Dialog State Tracking Challenge 4

Pilot Task Guidelines v.3.0

http://www.colips.org/workshop/dstc4

Seokhwan Kim[1], Luis Fernando D'Haro[1], Rafael E. Banchs[1],
Jason Williams[2], and Matthew Henderson[3]

[1] Human Language Technologies, Institute for Infocomm Research (I2R - A*STAR)
One Fusionopolis Way, #21-01 Connexis (South Tower), Singapore 138632
[2] Microsoft Research, One Microsoft Way, Redmond, WA, USA - 98052-6399
[3] Google, 1-13 St Giles High St, London WC2H 8LG, United Kingdom

August 5, 2015

TABLE OF CONTENTS

1. Motivation 1
2. Data 1
3. Evaluation 4
4. Included Scripts and Tools 9
5. JSON Data Formats 12
6. Frequently Asked Questions (FAQ) 18
7. Subscription to mailing list 18
8. Committees 19
9. References 19

1. Motivation

Dialog state tracking is one of the key sub-tasks of dialog management, which defines the representation of dialog states and updates them at each moment on a given on-going conversation. To provide a common test bed for this task, the first Dialog State Tracking Challenge (DSTC) was organized[1] [1]. More recently, Dialog State Tracking Challenges 2 & 3 have been successfully completed[2] [2], [3].

In this fourth edition of the Dialog State Tracking Challenge, we will focus on a dialog state tracking task on human-human dialogs. In addition to this main task, we also propose a series of pilot tracks for the core components in developing end-to-end dialog systems based on the same dataset. More specifically, four pilot tasks are available in DSTC4. These pilot tasks are optional for all participants in the challenge.

- **Spoken language understanding (SLU):** The objective of this task is to tag a given utterance (either from the tourist or the tour guide) with speech acts and semantic slots.
- **Speech act prediction (SAP):** The objective of this task is to predict the speech act of the next turn imitating the policy of one speaker (either the tourist or the tour guide).
- **Spoken language generation (SLG):** The objective of this task is to generate a response utterance for one of the participants (either the tourist or the tour guide) by using the corresponding speech act and semantic slot information.
- **End-to-end system (EES):** The objective of this task is to develop an end-to-end system playing the part of a guide or a tourist by pipelining and/or combining different SLU, SAP and SLG systems.

This document provides a comprehensive description about the pilot tasks in DSTC4. The rest of the document is structured as follows. In section 2 the data used during the challenge for the pilot tasks is described. Examples of the dialogs included are also given. In section 3, the evaluation metrics and format for the pilot task submissions is provided. Then, in section 4, description of several tools included with the data is provided. These tools are intended to allow participants to check the data, as well as to have a baseline system that participants can modify or combine with their proposed systems. Finally in section 5, the JSON data formats used for the annotations are described in detail.

2. Data

2.1. *General Characteristics*

In this challenge, participants will use TourSG corpus to develop the components. TourSG consists of 35 dialog sessions on touristic information for Singapore collected from Skype calls between three tour guides and 35 tourists. These 35 dialogs sum up to 31,034 utterances and 273,580 words. All the recorded dialogs, with a total length of 21 hours, have been manually transcribed and annotated with speech act and semantic labels for each turn level.

Different from the main task, in which dialog states are defined at the sub-dialog level and each of the sub-dialogs has a frame structure with slot value pairs to represent the subject discussed within it; in the pilot tasks, annotations are provided at the utterance level and, accordingly, systems must deal with slot value pairs at the utterance level. Annotations at the utterance level involve both, semantic slots and speech acts.

[1] http://research.microsoft.com/en-us/events/dstc/
[2] http://camdial.org/~mh521/dstc/

2.2. *Example of Dialog Annotations for the Main Task*

Tables 1 and 2 present two examples of annotations at the utterance level for both semantic tags and speech acts. Notice that in the TourSG dialogue dataset more than one speech act per utterance can occur.

Speaker	Semantic Tagged Utterance	Speech Act (Attribute)
Tourist	Can you give me some uh- tell me some **<DET CAT="PRICE">** cheap rate **</DET>** **<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL">** hotels **</LOC>**,	QST (RECOMMEND)
	because I'm planning just to leave my bags there and go somewhere take some pictures.	INF (EXPLAIN)
Guide	Okay.	FOL (ACK)
	I'm going to recommend firstly you want to have a **<DET CAT="MAIN">** backpack type **</DET>** of **<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL">** hotel **</LOC>**,	INI (RECOMMEND)
	right?	QST (PREFERENCE)
Tourist	Yes.	RES (POSITIVE)
	I'm just gonna bring my backpack and my buddy with me. So I'm kinda looking for a hotel that is **<DET CAT="PRICE">** not that expensive **</DET>**. Just gonna leave our things **<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL">** there **</LOC>** and, you know, stay out the whole day.	RES (PREFERENCE| EXPLAIN)
Guide	Okay. Let me get you hm hm.	FOL (ACK)
	So you don't mind if it's a bit uh **<DET CAT="MAIN">** not so roomy **</DET>** like hotel because you just back to sleep.	QST (PREFERENCE)
Tourist	Yes. Yes.	RES (POSITIVE)
	As we just gonna put our things **<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL">** there **</LOC>** and then go out to take some pictures.	RES (PREFERENCE| EXPLAIN)
Guide	Okay, um-	FOL (ACK)
Tourist	Hm.	-

Table 1. Example of utterance level annotations for sub-dialog segment #1

Speaker	Transcription	Annotation
Guide	Let's try **<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL">**this one**</LOC>**, okay?	INI (RECOMMEND)
Tourist	Okay.	FOL (ACK)
Guide	It's **<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL">** InnCrowd Backpackers Hostel **</LOC>** in **<AREA FROM-TO="NONE" REL="NONE" CAT="CITY">** Singapore **</AREA>**. If you take **<DET CAT="MAIN">** a dorm bed **</DET>** per person only **<FEE CAT="MAIN">** twenty dollars **</FEE>**. If you take **<DET CAT="MAIN">** a room **</DET>**, it's **<DET CAT="MAIN">** two single beds **</DET>** at **<FEE CAT="MAIN">** fifty nine dollars **</FEE>**.	INI (RECOMMEND) FOL (INFO|HOW_MUCH)
Tourist	Um. Wow, that's good.	FOL (POSITIVE)
Guide	Yah, **<FEE CAT="MAIN">** prices **</FEE>** are based on **<DET CAT="MAIN">** per person per bed or dorm **</DET>**. But this one is room. So it should be **<FEE CAT="MAIN">** fifty nine **</FEE>** for the **<DET CAT="MAIN">** two room **</DET>**. So you're actually paying about **<FEE CAT="MAIN">** ten dollars **</FEE>** more **<DET CAT="MAIN">** per person **</DET>** only.	FOL (INFO|HOW_MUCH)
Tourist	Oh okay. That's- **<FEE CAT="MAIN">** the price **</FEE>** is reasonable actually. It's good.	FOL (ACK) FOL (POSITIVE)

Table 2. Example of utterance level annotations for a sub-dialog segment #2

2.3. *Data Available for DSTC4*

For the purposes of the pilot tasks in the DSTC4 Challenge, the TourSG corpus has been divided in the following three parts:

1. **Train data:** manual annotations at the utterance levels will be provided for 14 dialogs (7 from tour guide-1 and 7 from tour guide-2) for training the different pilot task systems.
2. **Dev data:** similar to the training data. In this case, 6 dialogues (3 from tour guide-1 and 3 from tour guide-2) for optimizing the pilot task systems.
3. **Test data:** manual transcriptions will be provided for 6 dialogs (2 from tour guide-1, 2 from tour guide-2 and 2 from tour guide-3) for evaluating the trackers.

The three datasets will be released free of charge to all registered challenge participants after signing a license agreement with ETPL-A*STAR. The dataset will include transcribed and annotated dialogs, as well as ontology objects describing the annotations.

3. Evaluation

Eight different subtasks will be evaluated for the case of pilot tasks at DSTC4. Next, a comprehensive description for pilot tasks evaluation is provided.

3.1. *Evaluation modality for pilot tasks*

Regarding operational aspects of pilot tasks evaluation, a web-service (WS) implementation is required for a pilot task system to be evaluated. In this modality, participant teams are required to run their systems under a web-service architecture. The evaluation will be conducted by a master evaluation script which will be calling the corresponding web-services at specified times slots during the evaluation dates. ***More details on web-service operation will be made available at the beginning of July in an updated version of this report.***

3.2. *Subtasks to be considered under the pilot tasks*

As the TourSG corpus constitutes a collection of conversations between two specific roles: a tour guide and a tourist, pilot tasks are to be focalized in modeling one of the two interlocutor roles of the TourSG dataset. In this sense, each pilot task has two primary subtasks, one related to the modeling of the tour guide and the other related to the modeling of the tourist. Each subtask can be better defined in terms of the input data the system should use and the output data it should produce.

- SLU-TOURIST subtask: in this case the input to the systems will be the utterances from both the tourist and the guide, and the system must produce both semantic tags (slot values) and speech acts for the tourist utterances only. If we consider the first four utterances of the sample dialogue presented in table 1, system inputs and outputs for SLU-TOURIST subtask should be as follows:

INPUT	TOURIST: Can you give me some uh- tell me some cheap rate hotels, because I'm planning just to leave my bags there and go somewhere take some pictures.
OUTPUT	SLOTS: **<DET CAT="PRICE">** cheap rate **</DET>**, **<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL">** hotels **</LOC>**
	SPEECH_ACTS: **QST (RECOMMEND) , INF (EXPLAIN)**

INPUT GUIDE: Okay. I'm going to recommend firstly you want to have a backpack type of hotel, right?

OUTPUT -

INPUT TOURIST: Yes. I'm just gonna bring my backpack and my buddy with me. So I'm kinda looking for a hotel that is not that expensive. Just gonna leave our things there and, you know, stay out the whole day.

OUTPUT SLOTS: **<DET CAT="PRICE">** not that expensive **</DET>**
<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL"> there **</LOC>**

SPEECH_ACTS: **RES (POSITIVE) , RES (PREFERENCE| EXPLAIN)**

INPUT GUIDE: Okay. Let me get you hm hm. So you don't mind if it's a bit uh not so roomy like hotel because you just back to sleep.

OUTPUT -

- SLU-GUIDE subtask: in this case the input to the systems will be the utterances from both the tourist and the guide, and the system must produce both semantic tags (slot values) and speech acts for the guide utterances only. Again, if we consider the first four utterances of the sample dialogue presented in table 1, system inputs and outputs for SLU-GUIDE subtask should be as follows:

INPUT TOURIST: Can you give me some uh- tell me some cheap rate hotels, because I'm planning just to leave my bags there and go somewhere take some pictures.

OUTPUT -

INPUT GUIDE: Okay. I'm going to recommend firstly you want to have a backpack type of hotel, right?

OUTPUT SLOTS: **<DET CAT="MAIN">** backpack type **</DET>**
<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL"> hotel **</LOC>**

SPEECH_ACTS: **FOL (ACK)** , **INI (RECOMMEND)** , **QST (PREFERENCE)**

INPUT TOURIST: Yes. I'm just gonna bring my backpack and my buddy with me. So I'm kinda looking for a hotel that is not that expensive. Just gonna leave our things there and, you know, stay out the whole day.

OUTPUT -

INPUT GUIDE: Okay. Let me get you hm hm. So you don't mind if it's a bit uh not so roomy like hotel because you just back to sleep.

OUTPUT SLOTS: **<DET CAT="MAIN">** not so roomy **</DET>**

SPEECH_ACTS: **FOL (ACK) , QST (PREFERENCE)**

- SAP-TOURIST subtask: in this case the input to the systems will be the utterances and annotations (semantic tags and speech acts) from the guide along with the resulting semantic tags for the next tourist utterances, and the system must produce the speech acts for the corresponding tourist utterances. Considering the sample dialogue presented in table 1, inputs and outputs for SAP-TOURIST subtask should be as follows:

INPUT GUIDE UTTERANCE: Okay. I'm going to recommend firstly you want to have a backpack type of hotel, right?

GUIDE SLOTS: **<DET CAT="MAIN">** backpack type **</DET>**
<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL"> hotel **</LOC>**

GUIDE SPEECH_ACTS: **FOL (ACK) , INI (RECOMMEND) , QST (PREFERENCE)**

TOURIST SLOTS: **<DET CAT="PRICE">** not that expensive **</DET>**
<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL"> there **</LOC>**

OUTPUT TOURIST SPEECH_ACTS: **RES (POSITIVE) , RES (PREFERENCE| EXPLAIN)**

INPUT GUIDE UTTERANCE: Okay. Let me get you hm hm. So you don't mind if it's a bit uh not so roomy like hotel because you just back to sleep.

GUIDE SLOTS: **<DET CAT="MAIN">** not so roomy **</DET>**

GUIDE SPEECH_ACTS: **FOL (ACK) , QST (PREFERENCE)**

TOURIST SLOTS: **<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL">**there**</LOC>**

OUTPUT TOURIST SPEECH_ACTS: **RES (POSITIVE) , RES (PREFERENCE| EXPLAIN)**

- SAP-GUIDE subtask: in this case the input to the systems will be the utterances and annotations (semantic tags and speech acts) from the tourist along with the resulting semantic tags for the next guide utterances, and the system must produce the speech acts for the corresponding guide utterances. Inputs and outputs for SAP-GUIDE subtask should be as follows:

INPUT TOURIST UTTERANCE: Can you give me some uh- tell me some cheap rate hotels, because I'm planning just to leave my bags there and go somewhere take some pictures.

TOURIST SLOTS: **<DET CAT="PRICE">** cheap rate **</DET>,**
<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL"> hotels **</LOC>**

TOURIST SPEECH_ACTS: **QST (RECOMMEND) , INF (EXPLAIN)**

GUIDE SLOTS: **<DET CAT="MAIN">** backpack type **</DET>,**
<**LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL">** hotel **</LOC**>

OUTPUT GUIDE SPEECH_ACTS: **FOL (ACK) , INI (RECOMMEND) , QST (PREFERENCE)**

INPUT TOURIST UTTERANCE: Yes. I'm just gonna bring my backpack and my buddy with me. So I'm kinda looking for a hotel that is not that expensive. Just gonna leave our things there and, you know, stay out the whole day.

TOURSIT SLOTS: **<DET CAT="PRICE">** not that expensive **</DET>,**
<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL"> there **</LOC>**

TOURIST SPEECH_ACTS: **RES (POSITIVE) , RES (PREFERENCE| EXPLAIN)**

GUIDE SLOTS: **<DET CAT="MAIN">** not so roomy **</DET>**

OUTPUT GUIDE SPEECH_ACTS: **FOL (ACK) , QST (PREFERENCE)**

- SLG-TOURIST subtask: in this case the input to the systems will be the semantic tags and speech acts from the tourist only, and the system must produce the final surface form for the tourist utterances. Inputs and outputs for SLG-TOURIST subtask should be as follows:

INPUT SLOTS: **<DET CAT="PRICE">** cheap rate **</DET>,**
<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL"> hotels **</LOC>**

SPEECH_ACTS: **QST (RECOMMEND) , INF (EXPLAIN)**

OUTPUT TOURIST UTTERANCE: Can you give me some uh- tell me some cheap rate hotels, because I'm planning just to leave my bags there and go somewhere take some pictures.

INPUT SLOTS: **<DET CAT="PRICE">** not that expensive **</DET>**
<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL"> there **</LOC>**

SPEECH_ACTS: **RES (POSITIVE) , RES (PREFERENCE| EXPLAIN)**

OUTPUT TOURIST UTTERANCE: Yes. I'm just gonna bring my backpack and my buddy with me. So I'm kinda looking for a hotel that is not that expensive. Just gonna leave our things there and, you know, stay out the whole day.

- SLG-GUIDE subtask: in this case the input to the systems will be the semantic tags and speech acts from the guide only, and the system must produce the final surface form for the guide utterances. Inputs and outputs for SLG-GUIDE subtask should be as follows:

INPUT SLOTS: **<DET CAT="MAIN">** backpack type **</DET>**
<LOC FROM-TO="NONE" REL="NONE" CAT="HOTEL"> hotel **</LOC>**

SPEECH_ACTS: **FOL (ACK)** , **INI (RECOMMEND)** , **QST (PREFERENCE)**

OUTPUT GUIDE UTTERANCE: Okay. I'm going to recommend firstly you want to have a backpack type of hotel, right?

INPUT SLOTS: **<DET CAT="MAIN">** not so roomy **</DET>**

SPEECH_ACTS: **FOL (ACK) , QST (PREFERENCE)**

OUTPUT GUIDE UTTERANCE: Okay. Let me get you hm hm. So you don't mind if it's a bit uh not so roomy like hotel because you just back to sleep.

- EES-TOURIST subtask: in this case the objective is to deploy a system able to model the tourist behavior in the dialogues. The input to the systems will be the guide utterances and the system must produce the tourist utterances. Inputs and outputs for EES-TOURIST subtask should be as follows:

INPUT GUIDE UTTERANCE: Okay. I'm going to recommend firstly you want to have a backpack type of hotel, right?

OUTPUT TOURIST UTTERANCE: Yes. I'm just gonna bring my backpack and my buddy with me. So I'm kinda looking for a hotel that is not that expensive. Just gonna leave our things there and, you know, stay out the whole day.

INPUT GUIDE UTTERANCE: Okay. Let me get you hm hm. So you don't mind if it's a bit uh not so roomy like hotel because you just back to sleep.

OUTPUT TOURIST UTTERANCE: Yes. Yes. As we just gonna put our things there and then go out to take some pictures.

- EES-GUIDE subtask: in this case the objective is to deploy a system able to model the tour guide behavior in the dialogues. The input to the systems will be the tourist utterances and the system must produce the guide utterances. Inputs and outputs for EES-GUIDE subtask should be as follows:

  INPUT TOURIST UTTERANCE: Can you give me some uh- tell me some cheap rate hotels, because I'm planning just to leave my bags there and go somewhere take some pictures.

  OUTPUT GUIDE UTTERANCE: Okay. I'm going to recommend firstly you want to have a backpack type of hotel, right?

  INPUT TOURIST UTTERANCE: Yes. I'm just gonna bring my backpack and my buddy with me. So I'm kinda looking for a hotel that is not that expensive. Just gonna leave our things there and, you know, stay out the whole day.

  OUTPUT GUIDE UTTERANCE: Okay. Let me get you hm hm. So you don't mind if it's a bit uh not so roomy like hotel because you just back to sleep.

Finally, for all subtasks in DSTC4 pilot tasks, systems should always operate on current and/or past dialogue inputs only. No future dialogue information contained within the evaluation sets should be used by the systems when processing a given utterance.

3.3. *Evaluation metrics to be used for pilot tasks*

Two different families of metrics will be used for evaluating the pilot tasks: classification accuracy metrics such as Precision and Recall [3] will be used for subtasks related to SLU and SAP tasks, and semantic similarity metrics such as BLEU [4] and AM-FM [5] will be used for subtasks related to SLG and EES tasks.

Each pilot task system should generate the corresponding output for every utterance in a given log file described in Section 5.1. As mentioned already at the end of the previous subsection, while all the transcriptions and segment details provided in the log object from the beginning of the session to the current turn can be used, any information from the future turns are not allowed to be considered to produce system outputs at a given turn. For pilot task subtasks, evaluation schedule 1 will be used (i.e. system outputs are evaluated at all turns).

In DSTC4, the following evaluation metrics are used for the pilot tasks:

- SLU and SAP tasks:
  - Precision: Fraction of semantic tags and/or speech acts that are correctly generated
  - Recall: Fraction of semantic tags and/or speech acts in the gold standard that are correctly generated
  - F-measure: The harmonic mean of precision and recall

- SLG and EES tasks:
  - BLEU: Geometric average of n-gram precision (for n = 1, 2, 3, 4) of the system generated utterance with respect to reference utterance.
  - AM-FM: Weighted mean of (1) the cosine similarity between the system generated utterance and the reference utterance and (2) the normalized n-gram probability of the system generated utterance.

3.4. *Summary of evaluation strategy for pilot tasks at DSTC4*

The overall evaluation strategy for pilot tasks in DSTC4 is summarized in table 3.

Task	Subtask	Input	Output	Metrics
SLU	tourist	Utterances from both tourist and guide	Semantic tags and speech acts for tourist utterances	Precision / Recall
SLU	guide	Utterances from both tourist and guide	Semantic tags and speech acts for guide utterances	Precision / Recall
SAP	tourist	Semantic tags and speech acts from guide and semantic tags from tourist	Next speech act(s) from tourist	Precision / Recall
SAP	guide	Semantic tags and speech acts from tourist and semantic tags from guide	Next speech act(s) from guide	Precision / Recall
SLG	tourist	Semantic tags and speech acts from tourist only	Surface form of tourist utterance	BLEU / AM-FM
SLG	guide	Semantic tags and speech acts from guide only	Surface form of guide utterance	BLEU / AM-FM
EES	tourist	Guide utterances and semantic tags for tourist	Surface form of tourist utterance	BLEU / AM-FM
EES	guide	Tourist utterances and semantic tags for guide	Surface form of guide utterance	BLEU / AM-FM

Table 3. Evaluation strategy for pilot tasks at DSTC4.

4. Included Scripts and Tools

As in the previous DSTC 2 and 3 evaluations, the DSTC 4 evaluation includes a set of useful scripts and tools for dealing with the provided data. Below a brief description of the available tools is provided.

4.1. *Baseline Systems*

There is no plan to provide baseline systems for the different pilot tasks in DSTC4. If you are interested in contributing to DSTC4 by making available to other participants a baseline system for any of the pilot tasks, please contact DSTC4 the organizers: http://www.colips.org/workshop/dstc4/contact.html

4.2. *Evaluation Scripts*

One evaluation script will be made available for evaluating the outputs of the pilot tasks: *pilot.py*. This section serves as an introduction to using the evaluation script. The output of your system should be provided in a file *output.json* following the data structure described in section 5.3. You also should have a scripts directory with a config directory within it. The config directory contains the definitions of the datasets, e.g. *dstc4_dev.flist* which enumerates the sessions in the development set of DSTC 4.

The structure and contents of your system output file can be checked using *check_pilot.py*:

```
python scripts/check_pilot.py --dataset dstc4_dev --dataroot data --ontology
scripts/config/ontology_dstc4.json --pilotfile output.json --pilottask SLU|SAP|
SLG|EES --roletype GUIDE|TOURIST
```

This should output “Found no errors, taskfile is valid”.

For running the evaluations scripts, it is necessary to install Python 2.7 and some modules which are specified below. Most of these modules can be automatically downloaded and installed using pip (https://pip.pypa.io/en/latest/installing.html).

- Scikit-learn (http://scikit-learn.org/)
  - On a command line type: `sudo pip install scikit-learn`
  - For Windows: `C:\Python2.7\scripts\pip.exe install scikit-learn`
- Numpy (http://www.numpy.org/)
  - On a command line type: `sudo pip install numpy`
  - For Windows: `C:\Python2.7\scripts\pip.exe install numpy`

Then, the evaluation script, *pilot.py* can be run on the system output as follows:

```
python scripts/pilot.py --pilottask SLU|SAP|SLG|EES --roletype GUIDE|TOURIST
--dataset dstc4_dev --dataroot data --trackfile output.json --scorefile output.
score.csv
```

This creates a file *output.score.csv* which lists all the metrics. Then *report_pilot.py* can be used to format these results:

```
python scripts/report_pilot.py --scorefile output.score.csv
```

4.3. *Server-Client Scripts*

In order to run the test evaluations for the pilot tasks a server-client architecture is proposed. The server is setup by each team participating in this pilot tasks and it will be called by a client setup by the organizers on a given time slot in order to get answers for the different tasks and user roles for which the team is interested on participating.

During the real evaluation, the server will receive a JSON object containing the input parameters required for the given task and role (see examples in section3.2) and the server will use the input parameters to generate a corresponding answer that is send back to the client using a JSON message. Then, based on the retrieved result, the client will calculate the actual values for the proposed metrics explained in section 3.3.

In order to facilitate the evaluation of the pilot tasks, the organizers will provide both the server and client as python scripts that are configured by default to be used with the development set so each team can check that the systems are working and if the system is reachable from outside. Below, some details and requirements about the two programs are explained.

4.3.1. *Server:*

The server uses the Tornado[3] framework which can be installed using the command: pip install tornado. The default port is 8080 although it can be modified from the command line using the option –port. The server includes the logging module to keep record of all the requests and answers given by the server both by using the

[3] http://www.tornadoweb.org/en/stable/

stdout and on file. Three important issues must be considered when working with the server.

1. As mentioned before, the server is configured to work by default with the dstc4_dev set and therefore some pre-processing is required to read the file and to concatenate consecutive turns by the same user and convert them into one single utterance (this is done using the function fntLoadInfoSubmissionDemo). However, during the testing phase this function is not needed anymore and must be commented out.

2. The server is configured to provide the correct answer from the dstc4_dev file, therefore all the metric results will give the maximum score of 1.0. However, for the testing phase, the teams must modify the lines where the answer is provided with their own calls to actual functions to generate the answer. In case the teams are using python tools then it should be straightforward to import the programs and make the corresponding calls. In case it is not, each team will need to implement their own strategy to send the input arguments to their system and retrieve the output information. Here, the organizers recommend the use of the module subprocess[4] which allows for calling external tools using the operative system command line.

3. The server makes use of JSON messages to communicate with the client. Take a look to section 5.5 to check the format of the exchanging messages between both systems. This is especially relevant in case a team wants to make a different implementation that can be easily integrated with their systems.

Once, the server is setup and tested, each team will need to provide to the organizers the URL and port used. This data will be used for the organizers client during the testing phase.

4.3.2. *Client:*

The provided client resembles the system that the organizers will use during the test phase. This client is provided to the teams for debugging purposes so each team can check that their servers are correctly setup in terms of response and accessibility. The client uses a websocket connection using the module websocket-client[5] that can be installed using the command: pip install websocket-client. The internal process of the client can be divided into two parts:

1. Reading of the evaluation set (including both the label.json and log.json files) and unification of consecutive user's turns. This process is done to make easy the evaluation and preparation of the input data required for the server in order to perform each task.

2. Request of the information to the servers for each task and user's role. Here, the system prepares the input information for each case and send a JSON message to the server (see section 5.5 for details about the content and format of this message). Then, the answer is recorded in an internal structure that is used later to estimate the final performance of each team.

4.4. ***Other Tools***

There are a few other scripts included which may be of use for participants:

- **dataset_walker.py:** A Python script which makes it easy to iterate through a dataset specified by file list (.flist) in scripts/config. When the script is called without arguments it outputs the content of all the training data on the terminal. In case you want to check the content of the development data, you need to modify the parameter value from dstc4_train to dstc4_dev.

- **ontology_reader.py**: A Python script which makes it easy to get the information from the ontology.

[4] https://docs.python.org/2/library/subprocess.html
[5] https://github.com/liris/websocket-client

5. JSON Data Formats

The datasets are distributed as collections of dialogs, where each dialog has a log.json file containing a Log object in JSON, and possibly a label.json containing a Label object in JSON representing the annotations. Also distributed with the data is an Ontology JSON object, which describes the ontology/domain of the sessions. The below sections describe the structure of the Log, Label and Ontology objects.

5.1. *Log Objects*

The *log.json* file includes the information for each session between a given tourist and a given guide. The JSON files were generated following below the specification:

- session_id: a unique ID for this session (integer)
- session_date: the date of the call, in yyyy-mm-dd format (string)
- session_time: the time the call was started, in hh:mm:ss format (string)
- guide_id: a unique ID for the guide participated in this session (string)
- tourist_id: a unique ID for the tourist participated in this session (string)
- tourist_age: the age of the tourist (integer)
- tourist_sex: the gender of the tourist (string: "F"/"M")
- tourist_visited_sg: whether the tourist has visited or not Singapore in the past (string: "Y"/"N")
- utterances: [
    - utter_index: the index of this utterances in the session starting at 0 (integer)
    - speaker: the speaker of this utterance (string: "GUIDE"/"TOURIST")
    - transcript: the transcribed text of this utterance (string). Filler disfluencies in the recorded utterance are annotated with preceding percent sign (%) like "%ah", "%eh", "%uh", or "%um".
    - segment_info: [
        - topic: the topic category of the dialog segment that this utterance belongs to (string: "OPENING" / "CLOSING" / "ITINERARY" / "ACCOMMODATION" / "ATTRACTION" / "FOOD" / "SHOPPING" / "TRANSPORTATION")
        - target_bio: the indicator with BIO scheme whether this utterance belongs to a segment considered as a target for the main task or not. The value for this key should be 'B' if this utterance is located at the beginning of a target session or 'I' if the utterance is not at the beginning but inside the target session. Otherwise, it is assigned to 'O'. (string: "B"/"I"/"O")
        - guide_act: the dialog act of the guide through this segment (string: "QST" / "ANS" / "REQ" / "REQ_ALT" / "EXPLAIN" / "RECOMMEND" / "ACK" / "NONE")
        - tourist_act: the dialog act of the tourist through this segment (string: "QST" / "ANS" / "REQ" / "REQ_ALT" / "EXPLAIN" / "RECOMMEND" / "ACK" / "NONE")
        - initiativity: whether this segment is initiated by the guide or the tourist (string: "GUIDE" / "TOURIST")

      ]

  ]

5.2. ***Label Objects***

The annotations for each segment are given in the *label.json* file. The json object in the label file consists of three different types of labels: frame structures for the main task and speech acts and semantics for the other pilot tasks. Below is the specification of the object:

- session_id: a unique ID for this session (integer)
- utterances: [
    - utter_index: a unique ID for this utterance (integer)
    - frame_label
        - SLOT: [list of values (string)]
    - speech_act: [
        - act: speech act category (string)
        - attributes: [list of attributes (string)]

      ]
    - semantic_tagged: [list of tagged utterances (string)]

  ]

5.2.1. *Frame labels*

The gold standard frame structure for the dialog segment that the current utterance belongs to is given as the object value of the 'frame_label' key. Each object consists of a set of a slot and a list of values pairs defined for the topic category of a given segment. The detailed definitions of the frame structures can be found in the ontology object. Notice this information is useful for the main track of DSTC4 and no relevant for the pilot tasks.

5.2.2. *Speech acts*

Since the speech acts were originally analyzed for each sub-utterance unit divided based on the pauses in the recordings and then combined into the full utterance level, each utterance could have more than one speech act objects if it was generated by concatenating its multiple sub-utterances. Thus, a list of speech act annotations is taken as the value for the 'speech_act' key of a given utterance.

Each object has two types of information: speech act category and attributes. Every sub-utterance should belong to one of the four basic speech act categories that denote the general role of the utterance in the current dialog flow. More specific speech act information can be annotated by combination with attributes. By contrast to act category, there's no constraint on the number of attributes for a single utterance. Thus, a sub-utterance can have no attribute or more than one attributes in the list object. Below are the list of speech act categories and attributes with their descriptions.

- Speech act categories
    - QST (QUESTION) used to identify utterances that pose either a question or a request
    - RES (RESPONSE) used to identify utterances that answer to a previous question or a previous request
    - INI (INITIATIVE) used to identify utterances that constitute new initiative in the dialog, which does not constitute either a question, request, answer or follow up action to a previous utterance
    - FOL (FOLLOW) a response to a previous utterance that is not either a question or a request
- Speech act attributes

- ACK: used to indicate acknowledgment, as well as common expressions used for grounding
- CANCEL: used to indicate cancelation
- CLOSING: used to indicate closing remarks
- COMMIT: used to identify commitment
- CONFIRM: used to indicate confirmation
- ENOUGH: used to indicate/request that no more information is needed
- EXPLAIN: used to indicate/request an explanation/justification of a previous stated idea
- HOW_MUCH: used to indicate money or time amounts
- HOW_TO: used to request/give specific instructions
- INFO: used to indicate information request
- NEGATIVE: used to indicate negative responses
- OPENING: used to indicate, opening remarks
- POSITIVE: used to indicate positive responses
- PREFERENCE: used to indicate/request preferences
- RECOMMEND: used to indicate/request recommendations
- THANK: used to indicate thank you remarks
- WHAT: used to indicate concept related utterances
- WHEN: used to indicate time related utterances
- WHERE used to indicate location related utterances
- WHICH: used to indicate entity related utterances
- WHO: used to indicate person related utterances and questions

5.2.3. *Semantic tags*

Similarly to speech acts, semantic tags were annotated for at sub-utterance level. Thus it takes a list of tagged sub-utterances as its value, and the number of items in the list should be the same with the one for speech acts. We defined below the main categories for semantic annotation:

- AREA: It refers to a geographic area but not a specific spot or location
- DET: It refers to tourist's criteria used or reasons why the tourist would like to decide spot.
- FEE: It refers to admission fees, price of services or any other fare.
- FOOD: It refers to any type of food or drinks.
- LOC: It refers to specific touristic spots or commerce/services locations.
- TIME: It refers to time, terms, dates, etc.
- TRSP. It refers to expressions related to transportation and transportation services.
- WEATHER: It refers to any expression related to weather conditions.

Some of them include also subcategories, relative modifiers and from-to modifiers (Table 4).

MAIN	SUBCAT	REL	FROM-TO
AREA	COUNTRY, CITY, DISTRICT, NEIGHBORHOOD	NEAR, FAR, NEXT, OPPOSITE, NORTH, SOUTH, EAST, WEST	FROM, TO
DET	ACCESS, BELIEF, BUILDING, EVENT, PRICE, NATURE, HISTORY, MEAL, MONUMENT, STROLL, VIEW	-	-

FEE	ATTRACTION, SERVICES, PRODUCTS	-	-
FOOD	-	-	-
LOC	TEMPLE, RESTAURANT, SHOP, CULTURAL, GARDEN, ATTRACTION, HOTEL, WATERSIDE, EDUCATION, ROAD, AIRPORT	NEAR, FAR, NEXT, OPPOSITE, NORTH, SOUTH, EAST, WEST	FROM, TO
TIME	DATE, INTERVAL, START, END, OPEN, CLOSE	BEFORE, AFTER, AROUND	-
TRSP	STATION, TYPE	NEAR, FAR, NEXT, OPPOSITE, NORTH, SOUTH, EAST, WEST	FROM, TO
WEATHER	-	-	-

Table 4. List of Categories and Modifiers for Semantic Annotations

The semantic tags and their categories are indicated as follows:

- <MAIN CAT="SUBCAT" REL="REL" FROM-TO="FROM_TO"> at the beginning of the identified word or compound
- </TAG> at the end of the identified word or compound

When either no specific subcategory exists for a given semantic tag or it is not possible to select among the available subcategories, the 'CAT' field is assigned to cat="MAIN".

5.3. *Pilot Task Output Objects*

Pilot tasks outputs for offline evaluation should be organized following below the JSON specification:

- dataset: the name of the dataset over which the tracker has been run (string)
- wall_time: the time in seconds it took to run the tracker (float)
- sessions: a list of results corresponding to each session in the dataset [
    - session_id: the unique ID of this session (integer)
    - utterances: [
        - utter_index: a unique ID for this utterance (integer)
        - task_type: specifies the task: 0 SLU, 1 SPA, 2 SLG or 3 EES (integer)
        - role_type: specifies the subtask: 0 for the tour guide or 1 for the tourist (integer)
        - speech_act: [
            - act: speech act category (string)
            - attributes: [list of attributes (string)]

          ]
        - semantic_tagged: tagged utterance (string)
        - generated sentence: the final surface form generated by the system (string)

      ]

The expected format for the speech_act is the same as the ones in the reference label objects. However, and for the case of the semantic_tagged elements a single tagged utterance (instead of a list of tagged sub-utterances) will suffix. Notice that although the described object will be used to report outputs from all four pilot tasks, systems for the different pilot tasks will produce different elements of this output object. Those elements that are not supposed to be produced by a given system are ignored by the evaluation script.

5.4. *Ontology Object*

The ontology object in *ontology_dstc4.json* consists of tag sets for the pilot tasks. The ontology object is provided in the following format:

- **pilot_tagsets**
    - **speech_act**
        - **category: [list of speech act categories]**
        - **attributes: [list of attributes]**
    - **semantic**
        - **MAIN**
            - **CAT: [list of subcategories in the MAIN category]**
            - **REL: [list of relative modifiers for the MAIN category]**
            - **FROM-TO: [list of from-to modifiers for the MAIN category]**
- tagsets
    - TOPIC
        - SLOT: [list of possible values]
- knowledge
    - MRT_LINE
        - CODE: string
        - NAME: string
        - COLOR: string
    - SHOPPING
        - NAME: string
        - TYPE_OF_PLACE: [list of shopping place types]
    - RESTAURANT
        - NAME: string
        - TYPE_OF_PLACE: [list of restaurant types]
        - NEIGHBOURHOOD: [list of neighbourhoods]
        - CUISINE: [list of cuisines]
        - PRICERANGE: integer (from 1 to 5)
    - FOOD
        - NAME: string
        - CUISINE: string
    - MRT_STATION
        - NAME: string
        - CODE: [list of codes]
        - NEIGHBOURHOOD: [list of neighbourhoods]
    - HOTEL
        - NAME: string
        - TYPE_OF_PLACE: [list of hotel types]

        - NEIGHBOURTHOOD: [list of neighbourhoods]
        - RATING: integer (from 1 to 5)
        - PRICERANGE: integer (from 1 to 5)
    - ATTRACTION
        - NAME: string
        - TYPE_OF_PLACE: [list of attraction types]
        - NEIGHBOURHOOD: [list of neighbourhoods]
    - ROAD
        - NAME: string
        - NEIGHBOURHOOD: [list of neighbourhoods]
    - NEIGHBOURHOOD
        - REGION: string
        - DISTRICT: string
        - SUBDISTRICT: string

5.5. *Server-Client Object*

As commented in section 4.3, the server-client architecture proposed to evaluate the online systems during the testing phase requires the interchange of a JSON message for each user utterance and task in the test dataset. The JSON message is based on the structure presented in section 5.3.

- dataset: the name of the dataset over which the tracker has been run (string)
- wall_time: the time in seconds it took to run the tracker (float)
- session_id: the unique ID of this session (integer)
- utter_index: a unique ID for this utterance (integer). Take notice that this value not necessarily resemble the one in the JSON log and label files since there is first a process of concatenating utterances spoken by the same user but split in different turns.
- role_task: The role being evaluated, i.e. GUIDE or TOURIST (string)
- role_type: The role of the current utterance. In general this value is the same as the role_task field except for the SLU task where it is expected that the server only provides relevant information for agreement between the role_task and role_type, but even though the server could require saving information about the other user utterance.
- task_type: the task: SLU, SPA, SLG or EES (string)
- transcript: the concatenated surface form for the current user (string)
- speech_act: [
    - act: speech act category (string)
    - attributes: [list of attributes (string)]

  ]
- current_semantic_tagged: tagged utterance for the current user (string)
- next_semantic_tagged: tagged utterance for the next user (string)

6. Frequently Asked Questions (FAQ)

6.1. *Do I or my company need to pay a license fee for getting the TourSG dataset?*

No, the TourSG dataset will be provided under a free of charge end-user-license to all participants in the Fourth Dialog State Tracking Challenge.

6.2. *Can I get the TourSG dataset without participating in the Challenge?*

Yes, but no free of charge end-user-license is available to non-participants. You or your company will need to pay a license fee for getting the TourSG dataset without participating in the Fourth Dialog State Tracking Challenge.

6.3. *Is participation in the main task of the Challenge mandatory?*

Yes, participation in the main task of the Challenge is mandatory for registered participants.

6.4. *Are baseline systems and evaluation scripts going to be provided?*

A baseline system and evaluation scripts will be provided only for the main task of the Challenge. Baselines and evaluation protocols for pilot tasks and open track are to be agreed directly with participants on such tasks.

6.5. *Is participation in the pilot tasks and open track of the Challenge mandatory?*

No, participation in the pilot tasks and open track of the Challenge is optional for registered participants.

6.6. *Do I need to participate in the first three pilot tasks in order to be able to participate in the "end-to-end system" task?*

No, but you need that at least one participant participates in each of the other pilot tasks. Otherwise you will not be able to set an end-to-end system, as no baselines are provided for any of the pilot tasks.

7. Subscription to DSTC4 mailing list

To join the mailing list, send an email to listserv@lists.research.microsoft.com with 'subscribe DSTC' in the body of the message (without quotes). Joining the list is encouraged for those with an interest in the challenge, and is a necessity for those participating.

Post to the list using the address: dstc@lists.research.microsoft.com.

8. Committees

8.1. *Organizing Committee*

Seokhwan Kim - I2R A*STAR
Luis F. D'Haro - I2R A*STAR
Rafael E Banchs - I2R A*STAR
Matthew Henderson - Google
Jason Williams - Microsoft Research

8.2. *Advisory Committee*

Paul Crook - Microsoft Research
Maxine Eskenazi - Carnegie Mellon University
Milica Gasic - University of Cambridge
Sungjin Lee - Carnegie Mellon University
Oliver Lemon - Herriot Watt
Olivier Pietquin - SUPELEC
Joelle Pineau - McGill University
Deepak Ramachandran - Nuance Communications
Steve Young - University of Cambridge

9. References

[1] Jason D Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In In Proceedings 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), Metz, France, 2013. DSTC1 website: http://research.microsoft.com/en-us/events/dstc/

[2] Matthew Henderson, Blaise Thomson, and Jason Williams. The Second Dialog State Tracking Challenge. In Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, p. 263. 2014. DSTC 2 and 3 website: http://camdial.org/~mh521/dstc/

[3] Henderson, Matthew, Blaise Thomson, and Jason Williams. The Third Dialog State Tracking Challenge. In Proceedings of IEEE Spoken Language Technology (2014). DSTC 2 and 3 website: http://camdial.org/~mh521/dstc/

[4] K.M. Ting (2010), Encyclopedia of Machine Learning, p 781.

[5] K. Papineni et al., "BLEU: a method for automatic evaluation of machine translation", in Proc. of the 40th Annu. Meeting of the Assoc. for Computational Linguistics, Philadelphia, PA, USA, Jul 2002, pp. 311-318

[6] R.E. Banchs, L.F. D'Haro, H Li (2015) "Adequacy - Fluency Metrics: Evaluating MT in the Continuous Space Model Framework", IEEE/ACM Transactions on Audio, Speech and Language Processing - Special issue on continuous space and related methods in natural language processing Vol.23, No.3, pp.472-482