Dialog state tracking challenge handbook

Jason D. Williams, Antoine Raux, Deepak Ramachandran, Alan Black

The dialog state tracking challenge (DSTC) is a research community challenge task for accurately estimating a user's goal in a spoken dialog system

http://research.microsoft.com/en-us/events/dstc/

Background, motivation, and challenge overview

In dialog systems, "state tracking" – sometimes also called "belief tracking" – refers to accurately estimating the user's goal as a dialog progresses. Accurate state tracking is desirable because it provides robustness to errors in speech recognition, and helps reduce ambiguity inherent in language within a temporal process like dialog. Dialog state tracking is an important problem for both traditional uni-modal dialog systems, as well as speech-enabled multi-modal dialog systems on mobile devices, on tablet computers, and in automobiles.

Recently, a host of models have been proposed for dialog state tracking [DSTC position paper]. However, comparisons among models are rare, and different research groups use different data from disparate domains. Moreover, there is currently no common dataset which enables off-line dialog state tracking experiments, so newcomers to the area must first collect dialog data (which is expensive and time-consuming), or resort to simulated dialog data (which can be unreliable). All of these issues hinder advancing the state-of-the-art.

In this challenge, participants will use a provided set of labeled human-computer dialogs to develop a dialog state tracking algorithm. Algorithms will then be evaluated on a common set of held-out dialogs, offline, to enable comparisons [2]. Participants will not need to implement or operate a speech recognizer, parser, real-time system, or text-to-speech engine. The data for this challenge will be taken from the Spoken Dialog Challenge [1], which consists of human/machine spoken dialogs with real users (not usability subjects).

At the start of the challenge – the development phase – participants will be provided with a training set of transcribed and labeled dialogs. Participants will also be given code that implements the evaluation measurements. Participants will then have several months to optimize their algorithms.

At the end of the challenge, participants will be given an untranscribed and unlabeled test set, and a short period to run their algorithm against the test set. Participants will submit their algorithms' output to the organizers, who will then perform the evaluation. After the challenge, the test set transcriptions and labels will be made public.

The main objective of the challenge is to enable reliable comparisons between *different approaches* to dialog state tracking. To date, different algorithms have been tested on different datasets, and few direct comparisons have been done. The challenge seeks to advance the state-of-the-art by providing a forum for direct comparisons, in which different algorithms are trained and tested on the same data. We believe this is the best way to study the relative strengths of the algorithms.

The challenge is structured as an off-line task, in which algorithms are evaluated on a fixed corpus. This structure provides a low barrier to entry, broadening the audience of potential participants. In practice however, a dialog state tracker is used in an on-line system, to make (hopefully!) better decisions about which actions to take. So in a real deployment, we know in advance that the distribution of the training condition and runtime condition will be different. Thus partitioning a single corpus into a training set and a test set with the same distributions would fail to capture this difference.

The most robust test of this new tracker would be to re-deploy it and measure quantities like task completion, accuracy, user satisfaction, etc. However, in place of that, this challenge uses an off-line evaluation which does account for a strong difference between training and testing by using test sets which differ *in the dialog system used*. This design minimizes the chances of over-stating performance that would result by using train and test data drawn from the same distribution.

In addition, data scarcity is a constant problem with dialog systems. To test this, the test sets will also explore different amounts of same-system training data: large amounts (1000s of dialogs), small amounts (100s of dialogs), and none.

Concretely, at the start of the challenge, participants will be provided with:

1. Training data: System log files in a common JSON format [1]
2. Training labels: Utterance transcriptions, and labels indicating user goal
3. The scoring tool that will be used in the evaluation stage
4. A baseline dialog state tracker
5. A bus schedule database covering much of the relevant time period

At the end of the system development period, participants will be provided with test data, consisting of held-out system log files using the same format. Participants will run their dialog state trackers on this data, and send the output to the organizers for scoring. After scoring, the test logs will be shared with all participants.

Participation information

Participation is welcomed by any research team (academia, corporate, non-profit, government). Members of the organizational committee and advisory committee may also participate.

In general, the identity of participants will not be published or made public. In written results, teams will be identified as TEAM1, TEAM2, etc. There are 2 exceptions to this: (1) the organizers will *verbally* indicate the identities of all teams at the conference/workshop chosen for communicating results (planned for SigDial 2013); and (2) participants may identify their own team label (eg TEAM5), in publications or presentations, if they desire, but may not identify the identities of other teams.

Each participating team may enter up to 5 algorithms – i.e., each team may submit up to 5 sets of results.

The dialog data used in this challenge has been contributed by several researchers, and it is likely those researchers will also be interested in this challenge. As such it is not realistic to ensure that no participant has access to any of the test sets. Participants who may have access to a testing set agree that they will in no way use the testing set in the development of their algorithms. Further, when reporting results, the organizers will note which participants produced which test sets. Thus both participants and corpora will be anonymized, with corpora being labeled as train1, test2, etc. and participants labeled as TEAM1, TEAM2, etc.

Announcements and discussion about the challenge will be conducted on the group mailing list. Participants should be on the mailing list. Instructions for joining the mailing list can be found on the DSTC homepage: http://research.microsoft.com/en-us/events/dstc/

Overview of data, datasets, and labels

DSTC log data is taken from different spoken dialog systems which provide bus timetables for Pittsburgh, Pennsylvania, USA. The dialog systems were fielded by three different groups, here called teams A, B, and C. These names have been anonymized since some of these deployments were done as part of the Spoken Dialog Challenge [3].

An example conversation is provided below, from a Group B dialog system. The corresponding dialog acts are also provided. The full dialog act taxonomy is given in the appendix.

System: East Pittsburg Bus Schedules. Say a bus route, like 28X, or say I'm not sure.
hello(), request(route), example(route=28x), example(route=dont_know)
User: 61A
SLU: 0.77 inform(route=61a)
0.12 inform(route=61)
0.01 inform(route=61d)
System: Okay, 61A. To change, say go back. Where are you leaving from?
impl-conf(route=61a), example(act=goback), request(from)
User: Downtown
SLU: 0.59 inform(from.desc=downtown)
0.10 inform(from.desc=from downtown)
System: Okay, downtown. You can always say go back. And where are you going to?
impl-conf(from.desc=downtown), example(act=goback), request(to)
User: East Pittsburgh East Pittsburgh
SLU: 0.25 inform(to.desc=pittsburgh)
0.20 inform(to.desc=east pittsburgh)

0.01 inform(to.desc=to pittsburgh)
System: Pittsburgh. Is that right?
expl-conf(to.desc=pittsburgh)
User: No
SLU: 0.71 negate()
0.07 affirm()
System: Oh, sorry, where are you going to?
sorry(), request(to)
User: East Pittsburgh
SLU: 0.97 inform(to.desc=east pittsburgh)
System: Okay, East Pittsburgh. Do you want times for the next few buses? Say yes or no.
impl-confirm(to.desc=east pittsburgh) , expl-confirm(act=want-next-bus),
example(act=affirm), example(act=negate)
User: Yes
SLU: 0.97 affirm()
System: Ok, the next few buses. To change, say go back. At 7:45 PM today, there is a 61 A,
from Fifth Avenue And Market Street, Downtown, arriving Dynamo Alley And Beech
Avenue, In East Pittsburgh, at 8:42 PM. Say repeat, next, previous, or start over.
impl-conf(act=want-next-bus), example(act=goback),
schedule(result.from.time.hour=7, result.from.time.minute=45,
result.from.time.ampm=pm, result.from.date.day=today, result.route=61a,
result.from.desc=fifth avenue and market street, downtown, result.to.desc=dynamo
alley and beech avenue, in east pittsburgh, result.to.time.hour=8,
result.to.time.minute=42, result.to.time.ampm=pm), example(act=repeat),
example(act=nextbus), example(act=prevbus), example(act=restart)
User: [hang up]

The user's speech is converted into words by an *automatic speech recognizer* (ASR), and these words are converted into meaning by a *spoken language understanding* (SLU) processes. These processes yield multiple hypotheses, so for each user turn the dialog tracker receives zero or more *SLU hypotheses,* each with a local confidence score.

Each SLU hypothesis and each system output can be described as one or more *dialog acts,* such as *request(to)* or *inform(to.desc=east pittsburgh).* Each dialog act has a *type* (eg request) and zero or more slot/value pairs (eg the slot *to.desc* and the value *east pittsburgh*).

Tracker output

The objective of a dialog state tracker is to correctly identify the user's goal. In this challenge we will exploit two properties of the domain to make the challenge efficient.

First, in this domain, the user's goal is generally fixed. The user enters the call with a specific goal in mind. Further, when goal changes do occur, they are usually explicitly marked: since the system first collects information, and then provides bus information, if the user wishes to change their goal, they need to *start over* from the beginning. These "start over" transitions are obvious in the logs.

Second, in this domain, the user's goal is an assignment of values to various slots. There are generally a large number of values for each slot, and the coverage of N-best lists is generally good, so the likelihood of a tracker correctly guessing a value which has not been observed on an N-Best list is very small. This means that we can limit consideration of goals to slots and values that have been observed in a dialog act in an SLU result.

By exploiting these two aspects of this domain, the task of a dialog state tracker in this challenge can be defined as *assigning a score to each observed dialog act.* In other words, the tracker will output a list of observed slot/value pairs, with a score in the range [0,1] for each pointer. The sum of scores is limited to 1.0, and 1.0 – (sum of scores) is defined as the score of a special value that means the user's goal has not yet been observed on any SLU result. The scores themselves need not be probability, but must obey these conventions (and one of the evaluation metrics measures the quality of score as a probability). The set of observed dialog acts is cleared each time the system "starts over" from the beginning.

Labels indicate whether each dialog act is correct. Labels are assigned by people who look at the words spoken by the user, dialog context, and a recognized dialog act, and decide whether the dialog act correctly captures all of the information relevant to that slot. A state tracker's output can then be evaluated by looking up the correctness of each of the dialog acts it refers to. Note there is no distinction between utterances which are in-domain or out-of-domain – labeling merely assigns binary labels to recognized dialog acts.

In this challenge there are 5 high-level slots: **route, from, to, date**, and **time**. The **from** and **to** slots are sub-divided into **from.desc, from.neighborhood, from.monument**, and similarly for **to**, so in total there are 9 slots which are evaluated. This challenge will consider both marginal and joint representations of dialog states, since both are useful for making dialog decisions. In the joint list, each state hypothesis is an assignment of values to *all* slots. Thus the tracker will output 9 marginal lists and 1 joint list, which are all evaluated separately.

Evaluation metrics

There are a variety of aspects of tracker performance that will be measured.

1. Hypothesis accuracy: Percent of turns in which the tracker's 1-best hypothesis is correct. This measures raw 1-best accuracy
2. Mean reciprocal rank: Average of 1/R, where R is the rank of the first correct hypothesis. This measures the quality of the ranking
3. ROC performance: This includes several metrics that assess the discrimination of the top hypothesis's score.
    a. Equal error rate: The sum of false accepts (FAs) and false rejects (FRs) where FA=FR
    b. Correct accept 5: The percent correct accepts (CAs) when there are at most 5% False Accepts (FAs)
    c. Correct accept 10: The percent CAs when there are at most 10% FAs

d. Correct accept 20: The percent CAs when there are at most 20% FAs

4. Brier score (L2 norm): The L2 norm between the vector of scores output by dialog state tracker and a vector with 1 in the position of the correct item, and 0 elsewhere. If there are multiple correct items, the correct item which has been assigned the highest score is chosen. This measures the calibration of the scores – ie the extent to which they are good probabilities.
5. Average score: This measures the average score assigned to the correct item.

In this challenge task, each of these will be measured and reported separately for each concept, and also jointly for all concepts.

When measuring these metrics, which turns will be included? There are (at least) three reasonable "schedules" for determining which turns to include in each evaluation:

1. Schedule 1: Include all turns (regardless of dialog context).
2. Schedule 2: Include a turn for a given concept only if that concept either appears on the SLU N-Best list in that turn, or if the system's action references that concept in that turn (eg an explicit or implicit confirmation)
3. Schedule 3: Include only the turn before the system starts over from the beginning, and the last turn of the dialog.

Schedule 1 allows for a state tracker to change its hypothesis about a user goal which isn't in focus, and still receive credit. For example, new information about the location might change the hypothesis for the route. However, schedule 1 makes comparisons *across* concepts invalid: concepts which appear at the beginning of the dialog will be measured more times than concepts at the end of the dialog. Also, dialogs with more turns will have a greater effect than dialogs with fewer turns.

Schedule 2 makes comparisons across concepts valid, since concepts are only measured when they are in focus; however, it does not allow for a state tracker to receive credit for new guesses about concepts which aren't in focus. As with Schedule 1, dialogs with more turns will have a greater effect than dialogs with fewer turns.

In schedule 3, each "mini-dialog" carries the same weight. However schedule 3 doesn't attempt to measure correctness within the mini-dialog at all.

All metrics will be reported on all three schedules. This is a large set of metrics to report! The reason for reporting a large set is that different metrics are important to different downstream processes. For example, schedule2 accuracy may be most important to a hand-crafted uni-modal dialog manager; schedule3 mean reciprocal rank may be most important to a multi-modal interface that can display results to the user; schedule2 ROC performance may be most important in high noise; and schedule1 Brier score may be most important to a statistical dialog controller.

It is also desirable to measure CPU runtime. Since each team's tracker will run on their own machines, direct comparisons of runtime will not be possible. Nonetheless, it would be useful to know whether each algorithm can run in real time or not. Thus, for each turn in each test dialog, trackers will also output the wall-clock time required for that turn.

Finally, it is also desirable to compare performance on a validation set (ie a set which is not used for training directly, but which is used in the course of development). To do this, participants will be asked for results on the "half2" set for each of the training sets. In the final evaluation, participants are of course welcome to use the entire training set to train.

Datasets

The data is divided into 4 training sets and 4 test sets

Dataset	Source	Calls	Time period	Transcribed?	Labeled?	Notes
train1a	Group A	1013	September 2009	Yes	Yes	
train1b	Group A	14,545	16 Months (2008-2009)	Yes	No	
train2	Group A	678	Summer 2010	Yes	Yes	
train3	Group B	779	Summer 2010	Yes	Yes	
test1	Group A	765	Winter 2011-12	Yes	Yes	
test2	Group A	983	Winter 2011-12	Yes	Yes	
test3	Group B	1037	Winter 2011-12	Yes	Yes	
test4	Group C	451	Summer 2010	Yes	Yes	

To have standardized development sets, the training sets are each split in half. Participants will be asked to report results on the second half of each set (except train1c) at the time the test set is provided.

The data from group A in train1*, train2, and test1 was collected using essentially the same dialog system. Only a handful of updates were made to reflect changes to the bus schedule. The data in test2 was collected using a different version of group A's dialog manager. The data from group B in train3 and test3 were collected using essentially the same dialog system; the main difference is that test3 covers more bus routes. References to published papers describing the systems will be made available to challenge participants (and aren't included here to help maintain anonymity).

Using these training and test sets enables the challenge to explore a variety of testing conditions:

Dataset	Quantity of similar-system training data	Similarity to training data
test1	Large	Very similar to train1* and train2
test2	Large	Somewhat similar to train1* and train2
test3	Small	Very similar to train3
test4	None	Distinct from all training data

- Test1 tests the condition when training and testing use very similar dialog systems, and when there is a large quantity of same-system training data available.

- Test2 tests the condition when training and testing use somewhat similar dialog systems, and when there is a large quantity of same-system training data available.
- Test3 tests the condition when training and testing use very similar dialog systems, and when there is a small quantity of same-system training data available
- Test4 tests the condition when training and testing use totally different dialog systems, and when there is no same-system training data available. (A handful of Test4 dialogs will be made available in advance of the evaluation, to verify that algorithms run on this log format.)

Overall, these conditions test the effects on belief tracking of different speech recognizers/language understanding components, different dialog strategies (eg amount and type of confirmation, ordering of questions, initiative), different system prompt wordings and voices, among others.

Live and batch recognition for group A

One of the benefits of statistical dialog state tracking is that it can make use of all of the items on an SLU N-Best list. In deployment, group B and C systems produced N-Best lists of SLU output, and their logs include this information. However, group A systems produced only 1-best SLU output in deployment.

In an attempt to provide more useful information, recognition has been re-run in *batch* on the group A data to produce N-Best lists. Unfortunately it was not possible to replicate the deployed group A recognition and SLU process exactly. As a result, the 1-best batch result does not always match the 1-best live result. The logs from group A systems used in this challenge include both the live and batch recognition results.

Baseline tracker and scoring tool

A baseline dialog state tracker is provided. This dialog state tracker both provides a performance baseline, as well as a code template to follow. It scans all dialog acts for each slot observed so far in the dialog, and outputs the (one) hypothesis which has the highest score. For the joint output, it does the same, using the highest average score. Refer to the project webpage for download and installation instructions.

The tools assume that the data is all located in a directory (here called “../data”) with sub directories for each corpus. For example:

```
../data/train3/<session-id>/dstc.log.json
```

When labels are present, they are located alongside the logs:

```
../data/train3/<session-id>/dstc.labels.json
```

The baseline tracker can be run like this:

```
> bin/baseline --dataset=train3.half2 --dataroot=../data --trackfile=track.json
```

This runs the baseline on the train3 validation set and stores the output in out.json. To score this:

```
> bin/score --dataset=train3.half2 --dataroot=../data \
  --trackfile=track.json --scorefile=score.csv
```

This evaluates the output of the baseline and stores the results in out.csv, which contains the following columns:

- Column 1: slot (or “joint”)
- Column 2: schedule – one of “schedule1”, “schedule2”, “schedule3”
- Column 3: metric – one of accuracy, avgp, l2, mrr, roc.ca05, roc.ca10, roc.ca20, roc.eer
- Column 4: instances included in this line
- Column 5: metric. If the metric cannot be computed – for example, if there are no samples -- the value is “None”

Example excerpt from out.csv:

```
date,schedule1,accuracy,4459,0.91320923974
date,schedule1,avgp,4459,0.914002698811
date,schedule1,l2,4459,0.121618549668
date,schedule1,mrr,4459,0.955371159453
date,schedule1,roc.ca05,4459,0.868580399193
date,schedule1,roc.ca10,4459,0.91320923974
date,schedule1,roc.ca20,4459,0.91320923974
date,schedule1,roc.eer,4459,0.0461986992599
date,schedule2,accuracy,189,0.820105820106
date,schedule2,avgp,189,0.660067010582
```

To format into a pretty report and print to stdout:

```
> bin/report --scorefile=score.csv
```

This produces:

```
-----------------------------------------------------------------------------------
                                    schedule1
-----------------------------------------------------------------------------------
               route from.d from.m from.n to.des to.mon to.nei   date   time  joint
            N   4459   4459   4459   4459   4459   4459   4459   4459   4459   4459
     accuracy 0.7596 0.8143 1.0000 1.0000 0.8545 1.0000 1.0000 0.9132 0.9729 0.4833
         avgp 0.6729 0.7439 1.0000 1.0000 0.8254 1.0000 1.0000 0.9140 0.9578 0.4101
           l2 0.4627 0.3622 0.0000 0.0000 0.2469 0.0000 0.0000 0.1216 0.0597 0.8343
          mrr 0.8006 0.8863 1.0000 1.0000 0.9102 1.0000 1.0000 0.9554 0.9775 0.5052
     roc.ca05 0.3292 0.5414 1.0000 1.0000 0.7919 1.0000 1.0000 0.8686 0.9729 0.1563
     roc.ca10 0.4900 0.6638 1.0000 1.0000 0.8354 1.0000 1.0000 0.9132 0.9729 0.2541
     roc.ca20 0.7448 0.8143 1.0000 1.0000 0.8545 1.0000 1.0000 0.9132 0.9729 0.3030
      roc.eer 0.2554 0.2584 0.0000 0.0000 0.1133 0.0000 0.0000 0.0462 0.0184 0.3265

-----------------------------------------------------------------------------------
                                    schedule2
-----------------------------------------------------------------------------------
               route from.d from.m from.n to.des to.mon to.nei   date   time  joint
            N   1165   1159      0      0    956      0      0    189    172   2922
     accuracy 0.6403 0.6428      -      - 0.6820      -      - 0.8201 0.5988 0.4908
         avgp 0.5751 0.6138      -      - 0.6456      -      - 0.6601 0.5428 0.4214
           l2 0.6010 0.5462      -      - 0.5012      -      - 0.4807 0.6466 0.8183
          mrr 0.7047 0.7856      -      - 0.7965      -      - 0.8889 0.6715 0.5202
     roc.ca05 0.1957 0.2347      -      - 0.4351      -      - 0.4709 0.2965 0.1369
     roc.ca10 0.3193 0.3943      -      - 0.5209      -      - 0.5397 0.4302 0.2358
     roc.ca20 0.5373 0.5263      -      - 0.6161      -      - 0.8201 0.4593 0.3460
      roc.eer 0.3442 0.3046      -      - 0.2448      -      - 0.2328 0.2791 0.3552

-----------------------------------------------------------------------------------
                                    schedule3
-----------------------------------------------------------------------------------
```

```
               route from.d from.m from.n to.des to.mon to.nei   date   time  joint
         N       379    331      0      0    305      0      0     54     50    379
  accuracy    0.7177 0.7583      -      - 0.7738      -      - 0.8889 0.7000 0.5066
      avgp    0.6468 0.7003      -      - 0.7120      -      - 0.7522 0.6244 0.4316
        l2    0.4995 0.4239      -      - 0.4073      -      - 0.3505 0.5312 0.8039
       mrr    0.7718 0.8520      -      - 0.8607      -      - 0.9352 0.7600 0.5343
  roc.ca05    0.3087 0.4169      -      - 0.5574      -      - 0.6852 0.5200 0.1319
  roc.ca10    0.4433 0.6133      -      - 0.6590      -      - 0.8519 0.5600 0.2454
  roc.ca20    0.6649 0.7372      -      - 0.7639      -      - 0.8889 0.6000 0.3562
   roc.eer    0.2902 0.2236      -      - 0.2033      -      - 0.1111 0.2400 0.3377

-------------------------------------------------------------------------------
                                  basic stats
-------------------------------------------------------------------------------
            dataset : train3.half2
      scorer_version : TRUNK
            sessions : 344
     total_wall_time : 3.18700003624
               turns : 4459
  wall_time_per_turn : 0.000714734253474
```

Slot names are truncated, so “from.d” means “from.desc”; “from.m” means “from.monument”, etc. Train3 does not output *monument* or *neighborhood* values for locations, so these are blank.

References

[1] JavaScript Object Notation, http://www.json.org/

[2] Jason D. Williams, A belief tracking challenge task for spoken dialog systems, in NAACL HLT 2012 Workshop on Future directions and needs in the Spoken Dialog Community: Tools and Data, Association for Computational Linguistics, June 2012.

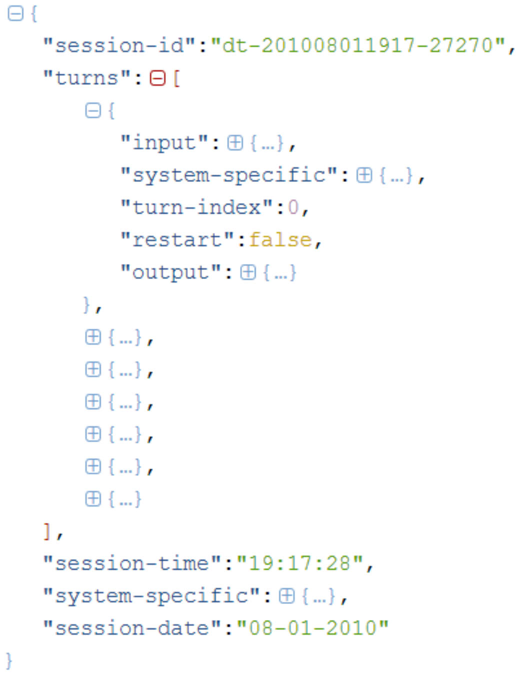
[3] Black, A., Burger, S., Conkie, A., Hastie, H., Keizer, S., Lemon, O., Merigaud, N., Parent, G., Schubiner, G., Thomson, B., Williams, J., Yu, K., Young, S., and Eskenazi, M. Spoken Dialog Challenge 2010: Comparison of Live and Control Test Results, SIGDial 2011 pp 22-27, Portland Oregon.

Appendix A: JSON format for log data

The output of each system has been mapped to a common log format in JSON format. There is one log file per call. This common log format includes:

- System output, at the word and dialog act level, using a common set of dialog acts and slot/value pairs
- System input, at the word (ASR) and dialog act level (SLU) level, using a common set of dialog acts and slot/value pairs. Each ASR hypothesis and each SLU hypothesis includes a confidence score. Group A logs also include batch ASR and SLU results.
- Start time of each turn, relative to the start of the dialog
- An indicator for when the system “restarts” from the beginning
- System-specific information. For example, the Group B logs include additional information about recognition results, the state of the belief tracker it used at runtime.

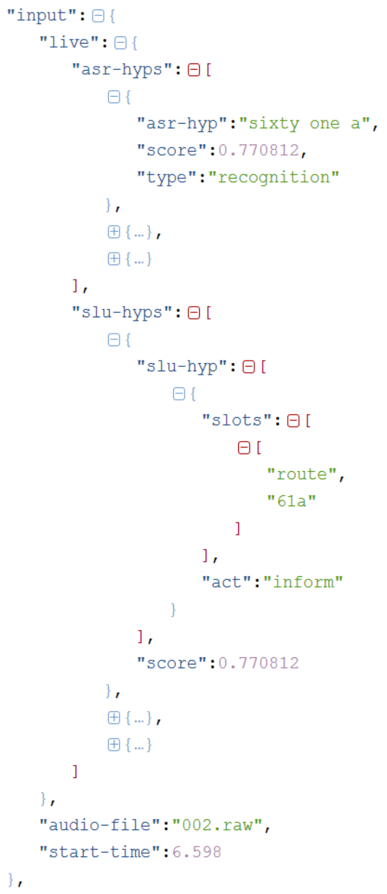
Here is an example of the top level for each log file:



```
⊟{
   "session-id":"dt-201008011917-27270",
   "turns":⊟[
      ⊟{
         "input":⊞{…},
         "system-specific":⊞{…},
         "turn-index":0,
         "restart":false,
         "output":⊞{…}
      },
      ⊞{…},
      ⊞{…},
      ⊞{…},
      ⊞{…},
      ⊞{…},
      ⊞{…}
   ],
   "session-time":"19:17:28",
   "system-specific":⊞{…},
   "session-date":"08-01-2010"
}
```

- session-id : a unique indicator for the call
- session-date: a string of the form MM-DD-YYYY
- session-time: a string of the form HH:MM:SS or HH:MM:SS.S+
- turns: an array of dictionaries, one dictionary per turn. Turn entries include:
    - input: the input received by the dialog manager for the turn (described below)
    - system-specific: system-specific information associated with this turn
    - turn-index: 0-based index of this turn
    - restart: a flag indicating if this turn re-started from the beginning
    - output: the output generated by the dialog manager in this turn (described below)
- system-specific: system-specific information associated with the whole call

Here is detail for the system input:



```
"input":⊟{
  "live":⊟{
    "asr-hyps":⊟[
      ⊟{
        "asr-hyp":"sixty one a",
        "score":0.770812,
        "type":"recognition"
      },
      ⊞{…},
      ⊞{…}
    ],
    "slu-hyps":⊟[
      ⊟{
        "slu-hyp":⊟[
          ⊟{
            "slots":⊟[
              ⊟[
                "route",
                "61a"
              ]
            ],
            "act":"inform"
          }
        ],
        "score":0.770812
      },
      ⊞{…},
      ⊞{…}
    ]
  },
  "audio-file":"002.raw",
  "start-time":6.598
},
```

The “input” dictionary has entries for:

- live: contains recognition results produced in deployment
- batch: contains recognition results generated in batch, off-line (present only for systems from group A. The format within the “live” and “batch” entries is identical.
- audio-file: a pointer to the audio file corresponding to the system input
- start-time: number of seconds since the start of the call that this turn started (not present for test4 dataset)
- end-time: number of seconds since the start of the call that this turn ended (only present for Group A systems)

Recognition results consist of an array of ASR results in “asr-hyps” and an array of SLU results in “slu-hyps”. The number of ASR and SLU hyps need not be equal.

Each ASR hyp has the following entries:

- asr-hyp: the words recognized
- score: The score of this ASR hypothesis as reported by the recognizer. The for “live” results, the score is guaranteed to be in [0,1], but scores might not sum to 1. For “batch” results, the score is a real number.

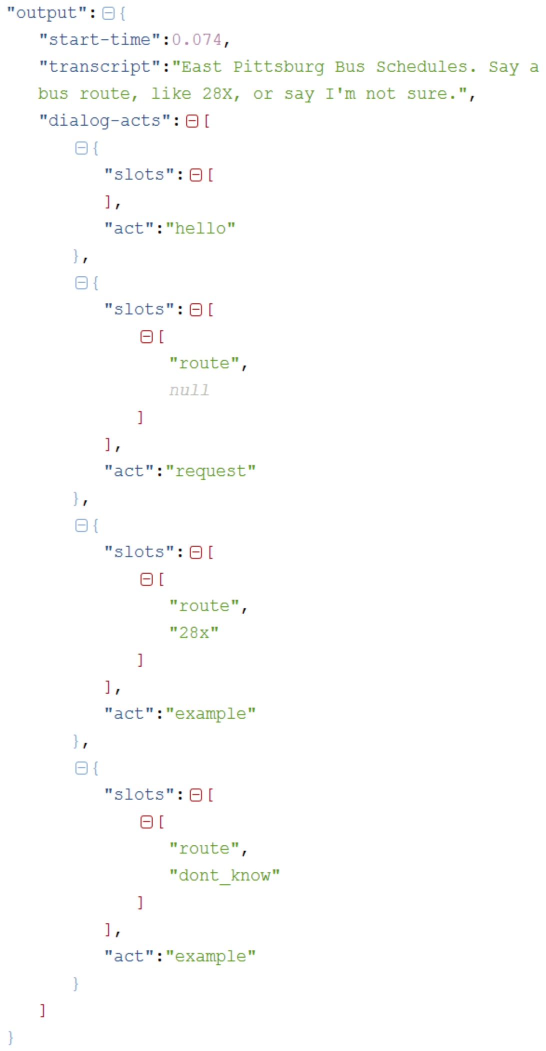
- type: currently always “recognition”; in future other types may be added, such as “dtmf”

Each SLU hyp has the following entries:

- slu-hyp: a structure containing the SLU hypothesis
- score: The score of this SLU hypothesis. The for “live” results, the score is guaranteed to be in [0,1], but scores might not sum to 1. For “batch” results, the score is a real number.

An SLU hypothesis contains one or more dialog acts. Each act consists of an act type (“act”), and a list of zero or more slot/value pairs (“slots”). See the dialog act taxonomy below for details.

Here is detail for the system output:



```
"output":⊟{
   "start-time":0.074,
   "transcript":"East Pittsburg Bus Schedules. Say a
   bus route, like 28X, or say I'm not sure.",
   "dialog-acts":⊟[
      ⊟{
         "slots":⊟[
         ],
         "act":"hello"
      },
      ⊟{
         "slots":⊟[
            ⊟[
               "route",
               null
            ]
         ],
         "act":"request"
      },
      ⊟{
         "slots":⊟[
            ⊟[
               "route",
               "28x"
            ]
         ],
         "act":"example"
      },
      ⊟{
         "slots":⊟[
            ⊟[
               "route",
               "dont_know"
            ]
         ],
         "act":"example"
      }
   ]
}
```

The “output” dictionary has entries for:

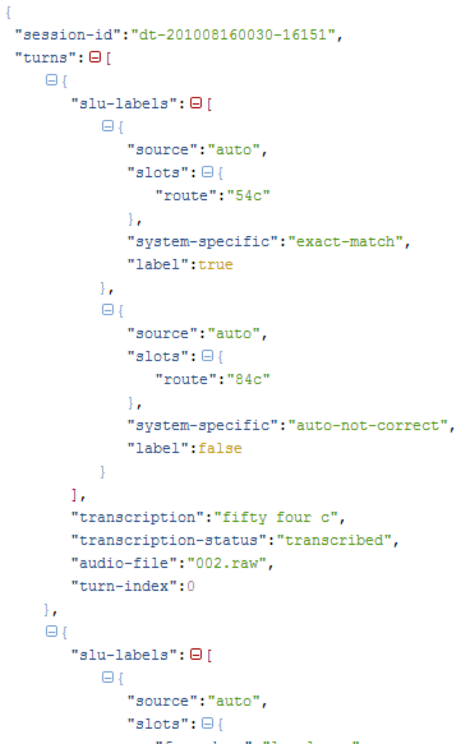
- start-time: number of seconds since the start of the call that the system output started (not present for test4 dataset)

- end-time: number of seconds since the start of the call that this turn ended (only present for Group A systems)
- transcript: the words spoken by the system. If the system detected user speech before all system words were spoken (and the system stopped speaking), it is possible that not all of this transcript was read to the user
- dialog-acts: An array of one or more dialog acts that describes the transcript. See the dialog act taxonomy below for details.

Appendix B: JSON format for labels

Label files indicate what the caller actually said in each turn, and the correctness of each SLU hypothesis. They are in JSON format and follow a structure similar to the log file JSON.

An excerpt of a label file is given below.



```
{
 "session-id":"dt-201008160030-16151",
 "turns":⊟[
    ⊟{
       "slu-labels":⊟[
          ⊟{
             "source":"auto",
             "slots":⊟{
                "route":"54c"
             },
             "system-specific":"exact-match",
             "label":true
          },
          ⊟{
             "source":"auto",
             "slots":⊟{
                "route":"84c"
             },
             "system-specific":"auto-not-correct",
             "label":false
          }
       ],
       "transcription":"fifty four c",
       "transcription-status":"transcribed",
       "audio-file":"002.raw",
       "turn-index":0
    },
    ⊟{
       "slu-labels":⊟[
          ⊟{
             "source":"auto",
             "slots":⊟{
```

A label file consists of a dictionary with two entries:

- session-id: a unique identifier for the session and matches the session-id in the log file
- turns: an array of dictionaries, which mirrors the "turns" array in the log file

Each entry in the "turns" array is a dictionary with entries:

- transcription: what the caller actually said (may not be present – see next entry)
- transcription-status: indicates if the transcription was done. Possible values are:

    - audio-not-available: There was no audio file captured from this turn. Transcription not done.
    - no-speech-for-system: The transcriber indicated that there was no speech directed at the system in this turn. Transcription not done.
    - not-understandable: Transcriber indicated that the user speech was not understandable
    - silence: Transcriber indicated that there was no speech in the audio. Transcription not done.
    - transcribed: audio was transcribed.
- audio-file: a pointer to the audio file for this turn (if any). Matches log file
- turn-index: a zero-based index of the turn. Matches log file
- slu-labels: a dictionary containing the labels – see below.
- system-specific: an optional dictionary with additional information about the label. For example, train3 contains labels for the correctness of each (live) dialog act hypothesis.

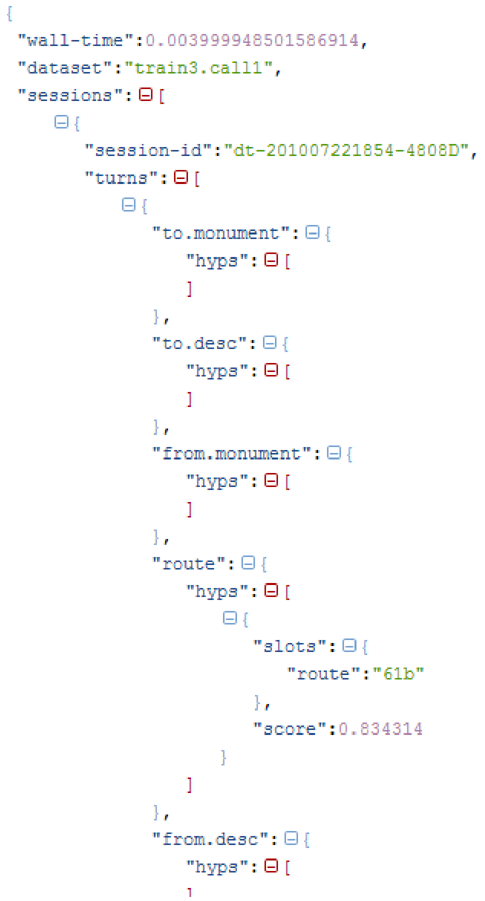
The "slu-labels" entry is an array of dictionaries. Each dictionary corresponds to one slot group and one label (correct or incorrect), with the entries:

- slots: a dictionary of 1 or more slot/value pairs. The slot/value pairs are all drawn from the same label group. Most label groups correspond to a single slot; the 2 exceptions are "date" and "time".
- label: indicates correctness of this dialog act
    - true: the meaning of this dialog act is (entirely) correct
    - false: the meaning of this dialog act is not (entirely) correct
    - null: this dialog act was not labeled
- source: indicates how the label was produced
    - auto: this label was automatically assigned
    - human: this label was assigned by a human judge
- system-specific: (if present) provides system-specific information about how the label was generated

Appendix C: JSON format for tracker output

Trackers output a JSON file that specifies their hypotheses. All sessions for a given corpus are included in one JSON file (so this file may be rather large).

Example tracker output:



```
{
 "wall-time":0.003999948501586914,
 "dataset":"train3.call1",
 "sessions":⊟[
    ⊟{
      "session-id":"dt-201007221854-4808D",
      "turns":⊟[
         ⊟{
            "to.monument":⊟{
               "hyps":⊟[
               ]
            },
            "to.desc":⊟{
               "hyps":⊟[
               ]
            },
            "from.monument":⊟{
               "hyps":⊟[
               ]
            },
            "route":⊟{
               "hyps":⊟[
                  ⊟{
                     "slots":⊟{
                        "route":"61b"
                     },
                     "score":0.834314
                  }
               ]
            },
            "from.desc":⊟{
               "hyps":⊟[
               ]
```

The top level is a dictionary, with entries:

- wall-time: the total wall time to run the tracker on this dataset
- dataset: the dataset used as input to the tracker
- sessions: an array of dictionaries, each being one session. The order of sessions much match the order in the dataset file

Each session is a dictionary with entries:

- session-id: the session-id of this session.
- turns: an array of turns. The order must match the order used in the logs (and labels) file.

Each turn is a dictionary with 10 entries: one for each slot group, and one for joint. Slot groups contain one or more slots:

Slot group	Slots in this slot group
route	route
from.desc	from.desc
from.neighborhood	from.neighborhood
from.monument	from.monument
to.desc	to.desc

to.neighborhood	to.neighborhood
to.monument	to.monument
date	date.day date.absmonth date.absday date.relweek
time	time.hour time.minute time.ampm time.arriveleave time.rel
joint	[contains all slot groups]

Each of these entries is a dictionary, with one entry, called “hyps”. “hyps” is an array; each entry in the array is a dictionary with 2 entries:

- slots: a dictionary with slots/value pairs in this slot group. This must match a slot/value pair that has been recognized earlier in the dialog, but not before a “restart”
- score: the tracker’s score, which must be in the range [0...1], and which must sum to 1.0 or less.

Appendix D: Dialog act taxonomy

System input and output for all systems has been mapped to a unified dialog act taxonomy. A user or system turn consists of one or more dialog acts. Each act consists of an act type, and zero or more slot/value pairs. The same slot may appear multiple times in an act.

Act types are as follows.

Act type	**Can act be produced by user?**	**Can act be produced by system?**	**Does act take any slots?**
hello	X	X	
bye	X	X	
goback	X	X	
restart	X	X	
null	X	X	
repeat	X		
nextbus	X		
prevbus	X		
tellchoices	X		
affirm	X		
negate	X		
deny	X		X
inform	X		X
want-next-bus		Only in expl-conf and impl-conf, eg: expl-conf(act=want-next-bus)	
ack		X	
hold-on		X	
open-request		X	
bebrief		X	

sorry		X	
please-repeat		X	
please-rephrase		X	
are-you-there		X	
didnthear		X	
impl-confirm		X	X
expl-confirm		X	X
request		X	X
schedule		X	X
morebuses		X	X
canthelp.nonextbus		X	
canthelp.route_doesnt_run		X	X
canthelp.no_connection		X	X
canthelp.uncovered_route		X	X
canthelp.uncovered_stop		X	X
canthelp.cant_find_stop		X	X
canthelp.no_buses_at_time		X	X
canthelp.from_equals_to		X	X

Slot names and values are

Slot name	Example values	User acts	System acts
route	61c	inform	request expl-confirm impl-confirm
from, to	--	--	request
from.stop to.stop	forbes	--	expl-confirm impl-confirm
from.neighborhood to.neighborhood	oakland	inform	expl-confirm impl-confirm
from.monument to.monument	cmu	inform	expl-confirm impl-confirm
from.desc to.desc	camp and duquesne	inform	expl-confirm impl-confirm
result.from.desc result.to.desc result.from.neighborhood result.to.neighborhood	(same as above)	--	schedule
date	--	--	request
date.day	today tomorrow monday	inform	expl-confirm impl-confirm
date.absmonth	1 … 12	inform	expl-confirm impl-confirm
date.absday	1 … 31	inform	expl-confirm impl-confirm
date.relweek	next	inform	expl-confirm impl-confirm
result.date.day result.date.absmonth result.date.absday	(same as above)	--	schedule
time	--	--	request
time.hour	1 … 12	inform	expl-confirm impl-confirm
time.minute	0 … 59	inform	expl-confirm

			impl-confirm
time.ampm	am, pm	inform	expl-confirm impl-confirm
time.arriveleave	arrive, leave	inform	expl-confirm impl-confirm
time.rel	next	inform	expl-confirm impl-confirm
from.time.hour from.time.minute from.time.ampm to.time.hour to.time.minute to.time.ampm result.from.time.hour result.from.time.minute result.from.time.ampm result.to.time.hour result.to.time.minute result.to.time.ampm	(same as above)	--	schedule

Approximate number of distinct values

Slot name	Approximate number of distinct values
route	100
from.desc to.desc	500-10000
from.neighborhood to.neighborhood	20-100
from.monument to.monument	50-500
date.day	9
date.absmonth	12
date.absday	31
date.relweek	1
time.hour	12
time.minute	60
time.ampm	2
time.arriveleave	2
time.rel	1

Examples of user acts:

Example transcription(s)	Example act
hello	hello()
goodbye	bye()
go back	goback()
start over restart	restart()
[empty]	null()
repeat	repeat()
next bus	nextbus()
previous bus	prevbus()
tell me my choices	tellchoices()

yes	affirm()
no	negate()
no i’m not leaving from mckeesport	deny(from.desc=mckeesport)
sixty one a	inform(route=61a)
leaving at eight a m	inform(time.hour=8, time.minute=0, time.ampm=am, time.arriveleave=leave)
leaving at eight a m on the sixty one a	inform(time.hour=8, time.minute=0, time.ampm=am, time.arriveleave=leave), inform(route=61a)
Where are you leaving from? forbes ave	inform(from.desc="forbes ave")
Where are you leaving from? mckeesport	inform(from.neighborhood=mckeesport) ***or*** inform(from.desc=mckeesport) ***[depends on system, see below]***
Where are you leaving from? c m u	inform(from.monument=cmu) ***or*** inform(from.desc=cmu) ***[depends on system, see below]***
Where are you leaving from? forbes and alger	inform(from.desc=”forbes,alger”) ***or*** inform(from.desc=”forbes and alger”) ***[depends on system, see below]***
the next 61c from forbes in oakland to mckeesport transportation center in mckeesport on the 61c	inform(time.rel=next), inform(route=61c), inform(from.desc=forbes, from.neighborhood=oakland), inform(to.monument=”mckeesport transportation center”, to.neighborhood=mckeesport) ***or*** inform(time.rel=next), inform(route=61c), inform(from.desc=”forbes in oakland”), inform(to.desc=”mckeesport transportation center in mckeesport”) ***[depends on system, see below]***
today	inform(date.day=today)
tonight	inform(date.day=today,time.ampm=pm)
tomorrow	inform(date.day=tomorrow)
tuesday	inform(date.day=tuesday)
july first	inform(date.absmonth=7, date.absday=1)
next tuesday	inform(date.day=tuesday, date.relweek=next)
sunday night	inform(date.day=sunday, time.ampm=pm)
tomorrow morning	inform(date.day=tomorrow, time.ampm=am)
now	inform(time.rel=next)
ten	inform(time.hour=10, time.minute=0)
ten p m	inform(time.hour=10, time.minute=0, time.ampm=pm)
ten thirty p m	inform(time.hour=10, time.minute=30, time.ampm=pm)
leaving at ten thirty p m	inform(time.hour=10, time.minute=30, time.ampm=pm, time.arriveleave=leave)
arrive by ten thirty p m	inform(time.hour=10, time.minute=30, time.ampm=pm, time.arriveleave=arrive)
noon	inform(time.hour=12, time.minute=0, time.ampm=pm)
midnight	inform(time.hour=12, time.minute=0, time.ampm=am)
this morning	inform(date.day=today, time.ampm=am)

Examples of system acts from group A systems:

System transcript	System dialog act(s)
Welcome to the CMU Let's Go bus information system. To get help at any time, just say Help or press zero. What can I do for you?	hello(), example(act=help), open-request()
What is your destination?	request(to)
I am an automated spoken dialogue system that can give you schedule information for bus routes in Pittsburgh's East End. You can ask me about the following buses: 28X, 54C, 56U, 61A, 61B, 61C, 61D and 64.	example(route=28x), example(route=54c), example(route=56u), example(route=61a), example(route=61b), example(route=61c), example(route=61d), example(route=64)
Alright I'm sorry, but the 61F no longer exists. I will try to find another bus you can take. Where are you leaving from?	ack(), example(route=28x), example(route=54c), example(route=56u), example(route=61a), example(route=61b), example(route=61c), example(route=61d), example(route=64), canthelp.route_doesnt_run(route=61f), request(from)
I need you to give me a short answer. Where do you wanna leave from?	bebrief(), request(from)
The 61C. Where are you leaving from?	impl-conf(route=61c), request(from)
Okay I'm sorry but I do not know where CENTRE AVENUE AT NORTH is located. Try using the closest intersection instead. Where would you like to leave from?	ack(), canthelp.cant_find_stop(from.desc=centre avenue at north), request(from)
ADELAIDE STREET. Did I get that right?	expl-conf(from.desc=adelaide street)
ANN IN EMSWORTH. Did I get that right?	expl-conf(from.desc=ann in emsworth)
Alright AVALON. Is this correct?	ack(), expl-conf(from.desc=avalon)
Alright I'm sorry but I do not have the schedule for buses serving BELLEVUE. Currently, I only cover some routes in the following neighborhoods: Downtown, Oakland, Shadyside, Squirrel Hill, Homestead, McKeesport, and the airport.	ack(), example(from.neighborhood=downtown), example(from.neighborhood=oakland), example(from.neighborhood=shadyside), example(from.neighborhood=squirrel hill), example(from.neighborhood=homestead), example(from.neighborhood=mckeesport), example(from.neighborhood=airport), canthelp.uncovered_stop(from.desc=bellevue)
Alright I'm sorry but I do not know where HILTON HOTEL is located. Try using the closest intersection instead. Where do you wanna leave from?	ack(), canthelp.cant_find_stop(from.desc=hilton hotel), request(from)
Leaving from CRAIG. Is this correct?	expl-conf(from.desc=craig)
Okay Leaving from AIRPORT. Is this correct?	ack(), expl-conf(from.desc=airport)
If you want to leave from AIRPORT say yes or press one, otherwise say no or press three.	expl-conf(from.desc=airport)
The 61C. Leaving from DOWNTOWN. Did I get that right?	impl-conf(route=61c), expl-conf(from.desc=downtown)
So far, I know that you want to take the 61A I think you said you wanted to leave from EIGHTEENTH. I need you to confirm that. If you want to leave from EIGHTEENTH say yes or press one, otherwise say no or press three.	impl-conf(route=61a), expl-conf(from.desc=eighteenth), expl-conf(from.desc=eighteenth)
I'm sorry, but the 61C does not go between SIXTH AVENUE and HOMESTEAD. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	canthelp.no_connection(route=61c, from.desc=sixth avenue, to.desc=homestead), example(act=restart)
Alright Just a minute. I'll look that up. PITTSBURGH and DOWNTOWN PITTSBURGH are both the same stop. Please provide a different start or end point.	ack(), hold-on(), canthelp.from_equals_to(from.desc=pittsburgh, to.desc=downtown pittsburgh)

Hold on. I'll look that up. I'm sorry, I don't know any routes that go from MCKEESPORT to NEVILLE. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	hold-on(), canthelp.no_connection(from.desc=mckeesport, to.desc=neville), example(act=restart)
At what time do you wanna travel?	request(time)
Going to AIRPORT. When would you like to travel?	impl-conf(to.desc=airport), request(time)
When are you going to take that bus?	request(time)
For example, you can say, 10:30 in the morning, I want the next bus, or say start over to restart.	example(time.hour=10, time.minute=30, time.ampm=am), example(time.rel=next), example(act=restart)
I think you want the next bus. Am I right?	expl-conf(time.rel=next)
Alright Arriving by 9 a.m.. Did I get that right?	ack(), expl-conf(time.arriveleave=arrive, time.hour=9, time.minute=0, time.ampm=pm)
Alright I think you want the next bus. Am I right?	ack(), expl-conf(time.rel=next)
Arriving by 1 a.m.. Did I get that right?	expl-conf(time.arriveleave=arrive, time.hour=1, time.minute=0, time.ampm=pm)
Okay Arriving by 4 a.m.. Is this correct?	ack(), expl-conf(time.arriveleave=arrive, time.hour=4, time.minute=0, time.ampm=pm)
I need you to give me a short answer. Leaving at 8 p.m.. Is this correct?	bebrief(), expl-conf(time.arriveleave=leave, time.hour=8, time.minute=0, time.ampm=pm)
If you want to leave at 1 a.m. say yes or press one, otherwise say no or press three.	expl-conf(time.arriveleave=leave, time.hour=1, time.minute=0, time.ampm=pm)
Alright Hold on. I'll look that up. The next 61C leaves FORBES AVENUE OPPOSITE MOREWOOD CARNEGIE MELLON at 6 57 p.m. and arrives at MURRAY AVENUE AT DARLINGTON at 7 12 p.m.. To get more information about buses related to this trip, you can say, when is the next bus, or, when is the previous bus. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	ack(), hold-on(), schedule(result.route=61c, result.from.desc=forbes avenue opposite morewood carnegie mellon, result.from.time.hour=6, result.from.time.minute=57, result.from.time.ampm=pm, result.to.desc=murray avenue at darlington, result.to.time.hour=7, result.to.time.minute=12, result.to.time.ampm=pm), example(act=nextbus), example(act=prevbus), example(act=restart), example(act=bye)
Alright Hold on. I'll look that up. The next 61D leaves FIFTH AVENUE AT BIGELOW PITT at 5 37 a.m. and arrives at FIFTH AVENUE AT MARKET at 5 52 a.m.. To get more information about buses related to this trip, you can say, when is the next bus, or, when is the previous bus. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	ack(), hold-on(), schedule(result.route=61d, result.from.desc=fifth avenue at bigelow pitt, result.from.time.hour=5, result.from.time.minute=37, result.from.time.ampm=am, result.to.desc=fifth avenue at market, result.to.time.hour=5, result.to.time.minute=52, result.to.time.ampm=am), example(act=nextbus), example(act=prevbus), example(act=restart), example(act=bye)
I looked for a 61C leaving KENNYWOOD for DOWNTOWN at around 7 a.m. I found a 61C leaving KENNYWOOD BOULEVARD OPPOSITE MIFFLIN at 7 oh 2 a.m.. and reaching FIFTH AVENUE AT MARKET at 8 oh 3 a.m..	schedule(route=61c, from.desc=kennywood, to.desc=downtown, result.route=61c, result.from.desc=kennywood boulevard opposite mifflin, result.from.time.hour=7, result.from.time.minute=2, result.from.time.ampm=am, result.to.desc=fifth avenue at market, result.to.time.hour=8, result.to.time.minute=3, result.to.time.ampm=am)
Just a second. There is a 28X leaving PITTSBURGH INTERNATIONAL AIRPORT LOWER LEVEL DOOR 6 at 4 20 p.m.. It will arrive at FORBES AVENUE AT BIGELOW SCHENLEY DRIVE A at 5 16 p.m.. To get more information about buses related to this trip, you can say, when is the next bus, or, when is the previous bus. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	hold-on(), schedule(result.route=28x, result.from.desc=pittsburgh international airport lower level door 6, result.from.time.hour=4, result.from.time.minute=20, result.from.time.ampm=pm, result.to.desc=forbes avenue at bigelow schenley drive a, result.to.time.hour=5, result.to.time.minute=16, result.to.time.ampm=pm), example(act=nextbus), example(act=prevbus), example(act=restart), example(act=bye)
Okay. There is a 28X leaving FIFTH AVENUE OPPOSITE ATWOOD CHILDRENS HOSPITAL B at 4 30 a.m.. It will arrive at PITTSBURGH INTERNATIONAL AIRPORT LOWER LEVEL DOOR 6 at 5 12 a.m.. To get more information about buses related to this trip, you can say, when is the	hold-on(), schedule(result.route=28x, result.from.desc=fifth avenue opposite atwood childrens hospital b, result.from.time.hour=4, result.from.time.minute=30, result.from.time.ampm=am, result.to.desc=pittsburgh international airport lower level door 6, result.to.time.hour=5, result.to.time.minute=12,

next bus, or, when is the previous bus. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	result.to.time.ampm=am), example(act=nextbus), example(act=prevbus), example(act=restart), example(act=bye)
I looked for a 61A leaving HAZELWOOD for NORTH BRADDOCK at around 7 30 p.m. However, I don't know any routes that go from HAZELWOOD to NORTH BRADDOCK.	canthelp.no_connection(route=61a, from.desc=hazelwood, to.desc=north braddock, from.desc=hazelwood, to.desc=north braddock)
Leaving at 11 a.m.. Hold on. Let me check that for you. I'm sorry, but the 61C does not go between THE ZOO and WOOD. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	impl-conf(time.arriveleave=leave, time.hour=11, time.minute=0, time.ampm=pm), hold-on(), canthelp.no_connection(route=61c, from.desc=the zoo, to.desc=wood), example(act=restart)
Alright Hold on. I'll look that up. I'm sorry, but the 61A does not go between KENNEDY and DOWNTOWN PITTSBURGH. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	ack(), hold-on(), canthelp.no_connection(route=61a, from.desc=kennedy, to.desc=downtown pittsburgh), example(act=restart)
Alright Hold on. I'll look that up. OAKLAND and PITT are both the same stop. Please provide a different start or end point. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	ack(), hold-on(), canthelp.from_equals_to(from.desc=oakland, to.desc=pitt), example(act=restart)
Alright Hold on. Let me check that for you. I'm sorry, I don't know any routes that go from MCKEESPORT to EIGHTEENTH. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	ack(), hold-on(), canthelp.no_connection(from.desc=mckeesport, to.desc=eighteenth), example(act=restart)
Alright Hold on. Let me check that for you. I'm sorry, but the 61A does not run between BRADDOCK and DOWNTOWN PITTSBURGH. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	ack(), hold-on(), canthelp.no_connection(route=61a, from.desc=braddock, to.desc=downtown pittsburgh), example(act=restart)
Just a second. I'm sorry, but that was the first 28X of the day between FIFTH AND ATWOOD and AIRPORT. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	hold-on(), canthelp.no_buses_at_time(route=28x, from.desc=fifth and atwood, to.desc=airport), example(act=restart), example(act=restart)
Just a second. I'm sorry, but that was the last 54C of the day between ATWOOD STREET and EIGHTEENTH. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	hold-on(), canthelp.no_buses_at_time(route=54c, from.desc=atwood street, to.desc=eighteenth), example(act=restart)
	null()
Alright	ack()
Alright Hold on. I'll look that up.	ack(), hold-on()
Are you still there?	are-you-there()
Are you sure you want to start over?	expl-conf(act=restart)
Hold on. Let me check that for you.	hold-on()
I need you to give me a short answer.	bebrief()
Just a minute. I'll look that up.	hold-on()
Okay	ack()
Okay, let's start from the beginning.	restart()
Okay, let's start from the beginning. What can I do for you?	restart(), open-request()

Okay. I'm sorry, I seem to have had an internal error while handling your request. Please try again later.	hold-on(), canthelp.system_error()

Examples of system acts from group B systems:

System transcript	System dialog act(s)
East Pittsburg Bus Schedules. Say a bus route, like 28X, or say I'm not sure.	hello(), request(route), example(route=28x), example(route=dont_know)
Say a bus route like 56U, or say I'm not sure.	request(route), example(route=56u), example(route=dont_know)
Oh, sorry, which route do you want?.	sorry(), request(route)
Going back. Say a bus route, like 28X, or say I'm not sure.	goback(), request(route), example(route=28x), example(route=dont_know)
I THOUGHT you said eighty four a, but I don't have schedules for that route. I only have schedules for the 28 X, 54 C, 56 U, 61 A B C and D, and the 64. You can say one of these routes, say I'm not sure which route, or just hang up.	impl-conf(route=84a), canthelp.uncovered_route(route=84a), example(route=28x), example(route=54c), example(route=56u), example(route=61a), example(route=61b), example(route=61c), example(route=61d), example(route=64), request(route), example(route=dont_know), example(act=hangup)
Ok, going back to the beginning. Say a bus route, like 28X, or say I'm not sure.	restart(), request(route), example(route=28x), example(route=dont_know)
28X. Is that right?	expl-conf(route=28x)
Sorry, do you want route 28X? Please say yes or no.	sorry(), expl-conf(route=28x), example(act=negate), example(act=affirm)
Say the East Pittsburgh neighborhood you're leaving from, like Mount Oliver or Swissvale, or say: TELL ME MY CHOICES.	sorry(), request(from.neighborhood), example(from.neighborhood=mount oliver), example(from.neighborhood=swissvale), example(act=tellchoices)
No problem. Where are you leaving from?	ack(), request(from)
Okay, 28X. To change, say go back. Where are you leaving from?	impl-conf(route=28X), example(act=goback), request(from)
Okay, 28X. Where are you leaving from?	impl-conf(route=28X), request(from)
Okay, 28X. You can always say go back. Where are you leaving from?	impl-conf(route=28X), example(act=goback), request(from)
Say the neighborhood or intersection or place you're leaving from like bell avenue.	request(from), example(from.desc=bell avenue)
Where are you leaving from?	request(from)
downtown. Is that right?	expl-conf(from.desc=downtown)
mckeesport transportation center. Is that right?	expl-conf(from.desc=mckeesport transportation center)
I heard carrick, but I don't have any stops in that neighborhood. I only know about certain neighborhoods. Say the neighborhood or place you're leaving from, or you can just hang up.	impl-conf(from.neighborhood=carrick), canthelp.uncovered_stop(from.neighborhood=carrick), request(from), example(act=hangup)
If you're leaving from hazelwood and murray in greenfield, Say yes, if not say no.	expl-conf(from.desc=hazelwood and murray in greenfield), example(act=affirm), example(act=negate)
If you're leaving from schenley, Say yes, if not say no.	expl-conf(from.desc=schenley), example(act=affirm), example(act=negate)
Sorry, are you leaving from pittsburgh? Please say yes or no.	sorry(), expl-conf(from.desc=pittsburgh), example(act=affirm), example(act=negate)
Okay, c m u. And where are you going to?	impl-conf(from.desc=c m u), request(to)
Say the East Pittsburgh neighborhood you're going to, like Lawrenceville, or say: TELL ME MY CHOICES.	request(to.neighborhood), example(to.neighborhood=lawrenceville),

	example(act=tellchoices)
And where are you going to?	request(to)
Going back. Where are you going to?	goback(), request(to)
Oh, sorry, where are you going to?	sorry(), request(to)
Say the neighborhood or intersection or place you're going to, like waterfront.	request(to), example(to.desc=waterfront)
The routes I cover have stops in the following neighborhoods. When you hear the one you're going to, just say it. the airport. allentown. bloomfield. bon air. braddock. carnegie mellon. crafton. downtown. duquesne. east allegheny. east liberty. east pittsburgh. edgewood. greenfield. hazelwood. homestead. knoxville. lawrenceville. mckeesport. moon township. mount oliver. mount washington. north braddock. north side. oakland. point breeze. polish hill. rankin. regent square. shadyside. sheraden. south side. squirrel hill. the strip. swissvale. uptown. west homestead. west mifflin. wilkinsburg. That's it. Say the one you're going to, or to hear them again say: TELL ME MY CHOICES.	example(to.neighborhood=the airport), example(to.neighborhood=allentown), example(to.neighborhood=bloomfield), example(to.neighborhood=bon air), example(to.neighborhood=braddock), example(to.neighborhood=carnegie mellon), example(to.neighborhood=crafton), example(to.neighborhood=downtown), example(to.neighborhood=duquesne), example(to.neighborhood=east allegheny), example(to.neighborhood=east liberty), example(to.neighborhood=east pittsburgh), example(to.neighborhood=edgewood), example(to.neighborhood=greenfield), example(to.neighborhood=hazelwood), example(to.neighborhood=homestead), example(to.neighborhood=knoxville), example(to.neighborhood=lawrenceville), example(to.neighborhood=mckeesport), example(to.neighborhood=moon township), example(to.neighborhood=mount oliver), example(to.neighborhood=mount washington), example(to.neighborhood=north braddock), example(to.neighborhood=north side), example(to.neighborhood=oakland), example(to.neighborhood=point breeze), example(to.neighborhood=polish hill), example(to.neighborhood=rankin), example(to.neighborhood=regent square), example(to.neighborhood=shadyside), example(to.neighborhood=sheraden), example(to.neighborhood=south side), example(to.neighborhood=squirrel hill), example(to.neighborhood=the strip), example(to.neighborhood=swissvale), example(to.neighborhood=uptown), example(to.neighborhood=west homestead), example(to.neighborhood=west mifflin), example(to.neighborhood=wilkinsburg), request(to.neighborhood), example(act=tellchoices)
I heard allentown, but I don't have any routes that run from ingram station to allentown. I only know about certain routes. Say the neighborhood or place you're going to, or you can just hang up.	impl-conf(to.desc=allentown), canthelp.no_connection(from.desc=ingram station, to.desc=allentown), request(to.neighborhood), example(act=hangup)
Do you want times for the next few buses? Say yes or no.	expl-conf(act=want-next-bus), example(act=affirm), example(act=negate)
Ok, say just the DAY you want, like today, tomorrow, or June 15th.	ack(), request(date), example(date.day=today), example(date.day=tomorrow), example(date.absmonth=6, date.absday=15)
Okay, edgewood. Do you want times for the next few buses? Say yes or no.	impl-conf(to.desc=edgewood), expl-conf(act=want-next-bus), example(act=affirm), example(act=negate)

Oh sorry, say just the DAY you want like today, Tuesday, or July 1st.	sorry(), request(date), example(date.day=today), example(date.day=tuesday), example(date.absmonth=7, date.absday=1)
Sorry, say the DAY you want to take the bus, like tonight, Monday, or July 1st.	sorry(), request(date), example(date.day=today, time.ampm=pm), example(date.day=monday), example(date.absmonth=7, date.absday=1)
august twenty second. Is that right?	expl-conf(date.absmonth=8, date.absday=22)
Say just the TIME you want, like DEPART AT 8 AM or ARRIVE BY 5:30 PM.	request(time), example(time.arriveleave=leave, time.hour=8, time.minute=0, time.ampm=am), example(time.arriveleave=arrive, time.hour=5, time.minute=30, time.ampm=pm)
arrive at five a.m.. Is that right?	expl-conf(time.arriveleave=arrive, time.hour=5, time.minute=0, time.ampm=am)
depart at eight p.m.. Is that right?	expl-conf(time.arriveleave=leave, time.hour=8, time.minute=0, time.ampm=pm)
leave at two p.m.. Is that right?	expl-conf(time.arriveleave=leave, time.hour=2, time.minute=0, time.ampm=pm)
leave now. Is that right?	expl-conf(time.arriveleave=leave, time.rel=next)
Sorry, do you want to leave at nine a.m.? Please say yes or no.	sorry(), expl-conf(time.arriveleave=leave, time.hour=9, time.minute=0, time.ampm=am), example(act=affirm), example(act=negate)
Ok, the next few buses. To change, say go back. At 10:03 PM today, there is a 64, from Penn Avenue And Forty Forth Street, Near Childrens Hospital, In Lawrenceville, arriving Waterfront Drive And Loew's Theatre, In Homestead, at 10:38 PM. Say repeat, next, previous, or start over.	impl-conf(act=want-next-bus), example(act=goback), schedule(result.from.time.hour=10, result.from.time.minute=3, result.from.time.ampm=pm, result.from.date.day=today, result.route=64, result.from.desc=penn avenue and forty forth street, near childrens hospital, in lawrenceville, result.to.desc=waterfront drive and loew's theatre, in homestead, result.to.time.hour=10, result.to.time.minute=38, result.to.time.ampm=pm), example(act=repeat), example(act=nextbus), example(act=prevbus), example(act=restart)
Ok, the next few buses. To change, say go back. There are more buses than just the 28X, that run from boquet street to schenley. I'll tell you about all of them. At 7:38 AM today, there is a 61 D, from Forbes Avenue And Boquet Street, arriving Forbes Avenue And Bigelow Boulevard And Schenley Drive, In Oakland, at 7:42 AM. Say repeat, next, previous, or start over.	impl-conf(act=want-next-bus), example(act=goback), morebuses(route=28x, from.desc=boquet street, to.desc=schenley), schedule(result.from.time.hour=7, result.from.time.minute=38, result.from.time.ampm=am, result.from.date.day=today, result.route=61d, result.from.desc=forbes avenue and boquet street, result.to.desc=forbes avenue and bigelow boulevard and schenley drive, in oakland, result.to.time.hour=7, result.to.time.minute=42, result.to.time.ampm=am), example(act=repeat), example(act=nextbus), example(act=prevbus), example(act=restart)
There are more buses than just the 54C, that run from oakland to south side. I'll tell you about all of them. At 11:25 AM tomorrow, there is a 54 C, from Centre Avenue And Craig Street, In Oakland, arriving East Carson Street And Twentieth Street, In South Side, at 11:56 AM. Say repeat, next, previous, or start over.	morebuses(route=54c, from.desc=oakland, to.desc=south side), schedule(result.from.time.hour=11, result.from.time.minute=25, result.from.time.ampm=am, result.from.date.day=tomorrow, result.route=54c, result.from.desc=centre avenue and craig street, in oakland, result.to.desc=east carson street and twentieth street, in south side, result.to.time.hour=11, result.to.time.minute=56, result.to.time.ampm=am), example(act=repeat), example(act=nextbus), example(act=prevbus), example(act=restart)
At 10:07 AM tomorrow, there is a 61 C, from Fifth Avenue And Market Street, Downtown, arriving Kennywood Boulevard And Hoffman Boulevard, In West Mifflin, at 10:58 AM. Say repeat, next, previous, or start over.	schedule(result.from.time.hour=10, result.from.time.minute=7, result.from.time.ampm=am, result.from.date.day=tomorrow, result.route=61c, result.from.desc=fifth avenue and market street, downtown,

	result.to.desc=kennywood boulevard and hoffman boulevard, in west mifflin, result.to.time.hour=10, result.to.time.minute=58, result.to.time.ampm=am), example(act=repeat), example(act=nextbus), example(act=prevbus), example(act=restart)
At 10:15 PM today, there is a 61 A, from Kelly Avenue And Hay Street, In Wilkinsburg, arriving Bell Avenue And Library, Near Jones Avenue, In North Braddock, at 10:35 PM. Say repeat, next, previous, or start over.	schedule(result.from.time.hour=10, result.from.time.minute=15, result.from.time.ampm=pm, result.from.date.day=today, result.route=61a, result.from.desc=kelly avenue and hay street, in wilkinsburg, result.to.desc=bell avenue and library, near jones avenue, in north braddock, result.to.time.hour=10, result.to.time.minute=35, result.to.time.ampm=pm), example(act=repeat), example(act=nextbus), example(act=prevbus), example(act=restart)
At 1:07 AM later tonight, there is a 61 C, from Fifth Avenue And Market Street, Downtown, arriving Lysle Boulevard And Mckeesport Transportation Center, at 1:58 AM. Say repeat, next, previous, or start over.	schedule(result.from.time.hour=1, result.from.time.minute=7, result.from.time.ampm=am, result.from.date.day=today, result.route=61c, result.from.desc=fifth avenue and market street, downtown, result.to.desc=lysle boulevard and mckeesport transportation center, result.to.time.hour=1, result.to.time.minute=58, result.to.time.ampm=am), example(act=repeat), example(act=nextbus), example(act=prevbus), example(act=restart)
At 5:15 PM earlier today, there was a 61 C, from Kennywood Boulevard Opposite Hoffman Boulevard, In West Mifflin, arriving Kennywood Park And Main Entrance, at 5:28 PM. Say repeat, next, previous, or start over.	schedule(result.from.time.hour=5, result.from.time.minute=15, result.from.time.ampm=pm, result.from.date.day=today, result.route=61c, result.from.desc=kennywood boulevard opposite hoffman boulevard, in west mifflin, result.to.desc=kennywood park and main entrance, result.to.time.hour=5, result.to.time.minute=28, result.to.time.ampm=pm), example(act=repeat), example(act=nextbus), example(act=prevbus), example(act=restart)
At 7:55 AM on Sunday August twenty second,, there is a 28 X, from Forbes Avenue And Morewood Avenue, Near Carnegie Mellon, In Oakland, arriving Pittsburgh International Airport Lower Level, Near Door 6, at 8:42 AM. Say repeat, next, previous, or start over.	schedule(result.from.time.hour=7, result.from.time.minute=55, result.from.time.ampm=am, result.from.date.day=sunday, result.from.date.absmonth=8, result.from.date.absday=22, result.route=28x, result.from.desc=forbes avenue and morewood avenue, near carnegie mellon, in oakland, result.to.desc=pittsburgh international airport lower level, near door 6, result.to.time.hour=8, result.to.time.minute=42, result.to.time.ampm=am), example(act=repeat), example(act=nextbus), example(act=prevbus), example(act=restart)
Next. At 10:00 PM today, there is a 61 B, from Eleventh Street And Woodlawn Street, In Braddock, arriving Fifth Avenue And Market Street, Downtown, at 10:45 PM. Say repeat, next, previous, or start over.	impl-conf(act=nextbus), schedule(result.from.time.hour=10, result.from.time.minute=0, result.from.time.ampm=pm, result.from.date.day=today, result.route=61b, result.from.desc=eleventh street and woodlawn street, in braddock, result.to.desc=fifth avenue and market street, downtown, result.to.time.hour=10, result.to.time.minute=45, result.to.time.ampm=pm), example(act=repeat), example(act=nextbus), example(act=prevbus), example(act=restart)
Here's that again. At 10:12 PM today, there is a 54 C, from South Hills Junction And Bus Turnaround, In Mount Washington, arriving Centre Avenue And Craig Street, at 10:45 PM. Say repeat, next, previous, or start over.	impl-conf(act=repeat), schedule(result.from.time.hour=10, result.from.time.minute=12, result.from.time.ampm=pm, result.from.date.day=today, result.route=54c, result.from.desc=south hills junction and bus turnaround, in mount washington, result.to.desc=centre avenue and craig street, result.to.time.hour=10, result.to.time.minute=45, result.to.time.ampm=pm), example(act=repeat),

	example(act=nextbus), example(act=prevbus), example(act=restart)
Here's that again. At 9:10 AM tomorrow, there is a 28 X, from Seventh Avenue And Smithfield Street, Downtown, arriving Pittsburgh International Airport Lower Level, Near Door 6, at 9:42 AM. Say repeat, next, previous, or start over.	impl-conf(act=repeat), schedule(result.from.time.hour=9, result.from.time.minute=10, result.from.time.ampm=am, result.from.date.day=tomorrow, result.route=28x, result.from.desc=seventh avenue and smithfield street, downtown, result.to.desc=pittsburgh international airport lower level, near door 6, result.to.time.hour=9, result.to.time.minute=42, result.to.time.ampm=am), example(act=repeat), example(act=nextbus), example(act=prevbus), example(act=restart)
I thought you said GOODBYE. If that's not right, say go back. Otherwise, thanks for calling.	impl-conf(act=bye), example(act=goback)
I thought you said START OVER. Is that right?	expl-conf(act=restart)

Group C uses the same dialog acts as group A and B; examples will be made available later in the challenge.

For the *from.** and *to.** slots in user acts, and the *from.**, *to.**, *result.from.** and *result.to.** slots in system acts:

- Group A dialog systems *sometimes* distinguished between neighborhoods, streets, and monuments. Group A dialog systems use *.desc, *.neighborhood, and *.monument.
- Group B dialog systems *never* distinguished between neighborhoods, streets, and monuments; thus group B dialog systems *always* use *.desc and *never* *.neighborhood, and *.monument.
- In *user* acts, Group C dialog systems *always* distinguished between neighborhoods, streets, and monuments; thus group C dialog systems *always* use *.desc, *.neighborhood, and *.monument. In *system* acts, Group C dialog systems *sometimes* distinguish between neighborhoods, streets, and monuments; thus group C dialog systems use a mixture of *.stop, *.neighborhood, and *.monument, *and* *.desc.

Appendix E: Labeling procedure

The goal of labeling is to specify whether each dialog act in each SLU hypothesis is correct or incorrect. To do this efficiently, a four-pass approach was used.

The first two phases deal with transcription. In the first phase, crowd-workers listen to the user audio, are shown the 1-best recognition result, and mark the recognition result as correct or incorrect. In the second phase, items marked “incorrect” are transcribed.

The third and fourth phases deal with labeling. These phases do not use the utterance audio. In the third phase, crowd-workers look at transcriptions (and recent dialog history) and indicate whether they contain a route, a *from* location, a *to* location, a date, or a time (5 yes/no judgments). In this phase, common transcriptions such as “yes”, “no”, “next bus”, and common routes and locations can be handled automatically. In the fourth phase, crowd-workers look at transcriptions which were labeled as having a given slot present (route, from.desc, from.neighborhood, from.monument, to.desc, to.neighborhood, to.monument, date, time), the previous 2 turns and the next 2 turns. They are shown all of the slot values recognized for that slot in that utterance and mark each as correct or incorrect (binary judgment for each slot value). For transcriptions which are completely unambiguous, such as utterances that include only a bus route such as “sixty one c”, this step can be done automatically.

When deciding if a slot value is correct or incorrect, labelers were instructed to mark items which complete captured the *meaning* of the user. Here are examples:

Example CORRECT labels

Transcription	All recognized concept value(s)	Reason
hazelwood at emahlea	location=hazelwood and emahlea	Refers to same intersection
next next	cmd=next	Same meaning, just repeated
ingram bus station	location=ingram station	Same location
next sorry trying to catch these busses	cmd=next	Same intention
mckeesport pa	location=mckeesport	Same place (and "pa" should be "p a")
dynamo way	location=dynamo alley	These are actually the same place
going back	cmd=go back	Same meaning
back go back	cmd=go back	Same meaning
oh back	cmd=go back	Same meaning
what (asked like a question)	cmd=repeat	Same meaning
braddock pennyslvania	location=braddock	Same location
the waterfront	location=the water front	Same meaning
fifteen charles street	route=fifteen	Refers to the same bus line
shady side	location=shadyside	Same meaning
fifty nine mon valley	route=fifty nine	Same bus route
fifty nine mon valley	route=the fifty nine	Same bus route
pittsburgh downtown pittsburgh	location=pittsburgh downtown	Same location
pittsburgh downtown pittsburgh	location=downtown	Same location
oh go back you're wrong you damn computer go back	cmd=go back	Complete and correct, after the fluff is removed
july twenty third friday	date=july twenty third	safe to assume that July 23rd IS friday
six one c	route=sixty one c	61C=61C -- same bus route
hawkins fourth and hawkins	location=fourth and hawkins	matched the most specific intent expressed by user

Example INCORRECT labels

Transcription	Recognized concept value(s)	Reason
braddock north braddock braddock	location=braddock	Braddock is only one of the locations the caller said
go back start over	cmd=start over	Correct but incomplete
no i'm not sure	cmd=no	"no" is correct but incomplete
li no liberty	cmd=no	"no" is correct but incomplete
um from rockin um from bell and jones	from=bell and jones avenue	Correct but incomplete (“from rockin” missing)
i'm not sure twenty eight x	route=i'm not sure	Correct but incomplete
uh beechwood boulevard and murray avenue	location=beechwood boulevard	Correct but incomplete
arlington and brownsville	location=arlington ave	(probably) correct, but incomplete
arlington and brownsville	location=arlington avenue	(probably) correct, but incomplete
for grant street downtown	location=grant street	Correct but incomplete
forbes downtown	location=downtown	Correct but incomplete
woodstock in washington	location=woodstock ave	Missed neighborhood
woodstock in washington	location=woodstock avenue	Missed neighborhood

forbes and murray	location=forbes ave	Incomplete
penn wood apartments in wilkinsburg	location=i'm at wilkinsburg	Meaning is correct but incomplete
penn wood apartments in wilkinsburg	location=wilkinsburg	correct but incomplete
seven a m	time=seven	correct but incomplete
sixty one a b c d	route=sixty one a	user wants any of the 61A, 61B, 61C, 61D -- so 61A is correct but incomplete
thirty sixth and liberty or thirty sixth and penn	location=thirty sixth and liberty	Correct but incomplete
hawkins village on south braddock avenue	location=hawkins village in rankin	user didn't say "rankin"
um from rockin um from bell and jones	from=bell and jones in north braddock	user didn't say "north braddock"
bell and jones	from=bell avenue in north braddock	user didn't say "north braddock"
twenty sixth and liberty	location=twenty fifth and liberty	close but no cigar
uh beechwood boulevard and murray avenue	location=beechwood boulevard and Ronald	ronald is wrong
goes from the busway	from=east busway	user didn't say "east" busway
arlington and brownsville	location=arlington and freeland	Different intersection
sixty one c	route=sixty one b	Different bus routes
penn wood apartments in wilkinsburg	location=pennwood avenue	Sounds similar, but different places
i'm going to east pittsburgh	to=pittsburgh	different neighborhoods
braddock pennyslvania	location=braddock avenue	braddock is a neighborhood; braddock ave is a street
braddock pennyslvania	location=braddock and verona	totally wrong
to mckeesport	from=mckeesport	Wrong direction
i'm going in to mckeesport from jefferson hills nay a lodge road	to=mckeesport transportation center	No reference to "transportation center"
thirty sixth and liberty or thirty sixth and penn	location=twenty sixth and liberty	Recognized intersection isn't one asked for
ok depart um	time=now	caller didn't ask for "now"
leaving from downtown	to=downtown	Wrong direction

Appendix F: Bus timetable database

Participants may wish to use the bus timetable database. A copy of the data is available here:

http://research.microsoft.com/en-us/events/dstc/

Note that this is the version of the bus timetable database that was active during Summer 2010 (and used by datasets train2, train3, and test4). The bus timetable database changes over time. From 2008-2012, there are approximately 10 different versions of the database. Also, note that each system had its own interface layer to the database, so no effort is made to align the SLU label classes with database tokens.

Appendix G: Notes for future challenges

Below are ideas and suggestions which could not be accommodated in this challenge task, but may be good for future challenges

- It would be interesting to report results with varying amounts of training data. This was not done for this challenge because of the extra overhead, and because there is limited data to start.
- It would be desirable to label dialog states as "correct" if they would retrieve an itinerary from the DB that is acceptable to the user. This was not done for this challenge to remove the dependency of the evaluation on the database (and may still require subjective judgements).
- This challenge has assumed that the user's goal is fixed throughout the dialog (at least until "start over"). Even if changes are rare, it is possible that the user's goal may change. It would be preferable to manually search for changes in the user's goal rather than assuming it is fixed. However this process is very difficult because (1) these changes are often difficult to identify at all, especially without local domain knowledge, (2) these changes often require highly subjective judgments, and (3) performing goal labeling on the scale of this challenge is expensive, even with crowd-sourcing.