length of $input$ > max_token_length then
    $input$ ← forget($input$). # truncation
end if
$outputs$ ← predict_gpt2($input$).
return $outputs$
update_blackboard():
Input: $updates$
Output: $blackboard$
for $element$ in $updates$ do
    update($blackboard$, $element$).
end for
return $blackboard$
```



Turn	Speaker	Utterance
1	user (gold):	I am in a plan to visit some nice and attractive places for which I need your help. Please search some nice places in LA.
2	sys (gen):	Do you have any particular category in your mind to search for and which is your selective city to search for the attractions?
	sys (gold):	I found 10 places those are most attractive places. Among them Angels Flight Railway is the best one and which is also a Historical Landmark. Hope you like it.
3	user (gold):	I need to check whether this place is a Child-friendly place and there will be no entrance fee for this place. Please check it for me.
4	sys (gen):	Found an attraction which is Autry Museum of the American West which is a good Museum and I think this is the nice place to check for child-friendly attractions Do you help me to find any other attractions?
	sys (gold):	I have checked and found that this is not the suitable place for children and also entry fees will be enable there?
5	user (gold):	That is okay. In need to go there with four persons for which I need a bus. Please search for me.
6	sys (gen):	Shall I search it for the date March 4th?
	sys (gold):	Let me know your planned leaving date, so that check accordingly.
7	user (gold):	search for the bus on 13th of this month.
8	sys (gen):	Where are you leaving from?
	sys (gold):	Also I need the information about your leaving place.
9	user (gold):	I have planned to leave from Sacramento.
10	sys (gen):	I found a bus leaving at 2:40pm that costs $45 with 0 transfers.
	sys (gold):	A bus with 0 transfers which leaves at 2:40 pm and it costs $45. What is your opinion?
11	user (gold):	Let me know the leaving station.
12	sys (gen):	The bus departs from Sacramento Valley Station
	sys (gold):	It is Sacramento Valley Station.
13	user (gold):	Destination station name is what?
14	sys (gen):	It is the Union Station which is the destination station of your bus
	sys (gold):	It is Union Station, was that information enough?
15	user (gold):	Yes, but I need to search for some other buses.
16	sys (gen):	Yes, What about a bus departs at 5 pm with 0 transfers under cost $54?
	sys (gold):	What about a bus leaves at 5 pm with 0 transfers under cost $54?
17	user (gold):	Yes, that is the one I want, please make a booking.
18	sys (gen):	Need your confirmation to book tickets for 4 persons in the bus leaving from Sacramento to Los Angeles on March 13th at 5 pm
	sys (gold):	Please confirm to book 4 tickets in a bus leaving from Sacramento to Los Angeles on March 13th at 5 pm.
19	user (gold):	This looks okay, can you check about the origin station?
20	sys (gen):	The bus station you will leave from is Sacramento Valley Station
	sys (gold):	It is Sacramento Valley Station from where the bus will leave and your tickets are booked as per your information.
21	user (gold):	Thank you for your help.
22	sys (gen):	Have a good day
	sys (gold):	Have a great day

Figure 2: Sample of a multi-domain dialogue. For each gold user utterance, we provide both the gold system utterance and the utterance generated by our system.