Assignment 3: Dialog Parsing for Task-Oriented Dialog Systems

Motivation

A task-oriented dialog (TOD) system assists the user in achieving goals like managing the calendar, sending emails, controlling devices, etc., through natural language. A crucial task in the TOD system is to extract meaning (intent) from the user input and its associated information (slot-values) which is the focus of this assignment. Specifically, you will develop a parsing model which takes input a user utterance, dialog history, and extra context information and predicts the parsed output (see sample below). The downstream applications can then execute the user's intent using the parsed output.

You can read about intents and slot-value

1. input: the user utterances
2. history: a list of past user and system utterances
3. user_lists: List of user's list. A user list is a named collection of items curated by the user. See example below.
4. user_notes: List of user's notes. A user note consists of a title and text. See example below
5. user_contacts: List of user's contacts
6. output: String indicating parsing output
7. pattern: Linguistic pattern in the user utterance

A sample is given below

```
{
  "input": "Message my mom.",
  "history": [
    {
      "usr": "help me create a list named songs",
      "sys": "Sorry, it looks like you already have a list with that name. Do you still want to make a new one?"
    }
  ],
  "user_lists": [
    {
      "name": "mom",
      "items": ["card"]
    }, {
      "name": "Necessities",
      "items": ["trash bags"]
    },...
  ],
  "user_notes": [
    {
      "name": "home repair",
      "content": ""
    }, {
      "name": "party packs",
      "content": ""
    },...
  ],
  "user_contacts": [
    "mom",
    "Mia Farrow",
    "Henrietta",...
  ],
  "output": "Send_digital_object ( medium « Message » recipient Personal_contact ( person « my mom » ) )",
  "pattern": ""
}
```


You must train a pre-trained language model based model to predict the parsed output given other input fields.

Consider the output string "Send_digital_object (medium « Message » recipient Personal_contact (person « my mom »))". Here, Send_digital_object is the intent and (medium, Message) and (recipient Personal_contact, (person, my mom) are slot-value pairs. Note that the data has nested slot-value pairs as well as can be seen for (person, my mom). There are a total 34 intents and 303 slot types in the dataset.

Some user utterances in the data are captured through ASR systems and thus lack fluency (e.g. Send text to umm, to Brian and add ETA.). These samples are annotated with "pattern": "disfluency".

Train and Test Data

All train, evaluation and sample test data are given here. All files use `utf-8` encoding.

1. There are 31k training samples given in the `train.jsonl` file. The format of the data is `jsonl` where each line represents a training sample serialized in json format. You can read about `jsonl` format here. You can use this data whichever way you want to build a

dialog-parsing model. This includes incorporating a combination of different input fields like input, history, user_lists, etc., into your model.
2. There are 9.2k evaluation samples given in the dev.jsonl file. It follows jsonl format same as in training data except “pattern” field (read the following point).
3. Sample test input and output data are given in the sample_test.jsonl and sample_output.txt files. Following a realistic setting where the linguistic pattern for a sample is not known beforehand, the field “pattern” is not available in the test data. These i/o formats are to be followed strictly. No errors in the formatting will be entertained.

The Task

1. You need to develop a parsing model using pre-trained language models (PLMs). Your model should take user utterance as input and predict the parsed output in the given format. You are encouraged to have a glance at the format of the output in the training data. As a starting point, you can consider the task as a sequence generation where you generate the parsed output given the user input. You can then fine-tune PLMs like GPT2 and BART on the task-specific data. You are expected to decide how to represent input and outputs to PLMs during training and inference.
2. You can further incorporate additional features like dialog history, user lists, user notes fields into your model. Feel free to experiment with different combinations and representations for these inputs and select the one that works the best.
3. Given training data consists of two linguistic patterns - None and disfluency. You can exploit this information during training to further improve the model performance. One such approach could be to define an auxiliary task for predicting the pattern during the training (there can be better ways though especially for PLMs).
4. You may incorporate a copy mechanism into your model which allows copying tokens directly from the input. You can find details here.
5. Observe that output follows a certain grammar. You may identify and exploit this grammar for improving your model/constraining your decoder
6. You can create the validation split in any way you want from the given data.

Evaluation Metrics

Your submission will be evaluated using exact-match accuracy where output predicted by your model will be directly matched with gold parsed output (case-sensitive).
Since the parsed output is to be used for downstream applications, it is crucial that the predicted output is grammatically correct. To this end, your predicted output will be checked for correct grammar.
Your submission will also be evaluated just on intent_detection accuracy (without slots).
Further evaluation details will be released on Piazza.

Submission Instructions

1. The submission deadline for the assignment is 29th April at 11:59 PM.

2. The Assignment can be done in groups of one or two.
3. A conda environment for this assignment has been created on HPC. You can activate the environment using command

   conda activate /scratch/cse/phd/csz208845/envs/nlp_a3.

   The environment has pytorch (v1.13) and Huggingface transformers (v4.21). You can assume PLMs GPT2 (gpt2) and BART (bart-base) to be pre-cached in the environment. Thus, you need not download them in your submission. Please raise a request on Piazza before using PLMs apart from these four. Note that your submission will be evaluated using the same environment.
4. Submit your code using a zip file on moodle with name <kerberos_id.zip> (E.g. `csz198394.zip'). Unzipping this should generate a directory with name <kerberos_id> (E.g. `csz198394') having 3 files – `run_model.sh' and `writeup.txt'.
5. The command for training is - ` bash run_model.sh train <train_file_path> <val_file_path>' . This should generate a tagging model named `<kerberos_id>_model' (E.g. `csz198394_model') along with possibly other files needed for inference.
6. Command for inference - `bash run_model.sh test <test_file_path> outputfile.txt' . This should generate `outputfile.txt' having predicted parsed output on each line.
7. You also need to submit your fine-tuned models. Submission instructions for the same will be shared on Piazza.
8. You must not discuss this assignment with anyone outside the class. Make sure you mention the names in your write-up in case you discuss with anyone from within the class. Please read academic integrity guidelines on the course home page and follow them carefully.
9. We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.
10. Your code will be automatically evaluated. You will get a significant penalty (at least 20%) if it does not conform to output guidelines.

Note:- Please see the sample input and output files provided to you. Same format for test input and expected output will be used during evaluation.