A SET OF TASKS FOR ANALYZING GOAL-ORIENTED END-TO-END DIALOG

Antoine Bordes, Y-Lan Boureau & Jason Weston
Facebook AI Research
New York, USA
{abordes, ylan, jase}@fb.com

1 TASKS

End-to-end dialog systems make no assumption on the domain or dialog state structure, thus holding the promise of easily scaling up to new domains. This proposal aims to make it easier to analyze the performance of end-to-end systems in a goal directed setting, using our published open resource (Bordes *et al.*, 2017). It breaks down a goal-directed objective into several subtasks to test some crucial capabilities that dialog systems should have (and hence provide error analysis by design). All our tasks involve a restaurant reservation system, where the goal is to book a table at a restaurant. Solving our tasks requires manipulating both natural language and symbols from a KB. The first five tasks are generated by a simulation, the last one uses real human-bot dialogs.

Grounded with an underlying KB of restaurants and their properties (location, type of cuisine, etc.), these tasks cover several dialog stages and test if models can learn various abilities such as performing dialog management, querying KBs, interpreting the output of such queries to continue the conversation or dealing with new entities not appearing in dialogs from the training set.

Task 1: Issuing API calls A user request implicitly defines a query that can contain from 0 to 4 of the required fields (sampled uniformly; in Figure 1, it contains 3). The bot must ask questions for filling the missing fields and eventually generate the correct corresponding API call. The bot asks for information in a deterministic order, making prediction possible.

Task 2: Updating API calls Starting by issuing an API call as in Task 1, users then ask to update their requests between 1 and 4 times (sampled uniformly). The order in which fields are updated is random. The bot must ask users if they are done with their updates and issue the updated API call.

Task 3: Displaying options Given a user request, we query the KB using the corresponding API call and add the facts resulting from the call to the dialog history. The bot must propose options to users by listing the restaurant names sorted by their corresponding rating (from higher to lower) until users accept. For each option, users have a 25% chance of accepting. If they do, the bot must stop displaying options, otherwise propose the next one. Users always accept the option if this is the last remaining one. We only keep examples with API calls retrieving at least 3 options.

Task 4: Providing extra information Given a user request, we sample a restaurant and start the dialog as if users had agreed to book a table there. We add all KB facts corresponding to it to the dialog. Users then ask for the phone number of the restaurant, its address or both, with proportions 25%, 25% and 50% respectively. The bot must learn to use the KB facts correctly to answer.

Task 5-6: Conducting full dialogs For Task 5, we combine Tasks 1-4 to generate full dialogs just as in Figure 1. Unlike in Task 3, we keep examples if API calls return at least 1 option instead of 3. Task 6 is the same task, but on DSTC2 data (see below).

2 DATA

2.1 RESTAURANT RESERVATION SIMULATION

The simulation is based on an underlying KB, whose facts contain the restaurants that can be booked and their properties. Each restaurant is defined by a type of cuisine (10 choices, e.g., French, Thai), a location (10 choices, e.g., London, Tokyo), a price range (cheap, moderate or expensive) and a rating (from 1 to 8). For simplicity, we assume that each restaurant only has availability for a single party size (2, 4, 6 or 8 people). Each restaurant also has an address and a phone number listed in the KB.

The KB can be queried using API calls, which return the list of facts related to the corresponding restaurants. Each query must contain four fields: a location, a type of cuisine, a price range and a party size. It can return facts concerning one, several or no restaurant (depending on the party size).

Using the KB, conversations are generated in the format shown in Figure 1. Each example is a dialog comprising utterances from a user and a bot, as well as API calls and the resulting facts. Dialogs are generated after creating a user request by sampling an entry for each of the four required fields: e.g. the request in Figure 1 is [cuisine: British, location: London, party size: six, price range: expensive]. We use natural language patterns to create user and bot utterances. There are 43 patterns for the user and 20 for the bot (the user can use up to 4 ways to say something, while the bot always uses the same). Those patterns are combined with the KB entities to form thousands of different utterances.

We want to test how well models handle entities appearing in the KB but not in the dialog training sets. We split types of cuisine and locations in half, and create two KBs, one with all facts about restaurants within the first halves and one with the rest. This yields two KBs of 4,200 facts and 600 restaurants each (5 types of cuisine $\times$ 5 locations $\times$ 3 price ranges $\times$ 8 ratings) that only share price ranges, ratings and party sizes, but have disjoint sets of restaurants, locations, types of cuisine, phones and addresses. We use one of the KBs to generate the standard training, validation and test dialogs, and use the other KB only to generate test dialogs, termed Out-Of-Vocabulary (OOV) test sets.

For training, systems have access to the training examples and both KBs. Evaluation is conducted on both test sets, plain and OOV. Beyond the intrinsic difficulty of each task, the challenge on the OOV test sets is for models to generalize to new entities (restaurants, locations and cuisine types) unseen in any training dialog – something natively impossible for embedding methods. Ideally, models could, for instance, leverage information coming from the entities of the same type seen during training.

We generate five datasets, one per task. Table 1 gives their statistics. Training sets are relatively small (1,000 examples) to create realistic learning conditions. The dialogs from the training and test sets are different, never being based on the same user requests. Thus, we test if models can generalize to new combinations of fields.

2.2 Dialog State Tracking Challenge

Since our tasks rely on synthetically generated language for the user, we supplement our dataset with real human - bot dialogs. We use data from DSTC2 (Henderson *et al.*, 2014), that is also in the restaurant booking domain. Unlike our tasks, its user requests only require 3 fields: type of cuisine (91 choices), location (5 choices) and price range (3 choices). The dataset was originally designed for dialog state tracking hence every dialog turn is labeled with a state (a user intent + slots) to be predicted. As our goal is to evaluate end-to-end training, we did not use that, but instead converted the data into the format of our 5 tasks and included it in the dataset as Task 6.

We used the provided speech transcriptions to create the user and bot utterances, and given the dialog states we created the API calls to the KB and their outputs which we added to the dialogs. We also added ratings to the restaurants returned by the API calls, so that the options proposed by the bots can be consistently predicted (by using the highest rating). We did use the original test set but use a slightly different training/validation split.

This dataset has similar statistics to our Task 5 (see Table 1) but is harder. The dialogs are noisier and the bots made mistakes due to speech recognition errors or misinterpretations and also do not always have a deterministic behavior (the order in which they can ask for information varies).

3 Evaluation

Evaluation uses two metrics, per-response and per-dialog accuracies, the latter tracking completion of the actual goal. Evaluation is conducted in a ranking, not a generation, setting: at each turn of the dialog, we test whether they can predict bot utterances and API calls by selecting a candidate, not by generating it.[1] Candidates are ranked from a set of all bot utterances and API calls appearing in training, validation and test sets (plain and OOV) for all tasks combined.

[1] Lowe *et al.* (2016) termed this setting Next-Utterance-Classification.

REFERENCES

Bordes, A., Boureau, Y.-L., and Weston, J. (2017). earning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Henderson, M., Thomson, B., and Williams, J. (2014). The second dialog state tracking challenge. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 263.

Liu, C.-W., Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L., and Pineau, J. (2016). How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L., and Pineau, J. (2016). On the evaluation of dialogue systems with next utterance classification. *arXiv preprint arXiv:1605.05414*.

Weston, J., Bordes, A., Chopra, S., and Mikolov, T. (2015). Towards ai-complete question answering: a set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

A DATA STATISTICS

Table 1: **Data provided** Tasks 1-5 were generated using our simulator and share the same KB. Task 6 was converted from the 2^{nd} Dialog State Tracking Challenge (Henderson *et al.*, 2014). $^{(*)}$ Tasks 1-5 have two test sets, one using the vocabulary of the training set and the other using out-of-vocabulary words.

	Tasks	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**
DIALOGS	Number of utterances:	12	17	43	15	55	54
Average statistics	- user utterances	5	7	7	4	13	6
	- bot utterances	7	10	10	4	18	8
	- outputs from API calls	0	0	23	7	24	40
DATASETS	Vocabulary size	3,747					1,229
Tasks 1-5 share the same data source	Candidate set size	4,212					2,406
	Training dialogs	1,000					1,618
	Validation dialogs	1,000					500
	Test dialogs	1,000$^{(*)}$					1,117

B ILLUSTRATION OF TASKS

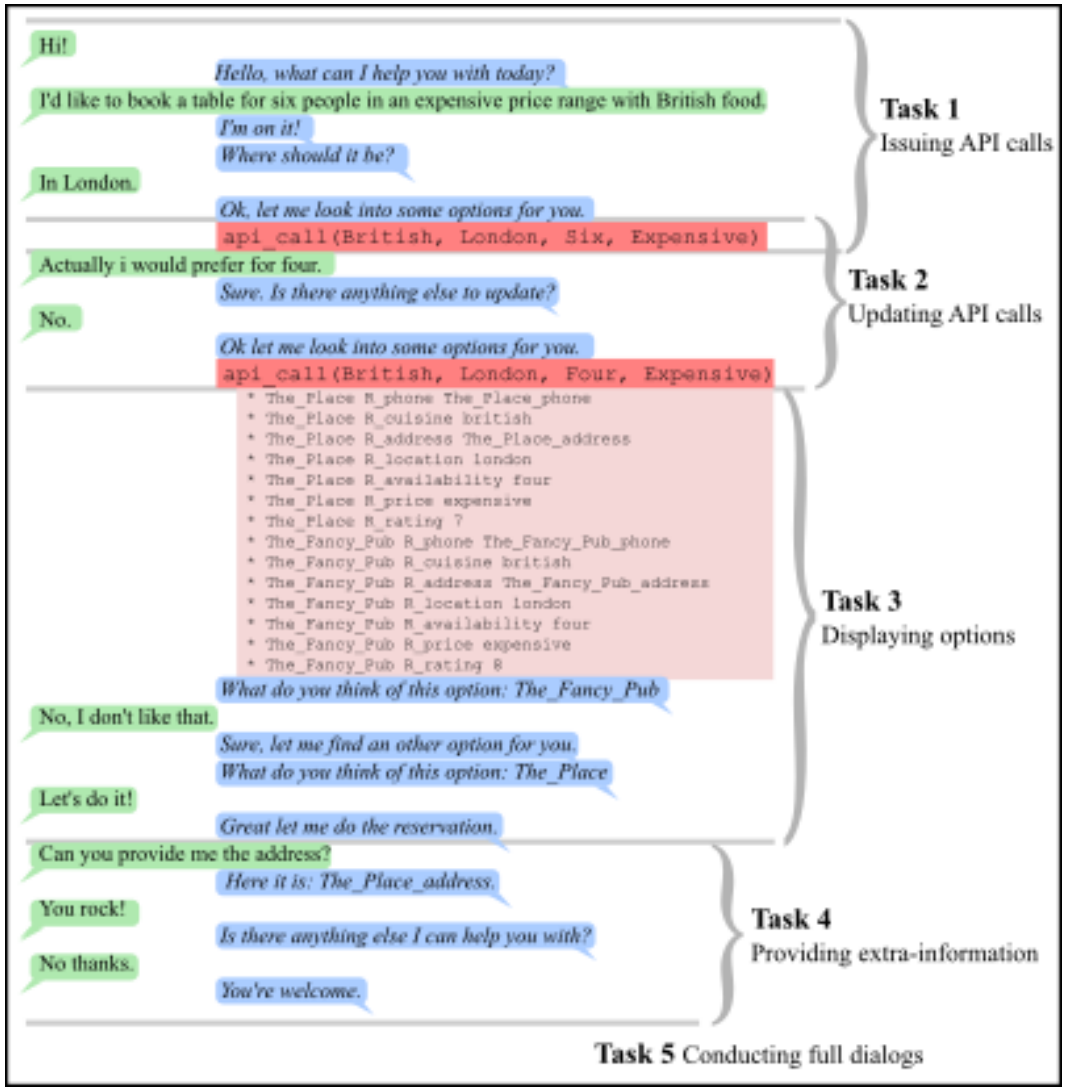

Figure 1: **Goal-oriented dialog tasks.** A user (in green) chats with a bot (in blue) to book a table at a restaurant. Models must predict bot utterances and API calls (in dark red). Task 1 tests the capacity of interpreting a request and asking the right questions to issue an API call. Task 2 checks the ability to modify an API call. Task 3 and 4 test the capacity of using outputs from an API call (in light red) to propose options (sorted by rating) and to provide extra-information. Task 5 combines everything.