UserSimCRS

A User Simulator for Conversational Recommender Systems

System Evaluation using User Simulation

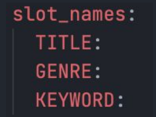
Present the steps involved in evaluating an actual conversational recommender system, referred to as target CRS, using our user simulation toolkit.

Prepare domain and item collection

A yaml file with domain-specific slot names must be prepared for the preference model. Additionally, a file containing the item collection is required.

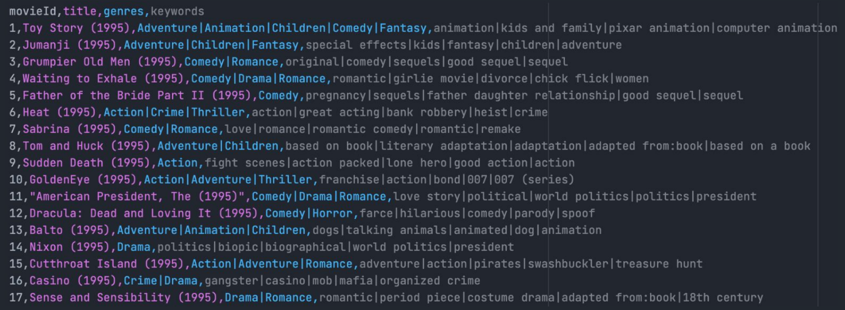
In our example:
The domain is `movies.yaml.` This document contains the slots necessary for the response generation and user modeling, when conversing with the target CRS.

The item collection is the movielens 25M movies dataset. The movies.csv was expanded to contain the relevant domain slots for an item.



```yaml
slot_names:
  TITLE:
  GENRE:
  KEYWORD:
```

Domains/movies.yaml



```
movieId,title,genres,keywords
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy,animation|kids and family|pixar animation|computer animation
2,Jumanji (1995),Adventure|Children|Fantasy,special effects|kids|fantasy|children|adventure
3,Grumpier Old Men (1995),Comedy|Romance,original|comedy|sequels|good sequel|sequel
4,Waiting to Exhale (1995),Comedy|Drama|Romance,romantic|girlie movie|divorce|chick flick|women
5,Father of the Bride Part II (1995),Comedy,pregnancy|sequels|father daughter relationship|good sequel|sequel
6,Heat (1995),Action|Crime|Thriller,action|great acting|bank robbery|heist|crime
7,Sabrina (1995),Comedy|Romance,love|romance|romantic comedy|romantic|remake
8,Tom and Huck (1995),Adventure|Children,based on book|literary adaptation|adaptation|adapted from:book|based on a book
9,Sudden Death (1995),Action,fight scenes|action packed|lone hero|good action|action
10,GoldenEye (1995),Action|Adventure|Thriller,franchise|action|bond|007|007 (series)
11,"American President, The (1995)",Comedy|Drama|Romance,love story|political|world politics|politics|president
12,Dracula: Dead and Loving It (1995),Comedy|Horror,farce|hilarious|comedy|parody|spoof
13,Balto (1995),Adventure|Animation|Children,dogs|talking animals|animated|dog|animation
14,Nixon (1995),Drama,politics|biopic|biographical|world politics|president
15,Cutthroat Island (1995),Action|Adventure|Romance,adventure|action|pirates|swashbuckler|treasure hunt
16,Casino (1995),Crime|Drama,gangster|casino|mob|mafia|organized crime
17,Sense and Sensibility (1995),Drama|Romance,romantic|period piece|costume drama|adapted from:book|18th century
```

Itemcollection/movies.csv

Provide preference data

Preference data should be provided in a CSV file with mappings between user, item, and the corresponding preference as a number (e.g., rating between 1-5).

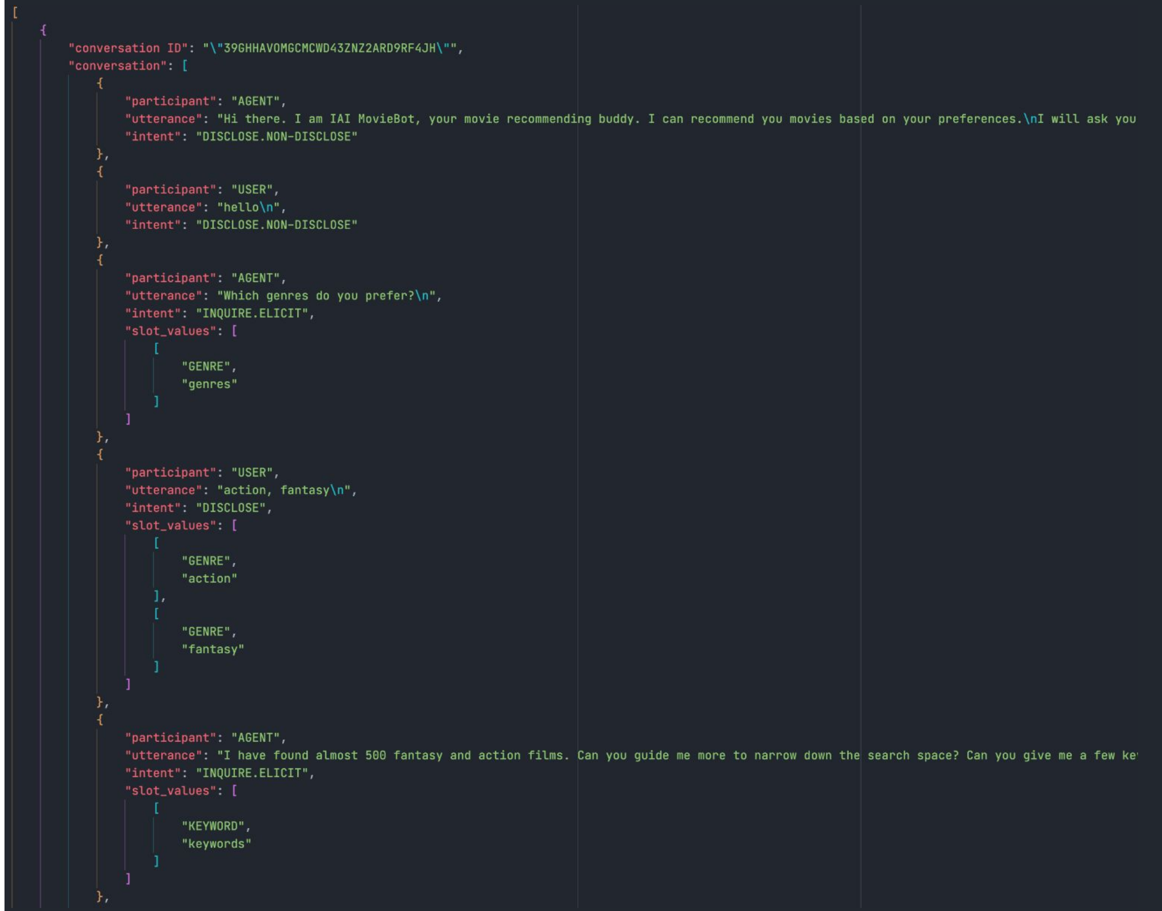
In our example:
The preference data is the movielens 25M ratings.

```
userId,movieId,rating
1,296,5.0
1,306,3.5
1,307,5.0
1,665,5.0
1,899,3.5
1,1088,4.0
1,1175,3.5
1,1217,3.5
1,1237,5.0
1,1250,4.0
1,1260,3.5
1,1653,4.0
1,2011,2.5
1,2012,2.5
1,2068,2.5
1,2161,3.5
1,2351,4.5
1,2573,4.0
1,2632,5.0
1,2692,5.0
1,2843,4.5
1,3448,4.0
1,3569,5.0
```

Itemcollection/ratings.csv

Dialogue sample

A small sample of dialogues with the target CRS need to be collected. The sample size depends on the complexity of the system, in terms of action space and language variety. Additionally, this sample of dialogues must contain annotations for both intents and entities, as this is required to train the NLU and NLG.

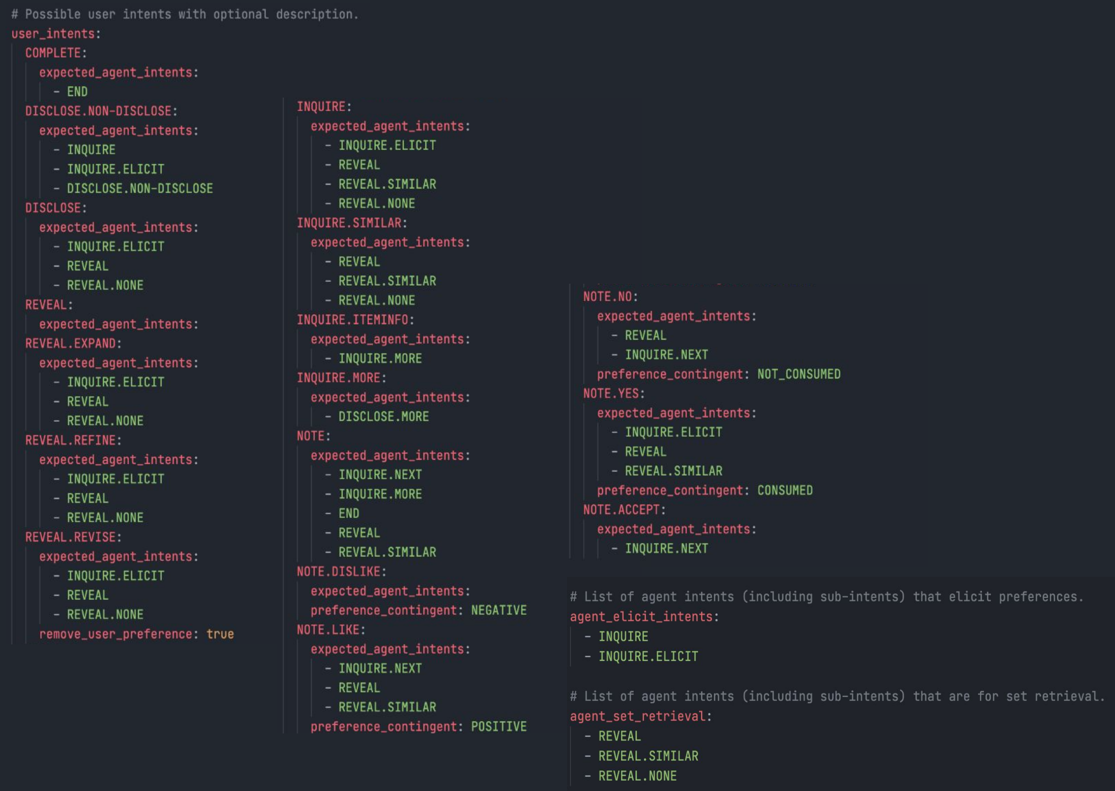


```
[
    {
        "conversation ID": "\"39GHHAVOMGCMCWD43ZNZ2ARD9RF4JH\"",
        "conversation": [
            {
                "participant": "AGENT",
                "utterance": "Hi there. I am IAI MovieBot, your movie recommending buddy. I can recommend you movies based on your preferences.\nI will ask you
                "intent": "DISCLOSE.NON-DISCLOSE"
            },
            {
                "participant": "USER",
                "utterance": "hello\n",
                "intent": "DISCLOSE.NON-DISCLOSE"
            },
            {
                "participant": "AGENT",
                "utterance": "Which genres do you prefer?\n",
                "intent": "INQUIRE.ELICIT",
                "slot_values": [
                    [
                        "GENRE",
                        "genres"
                    ]
                ]
            },
            {
                "participant": "USER",
                "utterance": "action, fantasy\n",
                "intent": "DISCLOSE",
                "slot_values": [
                    [
                        "GENRE",
                        "action"
                    ],
                    [
                        "GENRE",
                        "fantasy"
                    ]
                ]
            },
            {
                "participant": "AGENT",
                "utterance": "I have found almost 500 fantasy and action films. Can you guide me more to narrow down the search space? Can you give me a few ke
                "intent": "INQUIRE.ELICIT",
                "slot_values": [
                    [
                        "KEYWORD",
                        "keywords"
                    ]
                ]
            },
```

DialogueSamples/target_crs_sample.json

Define interaction model

A file containing the intent space and the interaction between these is required for the interaction model. Furthermore, dependencies between intents and the preference model should be explicitly indicated.



```yaml
# Possible user intents with optional description.
user_intents:
  COMPLETE:
    expected_agent_intents:
      - END
  DISCLOSE.NON-DISCLOSE:
    expected_agent_intents:
      - INQUIRE
      - INQUIRE.ELICIT
      - DISCLOSE.NON-DISCLOSE
  DISCLOSE:
    expected_agent_intents:
      - INQUIRE.ELICIT
      - REVEAL
      - REVEAL.NONE
  REVEAL:
    expected_agent_intents:
  REVEAL.EXPAND:
    expected_agent_intents:
      - INQUIRE.ELICIT
      - REVEAL
      - REVEAL.NONE
  REVEAL.REFINE:
    expected_agent_intents:
      - INQUIRE.ELICIT
      - REVEAL
      - REVEAL.NONE
  REVEAL.REVISE:
    expected_agent_intents:
      - INQUIRE.ELICIT
      - REVEAL
      - REVEAL.NONE
    remove_user_preference: true
```

```yaml
  INQUIRE:
    expected_agent_intents:
      - INQUIRE.ELICIT
      - REVEAL
      - REVEAL.SIMILAR
      - REVEAL.NONE
  INQUIRE.SIMILAR:
    expected_agent_intents:
      - REVEAL
      - REVEAL.SIMILAR
      - REVEAL.NONE
  INQUIRE.ITEMINFO:
    expected_agent_intents:
      - INQUIRE.MORE
  INQUIRE.MORE:
    expected_agent_intents:
      - DISCLOSE.MORE
  NOTE:
    expected_agent_intents:
      - INQUIRE.NEXT
      - INQUIRE.MORE
      - END
      - REVEAL
      - REVEAL.SIMILAR
  NOTE.DISLIKE:
    expected_agent_intents:
    preference_contingent: NEGATIVE
  NOTE.LIKE:
    expected_agent_intents:
      - INQUIRE.NEXT
      - REVEAL
      - REVEAL.SIMILAR
    preference_contingent: POSITIVE
```

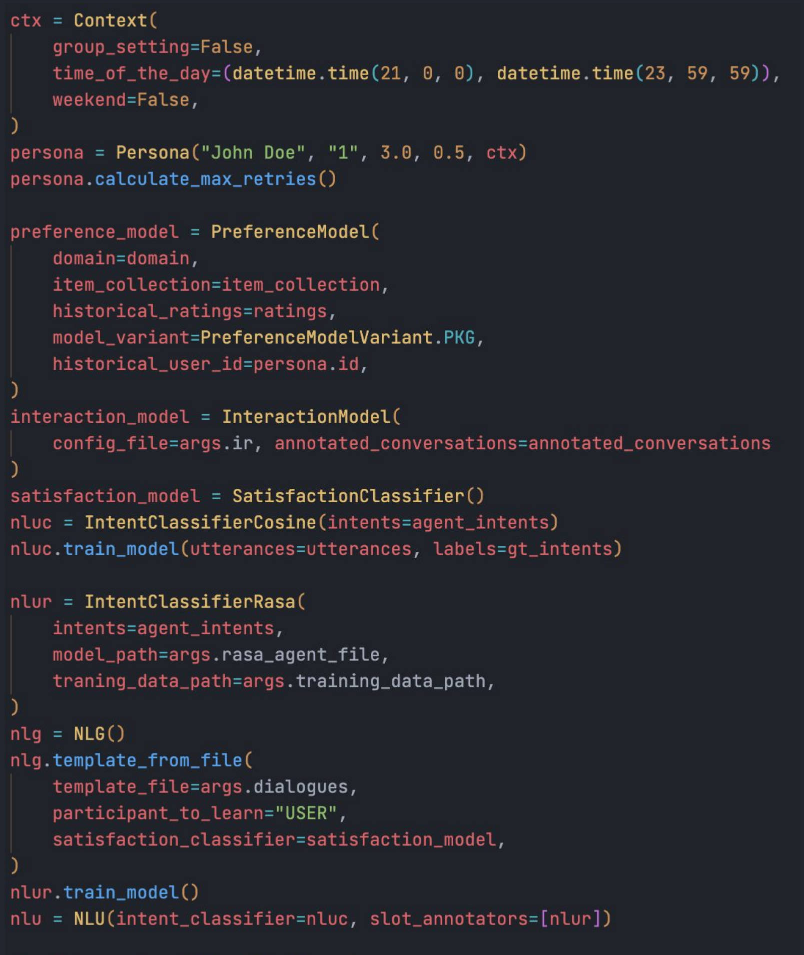
```yaml
  NOTE.NO:
    expected_agent_intents:
      - REVEAL
      - INQUIRE.NEXT
    preference_contingent: NOT_CONSUMED
  NOTE.YES:
    expected_agent_intents:
      - INQUIRE.ELICIT
      - REVEAL
      - REVEAL.SIMILAR
    preference_contingent: CONSUMED
  NOTE.ACCEPT:
    expected_agent_intents:
      - INQUIRE.NEXT

# List of agent intents (including sub-intents) that elicit preferences.
agent_elicit_intents:
  - INQUIRE
  - INQUIRE.ELICIT

# List of agent intents (including sub-intents) that are for set retrieval.
agent_set_retrieval:
  - REVEAL
  - REVEAL.SIMILAR
  - REVEAL.NONE
```

interaction_models/CRSv1.yaml

Define user model/population

Train simulator

The NLU and NLG components of the simulator are trained using the dialogue sample. The path to the required files can be defined in a config file, and optionally overridden via the command line

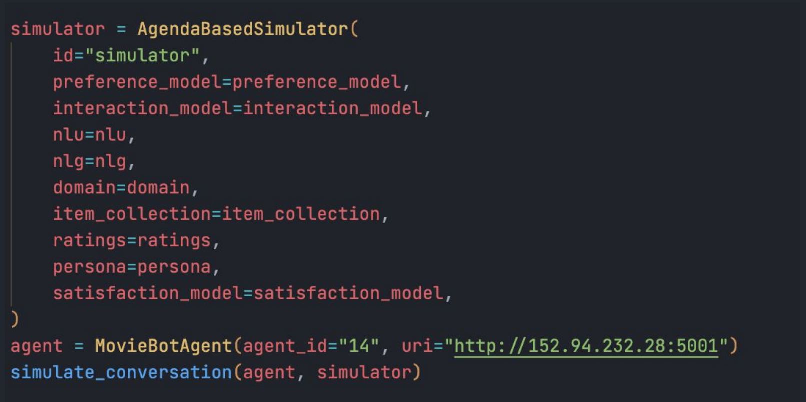


```python
ctx = Context(
    group_setting=False,
    time_of_the_day=(datetime.time(21, 0, 0), datetime.time(23, 59, 59)),
    weekend=False,
)
persona = Persona("John Doe", "1", 3.0, 0.5, ctx)
persona.calculate_max_retries()

preference_model = PreferenceModel(
    domain=domain,
    item_collection=item_collection,
    historical_ratings=ratings,
    model_variant=PreferenceModelVariant.PKG,
    historical_user_id=persona.id,
)
interaction_model = InteractionModel(
    config_file=args.ir, annotated_conversations=annotated_conversations
)
satisfaction_model = SatisfactionClassifier()
nluc = IntentClassifierCosine(intents=agent_intents)
nluc.train_model(utterances=utterances, labels=gt_intents)

nlur = IntentClassifierRasa(
    intents=agent_intents,
    model_path=args.rasa_agent_file,
    traning_data_path=args.training_data_path,
)
nlg = NLG()
nlg.template_from_file(
    template_file=args.dialogues,
    participant_to_learn="USER",
    satisfaction_classifier=satisfaction_model,
)
nlur.train_model()
nlu = NLU(intent_classifier=nluc, slot_annotators=[nlur])
```

run_simulation.py

Run simulation

The agenda-based simulator is configured with the previously defined parameters and models, before starting the simulation.

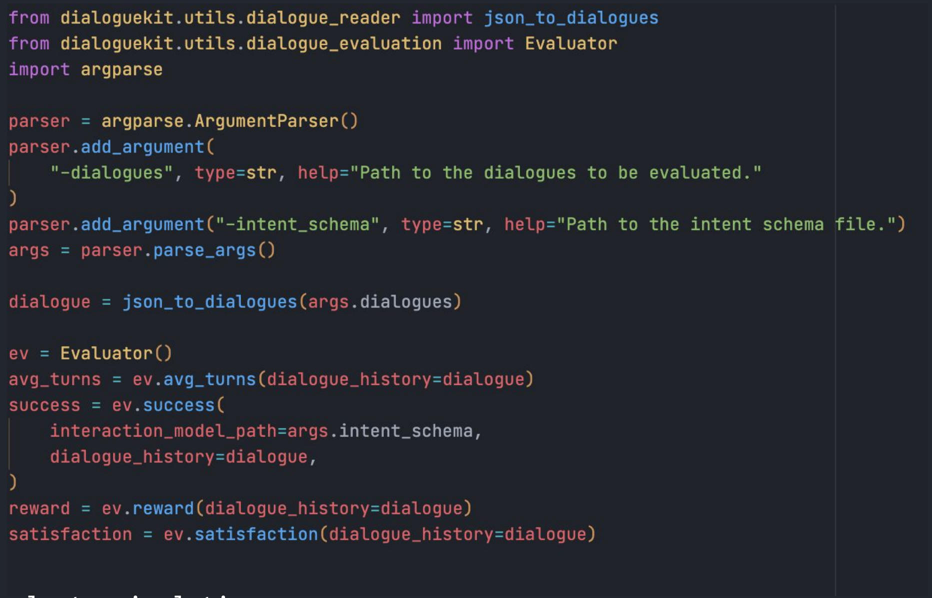


```python
simulator = AgendaBasedSimulator(
    id="simulator",
    preference_model=preference_model,
    interaction_model=interaction_model,
    nlu=nlu,
    nlg=nlg,
    domain=domain,
    item_collection=item_collection,
    ratings=ratings,
    persona=persona,
    satisfaction_model=satisfaction_model,
)
agent = MovieBotAgent(agent_id="14", uri="http://152.94.232.28:5001")
simulate_conversation(agent, simulator)
```

run_simulation.py

Perform evaluation

Evaluation is performed with regards to the metrics implemented in **DialogueKit**, i.e., **AvgTurns**, **AvgReward**, **AvgSatisfaction**, and **AvgSuccess**.



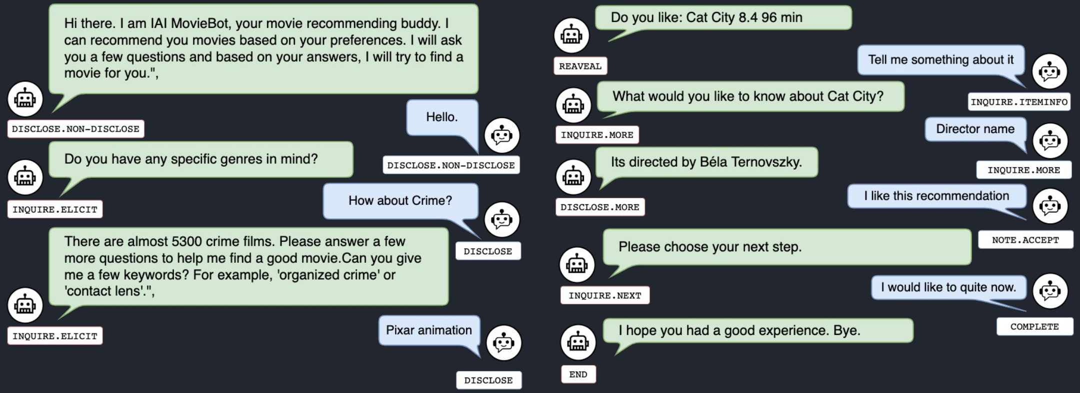
```python
from dialoguekit.utils.dialogue_reader import json_to_dialogues
from dialoguekit.utils.dialogue_evaluation import Evaluator
import argparse

parser = argparse.ArgumentParser()
parser.add_argument(
    "-dialogues", type=str, help="Path to the dialogues to be evaluated."
)
parser.add_argument("-intent_schema", type=str, help="Path to the intent schema file.")
args = parser.parse_args()

dialogue = json_to_dialogues(args.dialogues)

ev = Evaluator()
avg_turns = ev.avg_turns(dialogue_history=dialogue)
success = ev.success(
    interaction_model_path=args.intent_schema,
    dialogue_history=dialogue,
)
reward = ev.reward(dialogue_history=dialogue)
satisfaction = ev.satisfaction(dialogue_history=dialogue)
```

evaluate_simulation.py

Example simulated dialogue

In this example, the target CRS is IAI MovieBot.

TEST



Hi there. I am IAI MovieBot, your movie recommending buddy. I can recommend you movies based on your preferences. I will ask you a few questions and based on your answers, I will try to find a movie for you.",
DISCLOSE.NON-DISCLOSE

Hello.
DISCLOSE.NON-DISCLOSE

Do you have any specific genres in mind?
INQUIRE.ELICIT

How about Crime?
DISCLOSE

There are almost 5300 crime films. Please answer a few more questions to help me find a good movie.Can you give me a few keywords? For example, 'organized crime' or 'contact lens'.",
INQUIRE.ELICIT

Pixar animation
DISCLOSE

Do you like: Cat City 8.4 96 min
REAVEAL

Tell me something about it
INQUIRE.ITEMINFO

What would you like to know about Cat City?
INQUIRE.MORE

Director name
INQUIRE.MORE

Its directed by Béla Ternovszky.
DISCLOSE.MORE

I like this recommendation
NOTE.ACCEPT

Please choose your next step.
INQUIRE.NEXT

I would like to quite now.
COMPLETE

I hope you had a good experience. Bye.
END