纯粹一个比赛介绍。

Dialog System & Technology Challenge 6 Overview of Track 1 - End-to-End Goal-Oriented Dialog learning

Julien Perez[1] *and Y-Lan Boureau*[2] *and Antoine Bordes*[2]

[1]Naver Labs Europe, Grenoble, France
[2]Facebook AI Research, New York, USA

julien.perez@naverlabs.com, ylan@fb.com, abordes@fb.com

Abstract

End-to-end dialog learning is an important research subject in the domain of conversational systems. The primary task consists in learning a dialog policy from transactional dialogs of a given domain. In this context, usable datasets are needed to evaluate learning approaches, yet remain scarce. For this challenge, a transaction dialog dataset has been produced using a dialog simulation framework developed and released by Facebook AI Research. Overall, nine teams participated in the challenge. In this report, we describe the task and the dataset. Then, we specify the evaluation metrics for the challenge. Finally, the results of the submitted runs of the participants are detailed.

1. Introduction

Goal-oriented dialog requires skills that go beyond language modeling. For example, asking questions to clearly define a user request, querying Knowledge Bases (KBs), interpreting results from queries to display options to users or completing a transaction are some of the important competencies a dialog system has to master in order to be useful. On the one hand, such difficulties make it hard to ascertain how well end-to-end dialog models would do, and whether they are in a position to replace traditional dialog methods in a goal-directed setting. On the other hand, because end-to-end dialog systems make no assumption on the domain or dialog state structure, they are holding the promise of easily scaling up to new domains. This challenge aims to make it easier to analyze the performance of end-to-end systems in a goal directed setting, using an expanded version of the Facebook AI Research open resource proposed in [1]. The goal of the challenge is to assess the capabilities of the proposed systems to fulfill a set of four basic tasks related to transactional dialogs. The capability of accomplishing all four tasks on a single dialog corpus has been tested as a final task.

Roadmap: Section 2 describes the task, the dialog corpora and the knowledge bases used as support for each of the individual tasks. In Section 3 we describe the datasets of dialog produced for each of the individual tasks. Then, Section 4 recalls the file format and the evaluation metrics used for the challenge. Finally, we presents the methods proposed by the nine teams and the corresponding results.

2. The task - Restaurant Reservation

The transactional dialog simulation system is based on an underlying KB. The facts contain the restaurants that can be booked and their properties queried. 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), a rating (from 1 to more than 200), and other characteristics like dietary restrictions and atmosphere. 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 a certain number of slots: a location, a type of cuisine, a price range, a party size, and possibly other required slots like dietary restriction, depending on the set used. Each data file has the same set of required slots for every dialog. A query 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 3. 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 required slots: e.g. the request in Figure 3 is [cuisine: British, location: London, party size: six, price range: expensive]. We use natural language patterns to create user and bot utterances. There are more patterns for the user than for the bot. Indeed, the user can use several ways to say something, while the bot always uses the same way to make it deterministic. Those patterns are combined with the KB entities to form thousands of different utterances. 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. In [1], the two KBs had 4,200 facts and 600 restaurants each (5 types of cuisine $\times$ 5 locations $\times$ 3 price ranges $\times$ 8 ratings). The data provided here has been expanded to comprise more slots and thus yield many more restaurants, but the two KB still have disjoint sets of restaurants, locations, types of cuisine, phones and addresses, while sharing all other sets of values. We use one of the KBs to generate train and test dialogs, using only one of the extra slots in the queries. There are 4 sets of test dialogs: (1) one that uses the same KB as for the train dialogs, and the same set of slots in the queries; (2) one that uses the second KB (with disjoint sets of restaurants, locations, cuisines, phones and addresses), termed Out-Of-Vocabulary (OOV), but the same set of slots in the queries; (3) one that uses the same KB as for the train dialogs, but one additional slot for the queries; and (4) one that uses the second KB (OOV) and an additional required slot.

For training, systems have access to the training examples and both KBs. Evaluation is conducted on all four test sets. 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. Training sets are relatively small (10,000 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.

3. Description of the Dialog Dataset

We broke 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 the 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 tasks are generated by a simulation. Grounded with an underlying KB of restaurants and their properties (location, type of cuisine, etc.), these tasks cover several dialog stages and test if candidate 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. 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. 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, the KB is queried using the corresponding API call and the resulting facts are added to the dialog history (if too many facts satisfy the call, a random subset is returned to avoid overly lengthy data). 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: Conducting full dialogs. For Task 5, we combine Tasks 1-4 to generate full dialogs just as in Figure 3. Unlike in Task 3, we keep examples if API calls return at least 1 option instead of 3.

4. Dataset Format and Evaluation metrics

The dataset is organized in 5 JSON files corresponding to each of the tasks previously mentioned. All tasks are encoded with the same format. Basically, a dialog piece is followed by a list of next-utterance candidates. In the training set, the answer (the single correct candidate) is provided. The goal is to rank the candidates. Precisions @{1,2,5} will be used to evaluate the models. Rank of candidate utterances will be 1-indexed.

```
[
        {dialog_id : " ",
        utterances: [" ", ... ],
        candidates: [
                {candidate_id: " " ,
                    utterance: ""}, ...
                ]
        }
]
```

Figure 1: *JSON Format of a dialog dataset file*

```
[
        {dialog_id : " ",
        lst_candidate_id:
                [ {candidate_id: " ", rank
                    : " "}, ... ]
        }
]
```

Figure 2: *JSON Format of a result file*

Evaluation uses per-response accuracies. Evaluation is conducted in a ranking, not a generation, setting: at each turn of the dialog, the participants have to 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 candidate utterances and API calls.

5. Results

Table 1 introduces methods proposed during the challenge. End-to-End Memory Network [3], Dynamic Memory Network [4] and Hybrid Code Networks [5] were the main trainable building blocks of the proposed systems. In addition contextual rules were proposed to improve performances.

Table 2 details the results obtained for the participating teams. The two first teams managed to solve the task by obtaining 1.0 precision in the test-set.

6. Summary and Future work

We described our dialogue dataset for end to end dialog learning formalized as a ranking task. Nine teams have participated to the challenge proposing original contribution blending end-to-end trainable models associated to meaninfull priori knowledge. This results seems to confirm that such models are a promising direction toward end-to-end goal oriented dialog learning. In the future, it will be interesting to extend the current dialog task to even reacher context and reasoning capabilities like counting, time and space reasoning.

7. Acknowledgments

We would like to thank all of the participants for their effort in exploring new methods and submitting their runs and reports. We also thank Seokhwan Kim, Chiori Hori, Koichiro Yoshino

[1] [2] termed this setting Next-Utterance Classification.

Method	Organization	Description of the runs
End-to-End Recurrent Entity Network for Entity-Value Independent Goal-Oriented Dialog Learning	Team 3	Recurrent Entity Networks with a delexicalization mechanism.
Learning Dynamic Memory Network with Two Views	Team 5	A dialogue model learnt with a Dynamic Memory Network and negative sampling.
Modeling Conversations to Learn Responding Policies of E2E Task-oriented Dialog System	Team 2	a hierarchical Long Short-Term Memory (LSTM) based ranking module, a Conditional Random Field (CRF) based slot confirming module, and a heuristic scoring module.
End-to-End Memory Network with word abstraction and contextual-numbering for goal oriented task	Team 7	Combination of End-to-End Memory Networks with named entities abstraction and contextual numbering.
End to End Goal Oriented Dialog Learning Based on Memory Network	Team 6	Memory Network with an extra output memory representation named D-Layer with Knowledge based enhancement.
Extended Hybrid Code Networks for DSTC6 task-1 Dialog Dataset	Team 1	Extended Hybrid Code Networks with trainable submodules.
Quantized-Dialog Language Model for Goal-Oriented Conversational Systems	Team 9	The key idea is to quantize the dialog space into clusters and create a language model across the clusters, thus allowing for an accurate choice of the next utterance in the conversation.

Table 1: *Description of the systems based on workshop submission*

Rank	Teams
Group 1 : Solved, Precision@1 = 1.0	1, 2
Group 2 : 0.9 < Precision@1 < 1.0	3, 6, 9
Group 3 : 0.5 < Precision@1 < 0.9	5
Group 4 : Precision@1 < 0.5	7, 8, 10

Table 2: *Summaries of the team performances*

and Jason Williams for their precious help during all the stages of this challenge. We also thank the Japanese Society for Artificial Intelligence (JSAI) for supporting the event.

8. References

[1] A. Bordes, Y.-L. Boureau, and J. Weston, "earning end-to-end goal-oriented dialog," *arXiv preprint arXiv:1605.07683*, 2017.

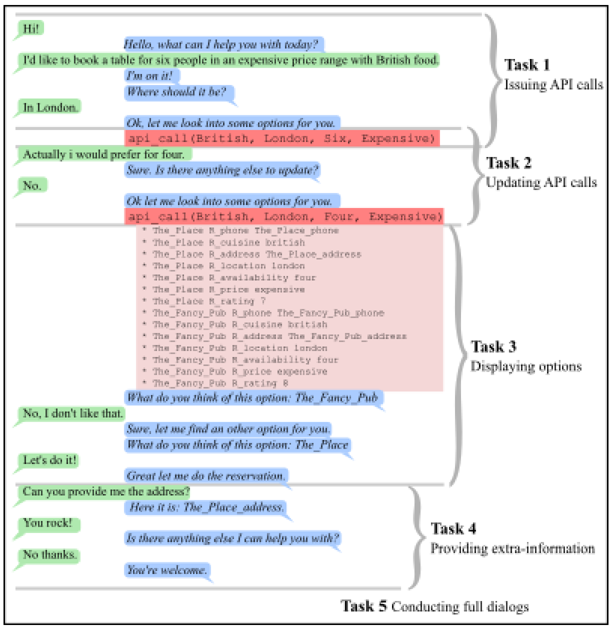
[2] R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau, "On the evaluation of dialogue systems with next utterance classification," *arXiv preprint arXiv:1605.05414*, 2016.

[3] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "Weakly supervised memory networks," *CoRR*, vol. abs/1503.08895, 2015. [Online]. Available: http://arxiv.org/abs/1503.08895

[4] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," *CoRR*, vol. abs/1506.07285, 2015. [Online]. Available: http://arxiv.org/abs/1506.07285

[5] J. D. Williams, K. Asadi, and G. Zweig, "Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning," *CoRR*, vol. abs/1702.03274, 2017. [Online]. Available: http://arxiv.org/abs/1702.03274

9. Illustration of tasks and detailed results


Hi!

Hello, what can I help you with today?

I'd like to book a table for six people in an expensive price range with British food.

I'm on it!

Where should it be?

In London.

Ok, let me look into some options for you.

```
api_call(British, London, Six, Expensive)
```

Actually i would prefer for four.

Sure. Is there anything else to update?

No.

Ok let me look into some options for you.

```
api_call(British, London, Four, Expensive)
```

```
* The_Place R_phone The_Place_phone
* The_Place R_cuisine british
* The_Place R_address The_Place_address
* The_Place R_location london
* The_Place R_availability four
* The_Place R_price expensive
* The_Place R_rating 7
* The_Fancy_Pub R_phone The_Fancy_Pub_phone
* The_Fancy_Pub R_cuisine british
* The_Fancy_Pub R_address The_Fancy_Pub_address
* The_Fancy_Pub R_location london
* The_Fancy_Pub R_availability four
* The_Fancy_Pub R_price expensive
* The_Fancy_Pub R_rating 8
```

What do you think of this option: The_Fancy_Pub

No, I don't like that.

Sure, let me find an other option for you.

What do you think of this option: The_Place

Let's do it!

Great let me do the reservation.

Can you provide me the address?

Here it is: The_Place_address.

You rock!

Is there anything else I can help you with?

No thanks.

You're welcome.

Task 1 Issuing API calls
Task 2 Updating API calls
Task 3 Displaying options
Task 4 Providing extra-information
Task 5 Conducting full dialogs

Figure 3: **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.*

File	Precision @1	Precision @2	Precision @5
tst1/task1	1.0	1.0	1.0
tst1/task2	1.0	1.0	1.0
tst1/task3	1.0	1.0	1.0
tst1/task4	1.0	1.0	1.0
tst1/task1	1.0	1.0	1.0
tst1/task2	1.0	1.0	1.0
tst1/task3	1.0	1.0	1.0
tst1/task4	1.0	1.0	1.0
tst1/task5	1.0	1.0	1.0
tst2/task1	1.0	1.0	1.0
tst2/task2	1.0	1.0	1.0
tst2/task3	1.0	1.0	1.0
tst2/task4	1.0	1.0	1.0
tst2/task5	1.0	1.0	1.0
tst3/task1	1.0	1.0	1.0
tst3/task2	1.0	1.0	1.0
tst3/task3	1.0	1.0	1.0
tst3/task4	1.0	1.0	1.0
tst3/task5	1.0	1.0	1.0
tst4/task1	1.0	1.0	1.0
tst4/task2	1.0	1.0	1.0
tst4/task3	1.0	1.0	1.0
tst4/task4	1.0	1.0	1.0
tst4/task5	1.0	1.0	1.0

Table 3: *Results for team-1 and team-4 (double submissions)*

File	Precision @1	Precision @2	Precision @5
tst1/task1	1.0	1.0	1.0
tst1/task2	1.0	1.0	1.0
tst1/task3	1.0	1.0	1.0
tst1/task4	1.0	1.0	1.0
tst1/task5	1.0	1.0	1.0
tst2/task1	1.0	1.0	1.0
tst2/task2	1.0	1.0	1.0
tst2/task3	1.0	1.0	1.0
tst2/task4	1.0	1.0	1.0
tst2/task5	1.0	1.0	1.0
tst3/task1	1.0	1.0	1.0
tst3/task2	1.0	1.0	1.0
tst3/task3	1.0	1.0	1.0
tst3/task4	1.0	1.0	1.0
tst3/task5	1.0	1.0	1.0
tst4/task1	1.0	1.0	1.0
tst4/task2	1.0	1.0	1.0
tst4/task3	1.0	1.0	1.0
tst4/task4	1.0	1.0	1.0
tst4/task5	1.0	1.0	1.0

Table 4: *Results for team-2*

File	Precision @1	Precision @2	Precision @5
tst1/task1	0.983	0.996	0.999
tst1/task2	0.958	0.988	0.996
tst1/task3	1.0	1.0	1.0
tst1/task4	0.999	1.0	1.0
tst1/task5	0.999	1.0	1.0
tst2/task1	0.991	0.997	1.0
tst2/task2	0.965	0.99	0.997
tst2/task3	0.999	1.0	1.0
tst2/task4	0.999	1.0	1.0
tst2/task5	0.997	1.0	1.0
tst3/task1	0.778	0.829	0.934
tst3/task2	0.953	0.968	0.997
tst3/task3	1.0	1.0	1.0
tst3/task4	1.0	1.0	1.0
tst3/task5	0.933	0.953	0.986
tst4/task1	0.803	0.848	0.942
tst4/task2	0.954	0.977	0.995
tst4/task3	1.0	1.0	1.0
tst4/task4	0.998	1.0	1.0
tst4/task5	0.941	0.956	0.982

Table 5: *Results for team-3*

File	Precision @1	Precision @2	Precision @5
tst1/task1	0.704	0.829	0.96
tst1/task2	0.662	0.729	0.919
tst1/task3	0.595	0.632	0.762
tst1/task4	0.554	0.599	0.717
tst1/task5	0.579	0.67	0.798
tst2/task1	0.567	0.695	0.822
tst2/task2	0.651	0.706	0.851
tst2/task3	0.612	0.654	0.779
tst2/task4	0.578	0.627	0.751
tst2/task5	0.544	0.656	0.78
tst3/task1	0.603	0.707	0.873
tst3/task2	0.68	0.743	0.908
tst3/task3	0.595	0.646	0.763
tst3/task4	0.572	0.61	0.747
tst3/task5	0.552	0.65	0.82
tst4/task1	0.547	0.667	0.79
tst4/task2	0.666	0.716	0.863
tst4/task3	0.576	0.619	0.757
tst4/task4	0.565	0.599	0.731
tst4/task5	0.509	0.622	0.77

Table 6: *Results for team-5*

File	Precision @1	Precision @2	Precision @5
tst1/task1	0.934	0.987	0.999
tst1/task2	0.995	0.995	0.995
tst1/task3	0.73	0.87	1.0
tst1/task4	1.0	1.0	1.0
tst1/task5	0.751	0.829	0.94
tst2/task1	0.927	0.985	0.998
tst2/task2	0.994	0.995	0.998
tst2/task3	0.724	0.858	0.999
tst2/task4	1.0	1.0	1.0
tst2/task5	0.775	0.861	0.96
tst3/task1	0.618	0.715	0.896
tst3/task2	0.634	0.647	0.687
tst3/task3	0.726	0.888	0.999
tst3/task4	1.0	1.0	1.0
tst3/task5	0.661	0.742	0.85
tst4/task1	0.672	0.75	0.922
tst4/task2	0.634	0.661	0.763
tst4/task3	0.714	0.893	0.998
tst4/task4	1.0	1.0	1.0
tst4/task5	0.663	0.741	0.842

Table 7: *Results for team-6*

File	Precision @1	Precision @2	Precision @5
tst1/task1	0.998	0.998	1.0
tst1/task2	1.0	1.0	1.0
tst1/task3	0.991	0.997	1.0
tst1/task4	0.995	1.0	1.0
tst1/task5	0.986	0.995	1.0
tst2/task1	0.997	0.999	0.999
tst2/task2	1.0	1.0	1.0
tst2/task3	0.993	0.998	1.0
tst2/task4	0.997	1.0	1.0
tst2/task5	0.984	0.995	0.999
tst3/task1	0.946	0.966	0.977
tst3/task2	0.932	0.994	1.0
tst3/task3	0.993	0.999	1.0
tst3/task4	0.996	1.0	1.0
tst3/task5	0.928	0.953	0.955
tst4/task1	0.957	0.978	0.989
tst4/task2	0.928	0.994	1.0
tst4/task3	0.997	1.0	1.0
tst4/task4	0.996	1.0	1.0
tst4/task5	0.932	0.957	0.959

Table 8: *Results for team-7*

File	Precision @1	Precision @2	Precision @5
tst1/task1	0.724	0.763	0.923
tst1/task2	0.594	0.612	0.785
tst1/task3	0.534	0.68	0.848
tst1/task4	0.655	0.813	0.956
tst1/task5	0.421	0.601	0.79
tst2/task1	0.652	0.76	0.899
tst2/task2	0.616	0.636	0.743
tst2/task3	0.565	0.653	0.825
tst2/task4	0.625	0.727	0.875
tst2/task5	0.336	0.504	0.754
tst3/task1	0.643	0.797	0.941
tst3/task2	0.626	0.646	0.794
tst3/task3	0.549	0.681	0.854
tst3/task4	0.678	0.829	0.956
tst3/task5	0.362	0.564	0.832
tst4/task1	0.583	0.766	0.926
tst4/task2	0.628	0.639	0.778
tst4/task3	0.528	0.659	0.804
tst4/task4	0.633	0.738	0.875
tst4/task5	0.342	0.495	0.797

Table 9: *Results for team-8*

File	Precision @1	Precision @2	Precision @5
tst1/task1	1.0	1.0	1.0
tst1/task2	1.0	1.0	1.0
tst1/task3	0.998	0.998	0.998
tst1/task4	1.0	1.0	1.0
tst1/task5	0.986	0.987	0.993
tst2/task1	1.0	1.0	1.0
tst2/task2	1.0	1.0	1.0
tst2/task3	0.997	0.997	0.999
tst2/task4	1.0	1.0	1.0
tst2/task5	0.985	0.985	0.991
tst3/task1	0.866	0.874	0.924
tst3/task2	1.0	1.0	1.0
tst3/task3	1.0	1.0	1.0
tst3/task4	1.0	1.0	1.0
tst3/task5	0.957	0.962	0.977
tst4/task1	0.882	0.892	0.921
tst4/task2	1.0	1.0	1.0
tst4/task3	1.0	1.0	1.0
tst4/task4	1.0	1.0	1.0
tst4/task5	0.953	0.957	0.97

Table 10: *Results for team-9*

File	Precision @1	Precision @2	Precision @5
tst1/task1	0.167	0.258	0.614
tst1/task2	0.381	0.421	0.564
tst1/task3	0.539	0.582	0.715
tst1/task4	0.409	0.443	0.669
tst1/task5	0.358	0.47	0.7
tst2/task1	0.161	0.29	0.601
tst2/task2	0.383	0.422	0.557
tst2/task3	0.528	0.575	0.706
tst2/task4	0.428	0.465	0.659
tst2/task5	0.307	0.471	0.696
tst3/task1	0.205	0.367	0.725
tst3/task2	0.05	0.094	0.385
tst3/task3	0.529	0.571	0.678
tst3/task4	0.425	0.458	0.664
tst3/task5	0.288	0.436	0.742
tst4/task1	0.229	0.407	0.745
tst4/task2	0.049	0.127	0.432
tst4/task3	0.525	0.567	0.673
tst4/task4	0.453	0.479	0.672
tst4/task5	0.314	0.424	0.717

Table 11: *Results for team-10*