6th Dialogue System Technology Challenge

Track #2
End-to-end Conversation Modeling Track

1. Task Description
   This track consists of two tasks, i.e., main and pilot tasks as follows:

   (1) Main task (mandatory): Customer service dialog using Twitter
   (*) The tools to download the twitter data and transform the data to the dialog format are provided.
   The training data will be collected from Aug. 1st to 31st

   Task A: Full or part of the training data will be used to train conversation models.
   Task B: Any open data, e.g. from web, are available as external knowledge to generate informative sentences.
   But they should not overlap with the training, validation and test data provided by organizers.

   (2) Pilot task: Movie scenario dialog using OpenSubtitle

2. Data collection

2.1 Twitter data

In the twitter task, we use dialog data collected from multiple twitter accounts for customer services, where each dialogue consists of real tweets between a customer and an agent. A customer usually asks a question or complains something about a product or a service of the company, and an agent responds to the customer accordingly. In this challenge, each participant is supposed to develop a dialog system that mimics agents' behaviors. The system will be evaluated based on the quality of automatically generated sentences in response to customers' tweets.

For the challenge, we provide a data collection tool to all participants so that the participants can collect the data by themselves because it is basically not allowed to distribute twitter data to third parties. In this task, it is assumed that each participant continues to collect the data from specific accounts in the challenge period. To acquire a large amount of data, the data collection needs to be done repeatedly, e.g. by running the script once a day, because the amount of data we can download is limited and older tweets cannot be accessed as time goes by.

At a certain point of time, we will provide an additional tool to extract subsets of collected data for training, development (validation), and evaluation so that all the

participants can use the same data for the challenge. Until the official data sets are fixed, trial data sets are available to develop dialog systems, which are selected from the data collected by each participant. But once the official data sets are determined, the system needs to be trained from scratch only using the official data sets.

We plan to determine the training, development and evaluation sets with the following amounts, where there are no overlaps between the data sets.

	#twitter accounts	#dialog	#utterance	Estimated release date
Training set	943	$\approx 1M$	$\approx 2.2M$	9/1
Development set	118	$\approx 110K$	$\approx 280K$	9/1
Evaluation set	118	≈ 500	≈ 1200	9/18

Participants need to use data collection tool "collect_twitter_dialogs" included in the provided package "DSTC6_ConversationModelTask".
Necessary steps are written in "collect_twitter_dialogs/README.md"

The trial data sets can be extracted from downloaded twitter dialogs using a data extraction script: make_trial_data.sh in 'tasks/twitter'.

2.2 OpenSubtitles2016

Open subtitles data is a collection of movie subtitles. We use English subtitles on the web site:

http://opus.lingfil.uu.se/OpenSubtitles2016.php

The tar file can be downloaded via URL:
http://opus.lingfil.uu.se/download.php?f=OpenSubtitles2016/en.tar.gz

and decompressed by

```
tar zxvf en.tar.gz
```

The file size of "en.tar.gz" is approximately 18GB. It will need more disk space for storing the decompressed files.
The trial data sets can be extracted from the decompressed directory using a data extraction script: make_trial_data.sh in 'tasks/opensubs'.

	#dialog	#utterance	Estimated release date
Training set	$\approx 30M$	$\approx 60M$	9/1
Development set	$\approx 300K$	$\approx 600K$	9/1
Evaluation set	≈ 500	≈ 1000	9/18

3. Data preprocessing

Twitter dialogs and OpenSubtitles contain a lot of noisy text with specific expressions. Therefore, text preprocessing is important to clean up and normalize the text. Moreover, all the participants need to use the same preprocessing at least for target references to assure fare comparisons between different systems in the Challenge.

2.1 Twitter data

Twitter data contains a lot of specific information such as twitter account names, URLs, e-mail addresses, telephone/tracking numbers and hashtags. This kind of information is almost impossible to predict correctly unless we use a lot of training data obtained from the same site. To alleviate this difficulty, we substitute those strings with abstract symbols such as <URL>, <E-Mail>, and <NUMBERS> using a set of regular expressions. In addition, since each tweet usually starts with a twitter account name of the recipient, we just remove the account name. But if such names appear within a sentence, we leave them because those names are used as a part of sentence. We also leave hashtags because of the same reason. We also substitute user names with <USER>, for example,

"Hi John, can you …" -> "Hi <USER>, can you …"

Actually, since the user's name can be extracted from the attribute information of each tweet, we can replace it. Note that the text preprocessing is not perfect, and therefore there may remain original phrases, which are not replaced or removed successfully. The text also includes many abbreviations, e.g. "pls hlp", special symbols, e.g. "(-:" and wide characters "☆◎❤… ", but they are left unaltered. The wide characters are encoded with UTF-8 in the text file.

The preprocessing is performed by function "preprocess()" in python script "tasks/twitter/extract_twitter_dialogs.py".

2.2 OpenSubtitles data

We basically follow the method in https://arxiv.org/abs/1506.05869 to extract dialogs and preprocess the subtitle data. Consecutive two sentences are considered a dialog consisting of one question and one answer.

4. Dialog data format

4.1 Basic format

Each dialog consists of two or more utterances, where each line corresponds to an utterance given by user 'U:' or system 'S:' indicated at the head of each utterance. An empty line indicates a breakpoint of dialogs. The following text is an example of dialog data file.

```
U: hello !
S: how may I help you ?
U: nothing …
S: have a good day !

U: your delivery timing & info leaves a lot to be desired . flowers ordered last wk for delivery yesterday are nowhere to be seen .
S: hello <USER> , i am sorry for the issue you are experiencing with us . please dm me so that i can assist you .
```

2.2 Evaluation data format

Evaluation data basically follows the basic format above, but it also contains partial dialogs that end with a system utterance. With this data, the dialog system has to predict the last utterance in each dialog, where the last utterance is considered the reference. Thus, the system can use the utterances before the last utterance as the context, and predict the last utterance, and is evaluated by comparing the predicted result with the reference.

```
U: hello !                          (context)
S: how may I help you ?             (reference)

U: hello !
S: how may I help you ?             (context)
U: nothing …
S: have a good day !                (reference)
```

2.3 System output format

Dialog systems are expected to read an evaluation data file and output the following file, where the reference is modified to have 'S_REF:' header and a system prediction is appended, which starts with 'S_HYP:'.

```
U: hello !                                  (context)
S_REF: how may I help you ?                 (reference)
S_HYP: hi .                                 (system prediction)

U: hello !
S: how may I help you ?                     (context)
U: nothing …
S_HYP: have a good day !                    (reference)
S_HYP: have a good day !                    (system prediction)
```

Note that the references will NOT be included in the official evaluation data as

```
U: hello !                                  (context)
S:                                          (empty reference)

U: hello !
S: how may I help you ?                     (context)
U: nothing …
S:                                          (empty reference)
```

Therefore, the final result submitted to the challenge does not have to include the reference part starting with "S_REF:".

2.4 Scoring

BLEU score can be computed for the system output if the file includes the references. A scirpt 'tasks/utils/bleu_score.py' are available to compute the score, which will report BLEU scores as the following result.

```
$ utils/bleu_score.py system_output_example.txt
--------------------------------
Evaluated file: system_output_example.txt
Number or references: 500
Number or hypotheses: 500
--------------------------------
Bleu1: 0.233459
Bleu2: 0.112580
Bleu3: 0.068386
Bleu4: 0.043992
--------------------------------
```