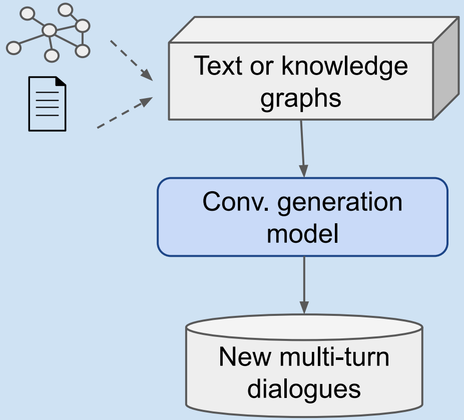
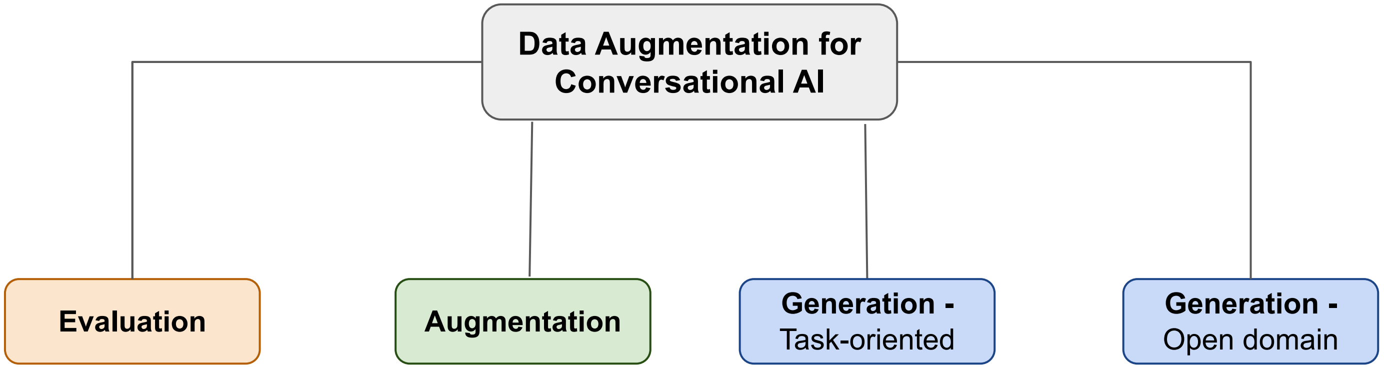
Part 3: Conversation Generation - Task Oriented

Duration: 40 min

Presenter: Evangelos Kanoulas (online) & Roxana Petcu



Overview


Data Augmentation for Conversational AI
Evaluation
Augmentation
Generation - Task-oriented
Generation - Open domain

Task-oriented **Conversation Dataset Augmentation/Generation**

Definition

Task-oriented dialogue (TOD) systems assist users in completing tasks, e.g., booking a restaurant or making an appointment.

Challenges

- Task-specific structural constraints
- As a consequence, it requires either a large corpus of annotated dialogue, or a schema
- Difficult to extend to new domains

Example of task-oriented dialogue data

User: find a restaurant in orlando.

System: what type of food and price range should i look for?

User: i’d like moderately priced taiwanese

Example of task-oriented dialogue data

User: find a restaurant in orlando.

System: what type of food and price range should i look for?

User: i'd like moderately priced taiwanese


```
"dialogue_state": [
        {
          "slot": "location", "value": "orlando"
        }],
"user_acts": [],
"user_intents": ["FIND_RESTAURANT"],
"user_utterance": {
        "slots": [
          {
            "exclusive_end": 5,
            "slot": "location",
            "start": 4
          }
        ]
```

Example of task-oriented dialogue data

User: find a restaurant in orlando.

System: what type of food and price range should i look for?

User: i'd like moderately priced taiwanese


```
"dialogue_state": [
  {"slot": "location",          "value": "orlando"},
  {"slot": "price_range",       "value": "moderately priced"},
  {"slot": "category",          "value": "taiwanese"}]

"system_acts": [
  { "slot": "price_range",     "type": "REQUEST" },
  { "slot": "category",        "type": "REQUEST" }],

"user_acts": [
        { "type": "INFORM" } ],
"user_utterance": {
        "slots": [
          { "exclusive_end": 6,
            "slot": "price_range",
            "start": 4},
          { "exclusive_end": 7,
            "slot": "category",
            "start": 6}],
        }
```

Background & challenges

- Task-oriented dialogue systems work best when trained on dialogues of the same task; for new tasks datasets of ***human-agent dialogues*** typically do not exists
- Solution A: Collect and annotate free form dialogues through ***crowdsourcing*** using a Wizard-of-Oz setup
    - ***Expensive***
    - May not ***cover*** all possible interactions
    - Unfit dialogues (e.g. strange language)
    - Errors in ***dialogue act annotations*** (e.g. MultiWOZ has still significant inconsistencies)

Datasets

- MultiWOZ (Budzianowski et al. 2018;)
  - Human-to-human dialogues; 7 domains related to travel and 10,000 dialogues, with corresponding goal instruction and KBs
- MultiWOZ 2.1 (Eric et al. 2019)
  - Human-to-human dialogues; 10,000 dialogues about one or more of 7 domains
- MultiWOZ 2.3 (Han et al. 2021)
  - Corrects the annotations of previous MultiWOZ versions
- WOZ 2.0 (Wen et al. 2017)
  - Human-to-human dialogues; single domain; 235 dialogues

Background & challenges

- Task-oriented dialogue systems work best when trained on dialogues of the same task; for new tasks datasets of ***human-agent dialogues*** typically do not exists
- Solution B: Develop dialogue experiences/skills (e.g. through wit.ai)
    - Engineer ***every aspect*** of the conversational interaction and anticipate ***all ways*** user might interact

Data Generation

Overall Goals:

(1) *Reduce the cos*t and effort required to build dialogue datasets by
  - automating the task-independent steps;
  - leaving the task-specific aspects to the developer.

(2) Improve the *quality* of dialogues by improving
  - the ***diversity*** of language and dialogue flow;
  - the ***coverage*** of all expected user behaviour;
  - the ***correctness*** of labels.

Synthetic Conversation Evaluation

Intrinsic Evaluation

Evaluate directly the quality of generated dialogue

- Automatic evaluation
- Human annotation

Extrinsic Evaluation

Train the dialogue model with synthetically generated data and evaluate the performance on downstream taks

Synthetic Conversation Evaluation

Dialogue diversity

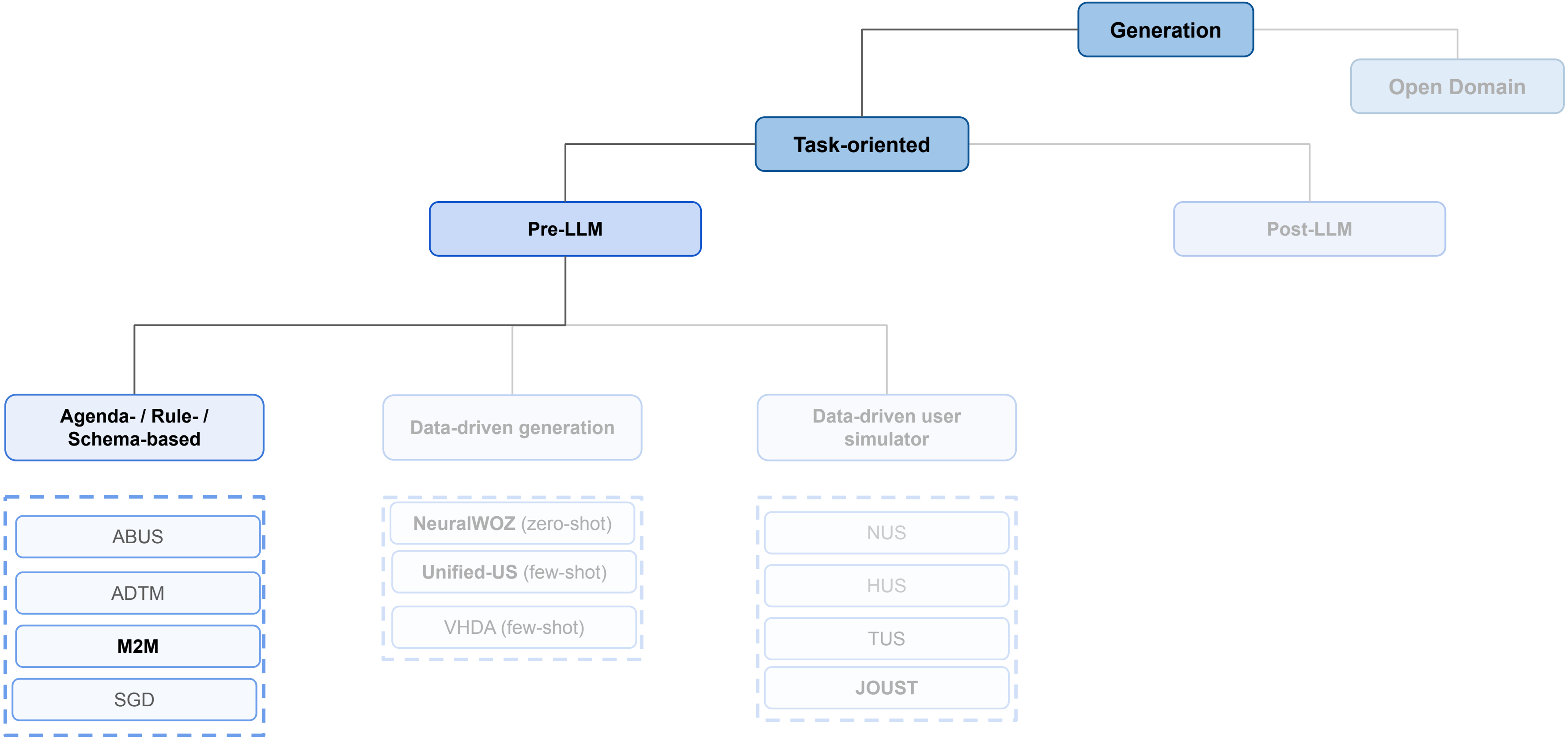
- Flow diversity
    - unique transitions at the semantic frame level (dialogue act, slots, values)

Few-shot abilities

- Compute the log probability of generated dialogues on the target domain using an LM trained on human-generated data in the target domain

- Train dialogue state tracking models in a zero-shot/few-shot scenario (withhold one domain from training) and measure *slot accuracy*

Overview



Generation
Open Domain
Task-oriented
Pre-LLM
Post-LLM
Agenda- / Rule- / Schema-based
Data-driven generation
Data-driven user simulator
ABUS
ADTM
M2M
SGD
NeuralWOZ (zero-shot)
Unified-US (few-shot)
VHDA (few-shot)
NUS
HUS
TUS
JOUST

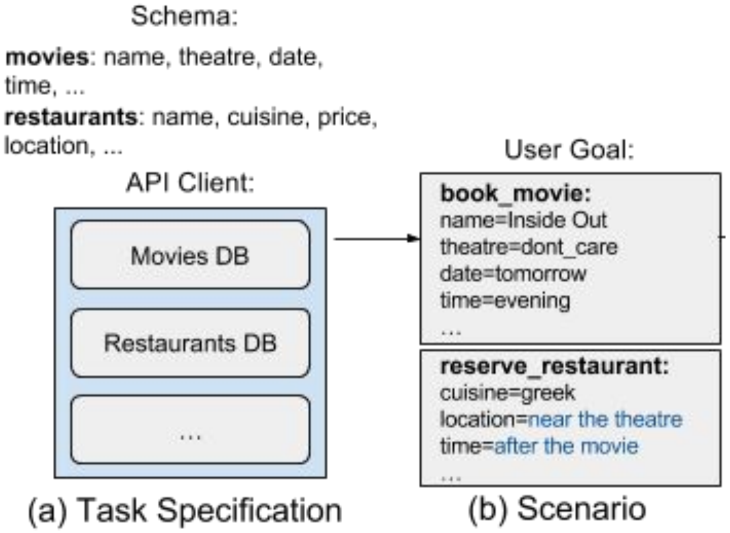
Rule-based Generation: **M2M** **(Shah et al.,2018)**

<u>Goal</u>: Automatically generate dialogues and annotations

Task: *Finding an entity* (e.g. a book, a movie, etc.)

Schema: Entity attributes (e.g. the columns of a database that stores all entities)

API client returns entities using valid combination of attributes



Rule-based Generation: **M2M**

Given a schema generate *a user goal* by randomly choosing values for all slots in the schema

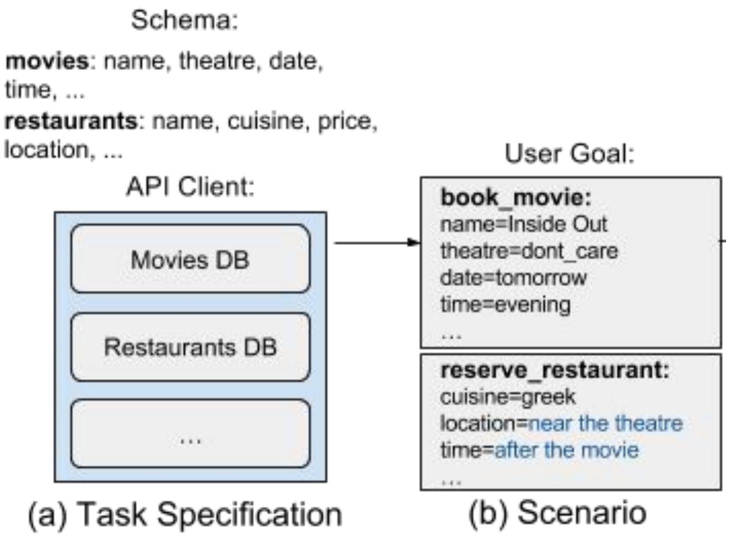


Table 4: List of dialogue acts.

Dialogue Act	Speaker	Description
GREETING	User/System	Greet the other speaker
INFORM	User/System	Inform a slot value
CONFIRM	User/System	Ask the other speaker to confirm a given slot value
REQUEST	User/System	Ask for the value of a slot
REQUEST_ALTS	User	Ask for more alternatives
OFFER	System	Offer a database entity to the user
SELECT	System	Offer more than one database entity to the user
AFFIRM	User/System	Agree to something said by the other speaker
NEGATE	User/System	Disagree to something said by the other speaker
NOTIFY_SUCCESS	System	Notify the user of a successful event, e.g. a booking is complete
NOTIFY_FAILURE	System	Notify the user of a failure event, e.g. a booking isn't available
THANK_YOU	User/System	Thank the other speaker
GOOD_BYE	User/System	Say goodbye to the other speaker
CANT_UNDERSTAND	User/System	Tell the other speaker that their utterance was not understood
OTHER	User	Unknown utterance type

Rule-based Generation: **M2M**

A framework that combines automation and crowdsourcing

- Machine-to-Machine: Given a user profile *p*, a user goal *g*, a task Schema *S and* an API client *C* generate dialogue outlines
  - User bot vs. System bot
  - User bot:
    - Agenda-based user simulator
    - $B_U = P(\alpha_j^i | \alpha_1^i, ..., \alpha_{j-1}^i, p_i, g_i)$
  - System bot:
    - Finite state machine that encodes **a manually defined set of rules** that follows a predetermined sequence of sub-dialogues (i.e. dialogue act transitions)
    - $B_S = P(\alpha_{j+1}^i | \alpha_1^i, ..., \alpha_j^i, S, C)$

Rule-based Generation: **M2M**

A framework that combines automation and crowdsourcing

- Machine-to-Machine: Given a user profile *p*, a user goal *g*, a task Schema *S and* an API client *C* generate dialogue outlines
    - User bot vs. System bot

Annotation (a_i)
S: greeting()
U: inform(intent=book_movie, name=Inside Out, date=tomorrow, num_tickets=2)
S: ack() request(time)
U: inform(time=evening)
S: offer(theatre=Cinemark 16, time=6pm)
U: affirm()
S: notify_success()
U: inform(intent=find_restaurant, meal=dinner, location=near the theatre)
S: request(cuisine, price_range)
U: inform(cuisine=DontCare, price_range=moderate, rating=high)
S: select(restaurant={First Wok, Lucy's Grill}, location=near the theatre)
U: inform(intent=reserve_restaurant, restaurant=First Wok, time=after the movie)
S: ack() confirm(restaurant=First Wok, time=8pm, num_people=2)
U: affirm()
S: notify_success()
U: thank_you() good_bye()

Rule-based Generation: **M2M**

A framework that combines automation and crowdsourcing

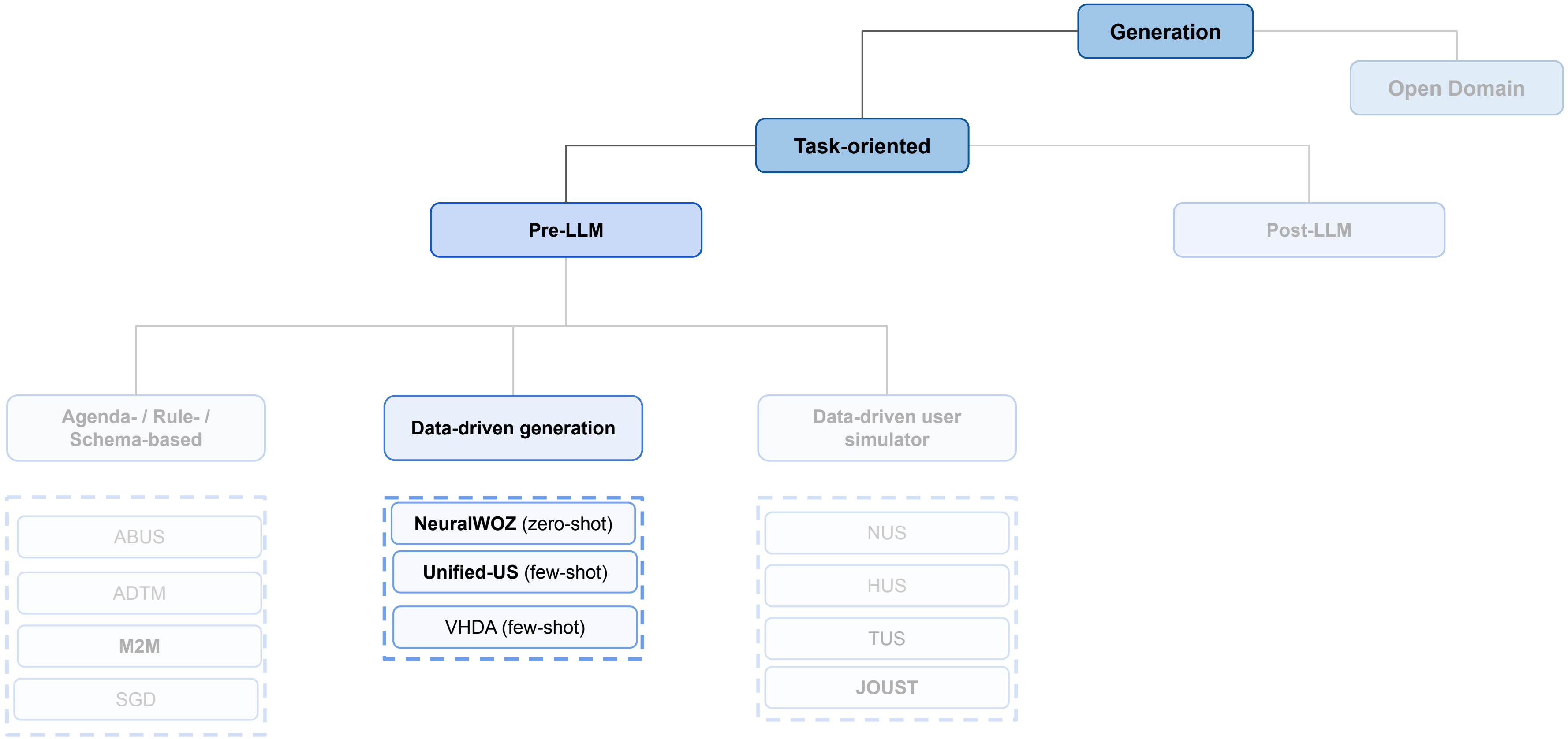
- Machine-to-Machine: Given a user profile *p*, a user goal *g*, a task Schema *S and* an API client *C* generate dialogue outlines
  - User bot vs. System bot
  - Template-based natural language + paraphrasing through crowdsourcing

Annotation (a_i)	**Template utterances** (t_i)
S: greeting()	Greeting.
U: inform(intent=book_movie, name=Inside Out, date=tomorrow, num_tickets=2)	Book movie with name is Inside Out and date is tomorrow and num tickets is 2.
S: ack() request(time)	OK. Provide time.
U: inform(time=evening)	Time is evening.
S: offer(theatre=Cinemark 16, time=6pm)	Offer theatre is Cinemark 16 and time is 6pm.
U: affirm()	Agree.
S: notify_success()	Reservation confirmed.
U: inform(intent=find_restaurant, meal=dinner, location=near the theatre)	Find restaurant with meal is dinner and location is near the theatre.
S: request(cuisine, price_range)	Provide cuisine and price range.
U: inform(cuisine=DontCare, price_range=moderate, rating=high)	Cuisine is I don't care and price range is moderate and rating is high.
S: select(restaurant={First Wok, Lucy's Grill}, location=near the theatre)	Select restaurant from First Wok, Lucy's Grill with location is near the theatre.
U: inform(intent=reserve_restaurant, restaurant=First Wok, time=after the movie)	Reserve restaurant with restaurant is First Wok and time is after the movie.
S: ack() confirm(restaurant=First Wok, time=8pm, num_people=2)	OK. Confirm restaurant is First Wok and time is 8pm and num people is 2.
U: affirm()	Agree.
S: notify_success()	Reservation confirmed.
U: thank_you() good_bye()	Thank you and good bye.

Rule-based Generation: Drawbacks

- Developers define many ingredients of the simulations
  - developers define domain schema, rules, and dialogue templates to simulate user behavior under certain goals, and
  - dialogues are realized by predefined mapping rules or paraphrasing by crowdworkers.
- Requires expert knowledge
- Rules, templates, schemas become intractable for complex domains
- Hard to transfer knowledge across domains

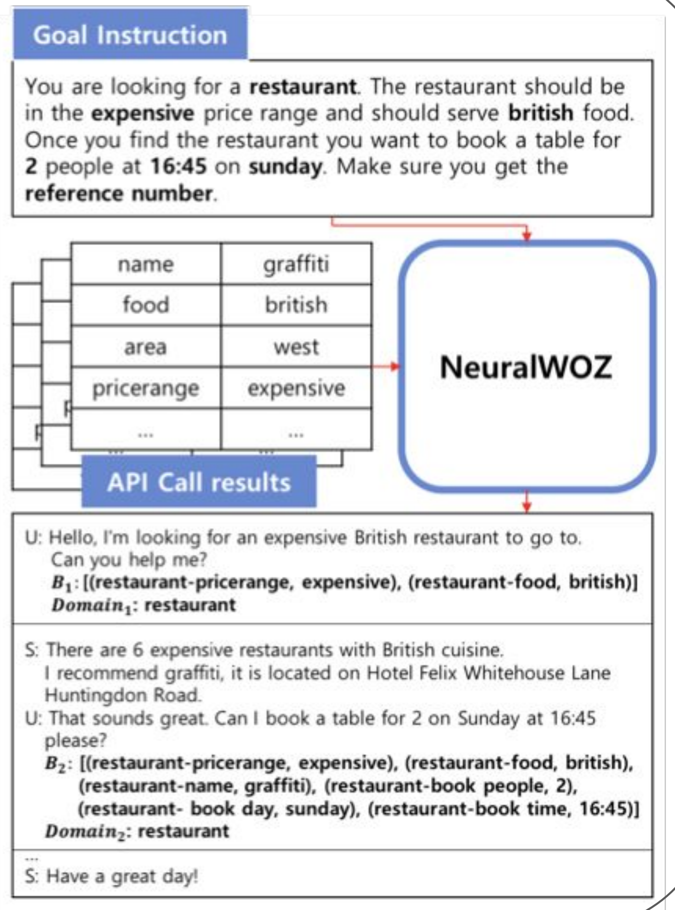
Overview



Generation
Open Domain
Task-oriented
Pre-LLM
Post-LLM
Agenda- / Rule- / Schema-based
Data-driven generation
Data-driven user simulator
ABUS
ADTM
M2M
SGD
NeuralWOZ (zero-shot)
Unified-US (few-shot)
VHDA (few-shot)
NUS
HUS
TUS
JOUST

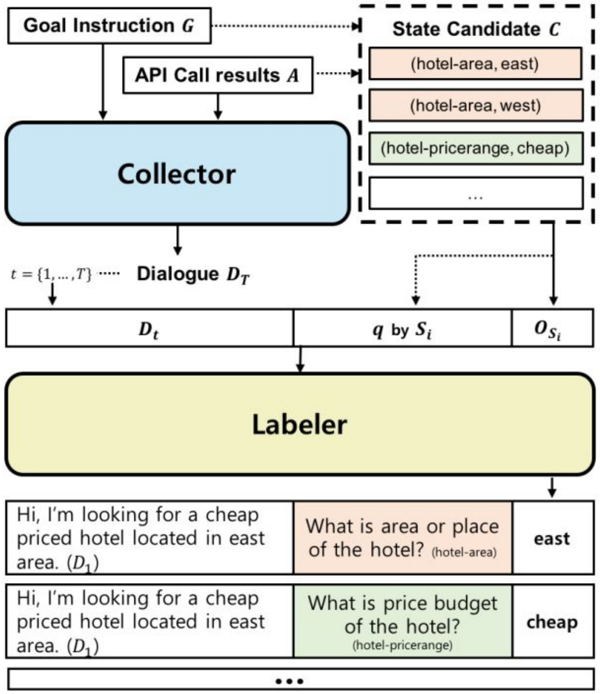
Data-driven Generation: NeuralWOZ

- Define a set of *goal templates*
  - instructions w/ slots
- A goal template is then sampled and filled in with values from a KB and a goal instruction is produced.
- The goal instruction is a natural language text describing constraints of user behavior in the dialogue including informable and requestable slots
- The API call results are corresponding *query results*



Data-driven Generation: NeuralWOZ

- Define state candidate C as all informable slots of API results not specified in instructions
- The Collector is a sequence-to-sequence model
  - Inputs: user goal and state candidate: <s> ⊕ G ⊕ </s> ⊕ <domain>⊕domain_a⊕<slot>⊕S^{ai}_1⊕V^{ai}_1 ⊕ ⋯
  - Model: BART trained on MultiWOZ
  - Output: Dialogue
- Labeler
  - Input: the dialogue, a question (description of corresponding slot) and the set of answer options
  - Model: Roberta trained on MultiWOZ



Data-driven Generation: **Unified-US**

Motivation

- Despite NeuralWOZ being zero-shot in terms of data it can mainly be effective to domains with similar schemas

Data-driven Generation: **Unified-US**

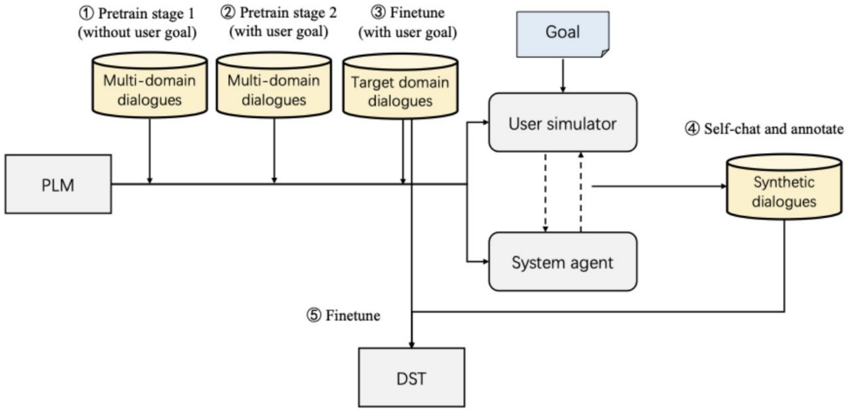
- End-to-end user simulator
  - Context-to-response model
  - Input: task description, user goal and dialogue context
  - Trained on public dialogue datasets
- End-to-end system response
  - Input: task description, dialogue context
- No access to database (entities or schema)
  - Mark values of potential slots with special tokens in training
- Mark slot values with special token (| |)

Goal: You want to book a taxi. The taxi should go to anatolia and should leave after 22:00. The taxi should depart from kymmoy. Make sure you get car type and contact number.

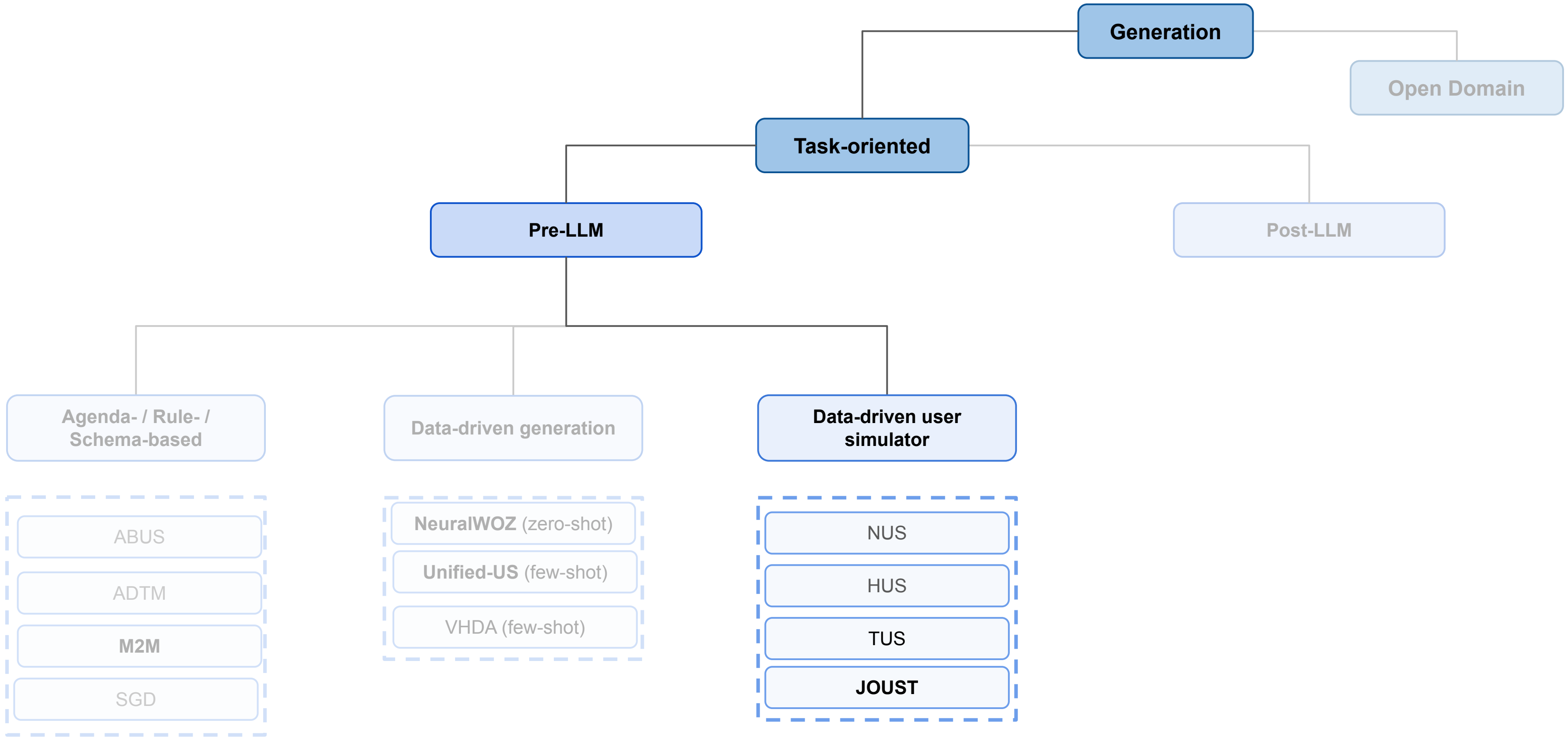
User: i need a taxi to |anatolia|>.
System: what time would you like to leave?
User: i would like to leave after |22:00|>.
System: there are |9|> options available to you. where would you like to leave from?
User: i 'd like to leave from |kymmoy|>.
System: the fare estimate is |9.78|>. do you want to book this taxi now?
User: no, i would like the contact number if possible.
System: the contact number is |07356725299|>
User: thank you very much for your help.
System: you are welcome. do you need anything else today

Data-driven Generation: **Unified-US**

- Pretrained-Language Model: T5
- Pretrain with and without user goal
  - When user goal annotations are not available, dialog acts are used to deduce goals: through manually designed templates + paraphrasing
- Finetune with target domain; 5%-10% of dataset to simulate low-resource setting
- Self-chat between user and system agents
- Annotation: extract special tokens from simulator generation and match



Overview



Generation
Open Domain
Task-oriented
Pre-LLM
Post-LLM
Agenda- / Rule- / Schema-based
Data-driven generation
Data-driven user simulator
ABUS
ADTM
M2M
SGD
NeuralWOZ (zero-shot)
Unified-US (few-shot)
VHDA (few-shot)
NUS
HUS
TUS
JOUST

Data-driven User Simulator: JOUST (Tseng et al. 2021)

- Trains a **user simulator** and a dialogue policy through **reinforcement learning** methods for task-oriented dialogues
- Supervised learning for user simulator and dialogue policy
- Reinforcement learning allows the user simulator (and the dialogue policy) to depart from known strategies learnt from fixed limited corpus

- Similar approaches: Liu and Lane, 2017 (shared reward); Papangelis et al., 2019 (... but for single domain dialogues); Takanobu et al., 2020 (role-aware rewards ... but at semantic level)

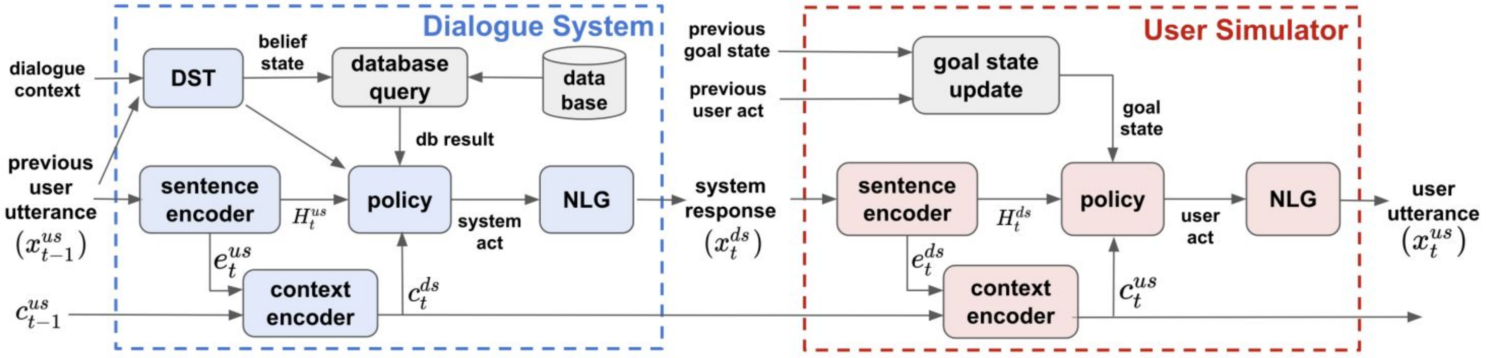
Data-driven User Simulator: JOUST

The DS:

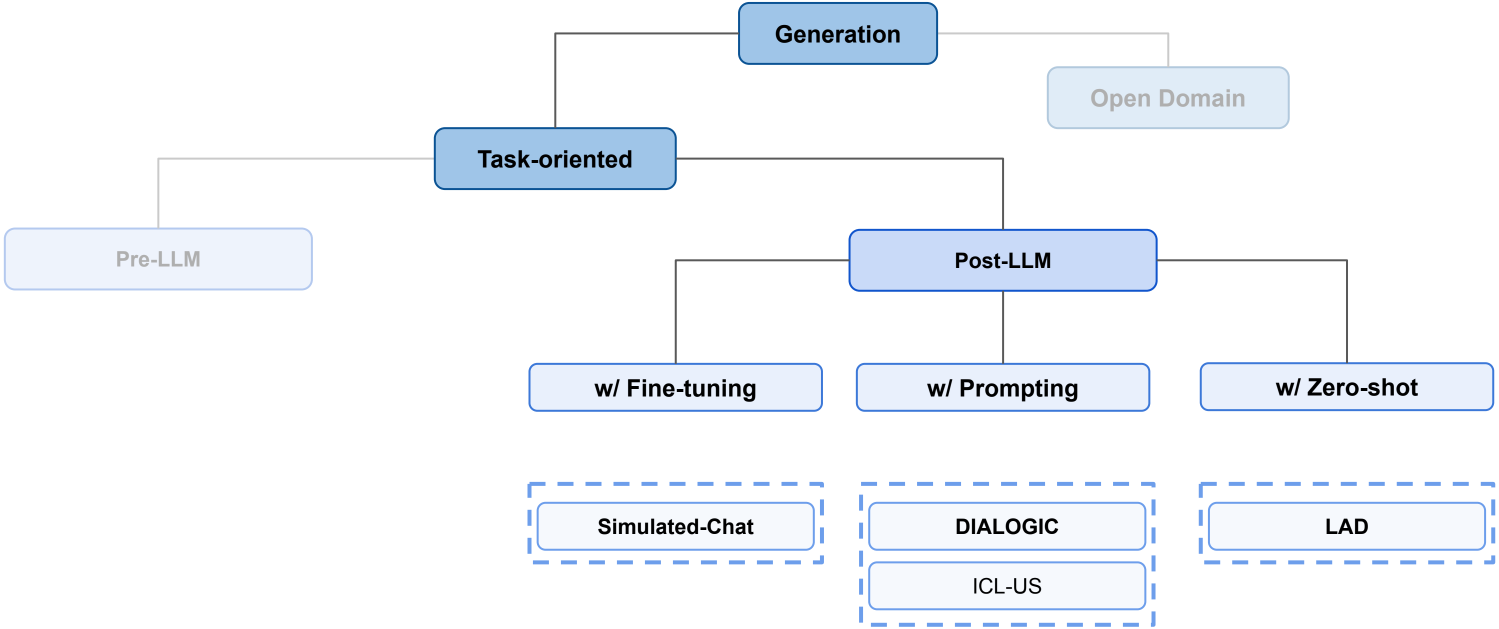
- Processes the dialogue history and generates a belief state
- Solves a dialogue state tracking task, e.g. {hotel_area=north}.

The US:

- Tracks a goal state, update at each dialogue turn



Overview

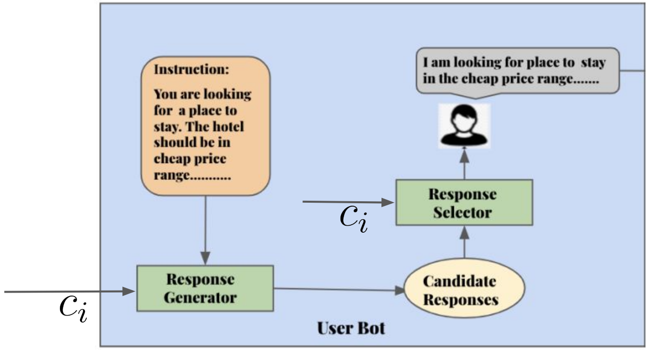

Generation
Open Domain
Task-oriented
Pre-LLM
Post-LLM
w/ Fine-tuning
w/ Prompting
w/ Zero-shot
Simulated-Chat
DIALOGIC
ICL-US
LAD

Post-LLM w/ Fine-tuning: **SimulatedChat**

- Fine-tuning pre-trained LLMs: **GPT-2**, **Longformer**
- Simulation framework: user simulator and agent simulator
    - User **input**: instructions $\mathcal{I}$
    - Agent **input**: knowledge base $\mathcal{KB}$
- Generative Data Augmentation

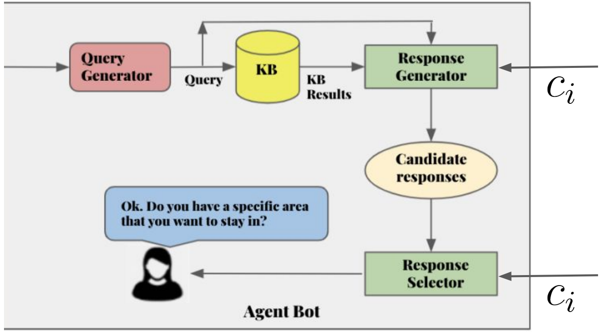
Post-LLM w/ Fine-tuning: **SimulatedChat**

- User simulator
  - **Inputs**: dialog history c_i and instructions $\mathcal{I}$
  - **Response Generator** (GPT-2):
    - Autoregressively generates a pool of candidate utterances
  - **Response Selector** (Longformer):
    - Assigns a context score for each candidate and returns a user utterance
    - Why Longformer? It can handle longer contexts (10 negative for each positive sample)
    - Each response is concatenated to the dialog history and fed to the Longformer
  - **Output**: User utterance



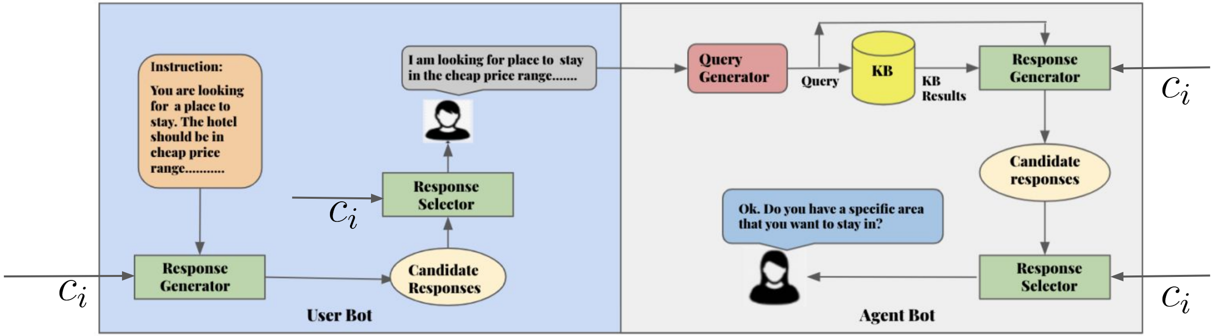
Post-LLM w/ Fine-tuning: **SimulatedChat**

- Agent simulator:
  - **Inputs**: dialog history c_i, knowledge base $\mathcal{KB}$ and previous user utterance u_i
  - **Query Generator** (GPT-2):
    - Generates belief state/query: *domain*, and *key-value* pairs <attribute_name=attribute_value>; with *greedy sampling*
  - **Knowledge Base**:
    - Retrieves a set of results *key-value* pairs <attribute_name=attribute_value>
  - **Response Generator** (GPT-2):
    - Autoregressively generates a pool of candidate utterances
  - **Response Selector** (Longformer):
    - Assigns a context score for each candidate
  - **Output**: Agent utterance



Post-LLM w/ Fine-tuning: **SimulatedChat**

- First, train:
  - Modules for the user bot (Response Generator + Response Selector)
  - Modules for the agent bot (Query Generator + Response Generator + Response Selector)
- Then, fine-tune simulator with 5-20% crowdsourced data
- Finally, concatenate the generated data with the original 5-20% data and train a *student* model
- Compare *student* with baselines

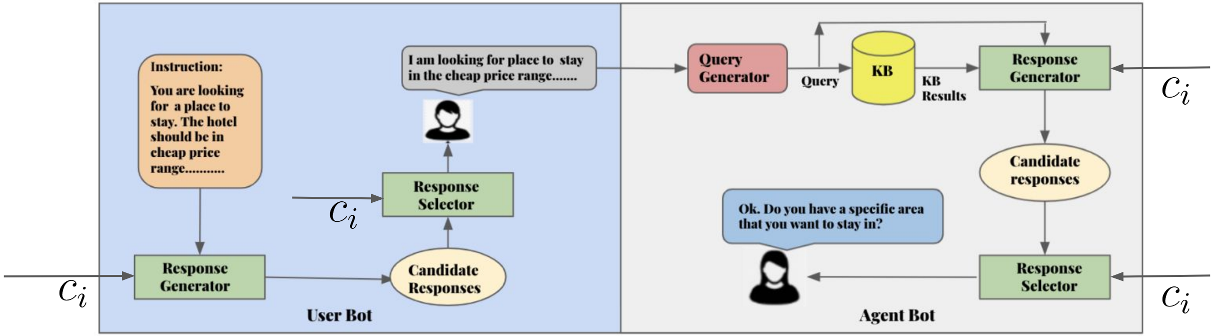


Post-LLM w/ Fine-tuning: **SimulatedChat**

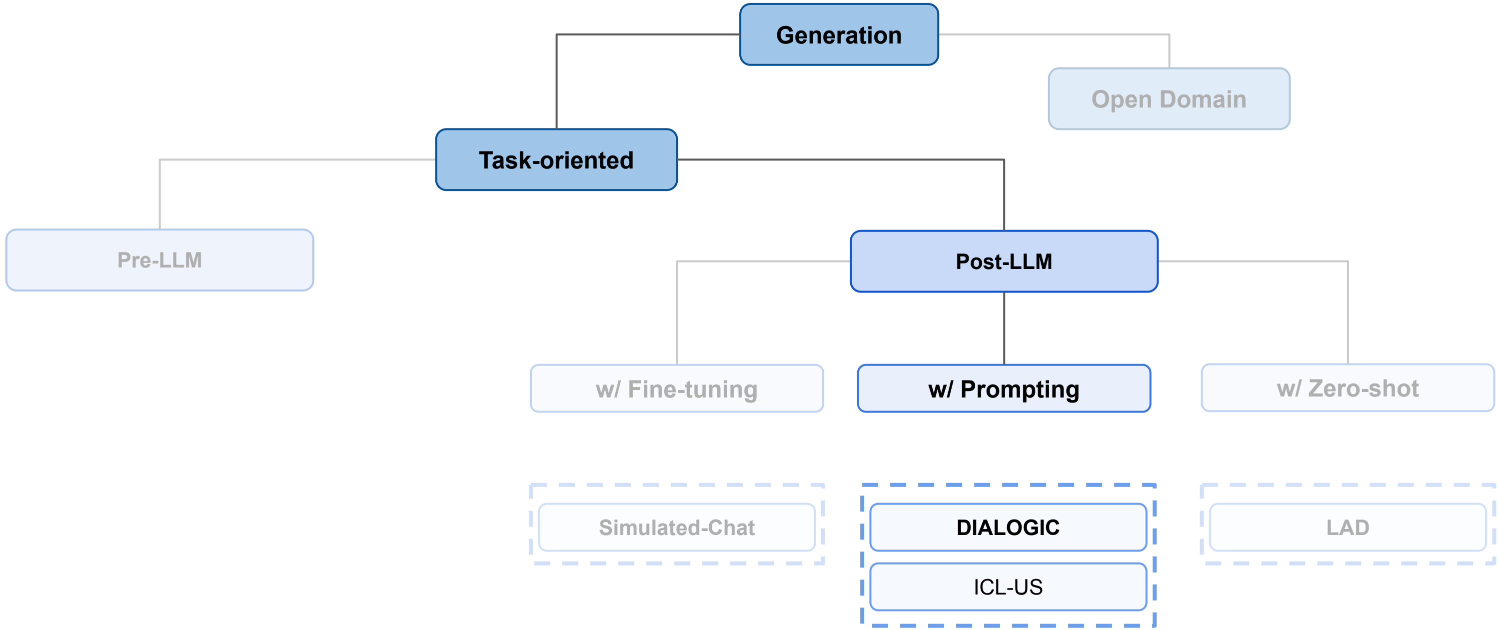
BOT	MODULE	INPUT	OUTPUT
User	Response Generator	[CLS]You are looking for a….[GOAL][St@rt][User] I need a train to Cambridge.[Agent] Sure, which day would you like to travel?[SEP]	[User] I would like to travel this [DAY] at [TIME].
	Response Selector	[CLS][St@rt][User] I need a....[Agent] Sure, which day would you like to travel?[CAN][User] I would like to travel this [DAY] at [TIME].[SEP]	0.92
Agent	Query Generator	[CLS][St@rt][User] I need a....[Agent] Sure, which day would you like to travel?[User] I would like to travel this Sunday at 9PM.[SEP]	[Q] Train | Destination=Cambridge | Day=Sunday | Time=9PM [Q]
	Response Generator	[CLS][Q] Train | Destination=Cambridge | Day=Sunday | Time=9PM [KB] Total = 2 [St@rt][User] I need a....[Agent] Sure,.....travel?[User] I would…. at 9PM.[SEP]	[Agent] I can see [value_count] trains available for [train_destination]. How many tickets do you need?
	Response Selector	[CLS][St@rt][User] I need a....[Agent] Sure….travel? [User] I would….. at 9PM.[CAN][Agent] I can….for [train_destination]. How many tickets do you need?[SEP]	0.83

Post-LLM w/ Fine-tuning: Challenges

- Requires a lot of computational resources and time for fine-tuning
- When using a complex framework (like this one) we also need to pre-train modules for specific tasks (Response Generator/Selector, Query Generator)



Overview

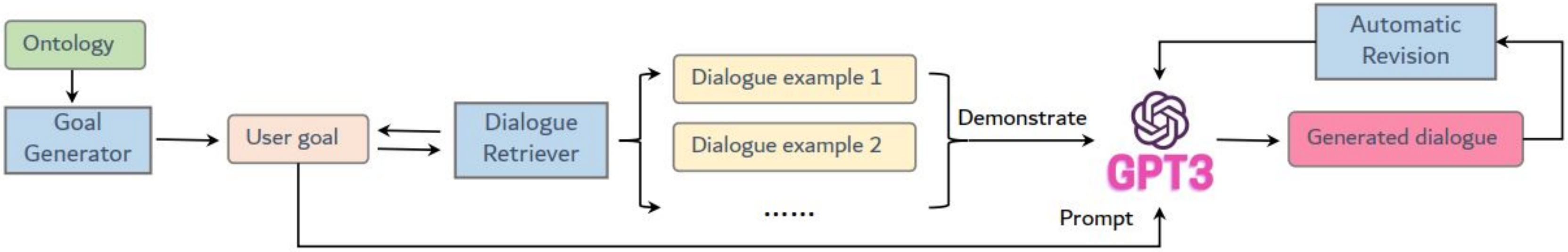


Generation
Open Domain
Task-oriented
Pre-LLM
Post-LLM
w/ Fine-tuning
w/ Prompting
w/ Zero-shot
Simulated-Chat
DIALOGIC
ICL-US
LAD

Post-LLM w/ Prompting: **DIALOGIC**

- LLM (GPT-3) with prompting
- Dialogue Simulation
- Generative Data Augmentation

- In-context generation by prompting with similar examples from a small dataset
- Human involvement is limited: small seed dataset creation



Post-LLM w/ Prompting: **DIALOGIC**

- Inputs:
    - Ontology $\mathcal{O}$ (for each domain; slots and possible values)
    - Database DB
    - Small seed dataset $\mathcal{D}_s$

Post-LLM w/ Prompting: **DIALOGIC**

Methodology:

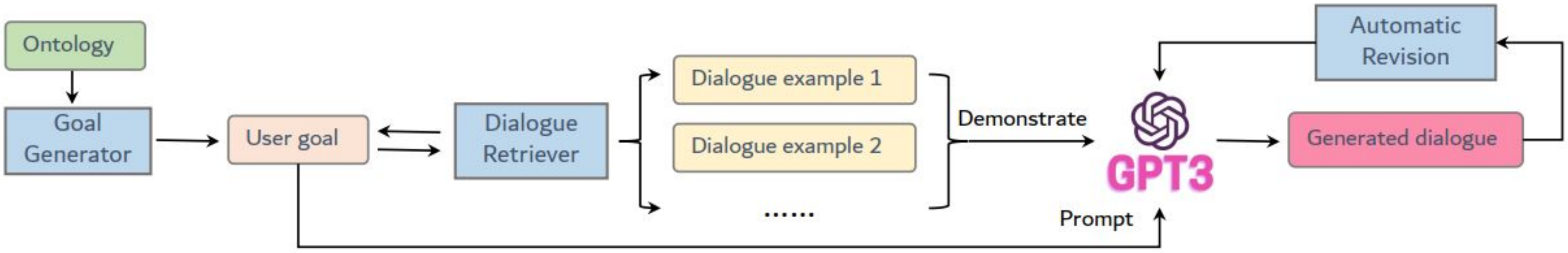
1. For specified domain, pick ontology $\mathcal{O}_j$

Post-LLM w/ Prompting: **DIALOGIC**

Methodology:

1. For specified domain, pick ontology $\mathcal{O}_j$
2. Generate target goal $\mathcal{G}_i$: random sampling, value substitution or combination selection
   Select in-context dialogues $\mathcal{D}_{s,i}$ from seed dataset $\mathcal{D}_s$
   a. Select dialogues whose goals contain as many common slots as possible with $\mathcal{G}_i$
   b. Measured with Jaccard similarity of domain set, and slot set + temp. softmax

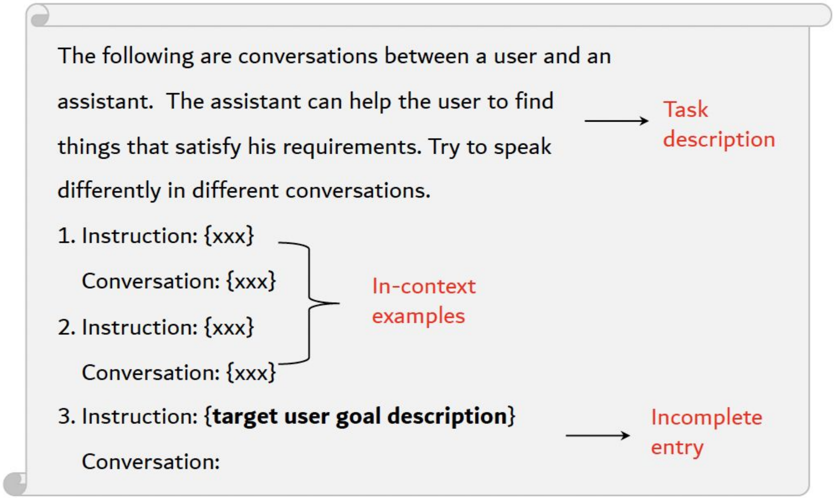
$$w_{ij} = \left|\frac{D(\mathcal{G}_i)\bigcap D(\mathcal{G}_j)}{D(\mathcal{G}_i)\bigcup D(\mathcal{G}_j)}\right| \cdot \left|\frac{S(\mathcal{G}_i)\bigcap S(\mathcal{G}_j)}{S(\mathcal{G}_i)\bigcup S(\mathcal{G}_j)}\right|$$



Post-LLM w/ Prompting: **DIALOGIC**

Methodology:

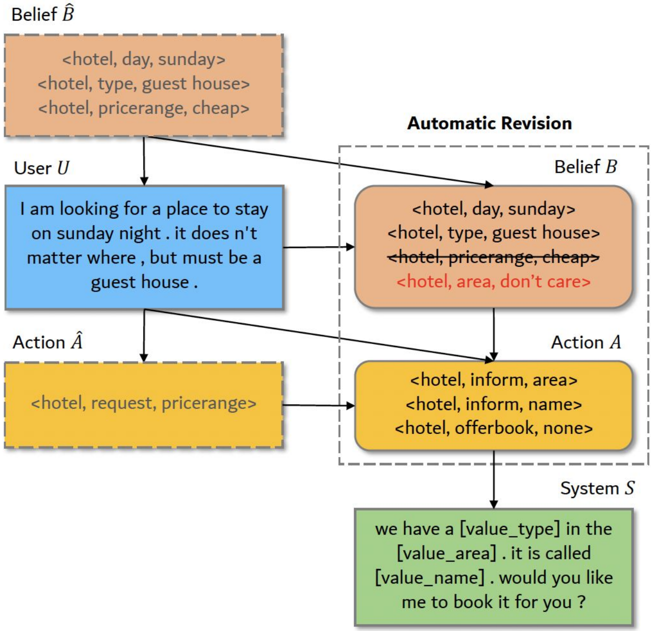
1. For specified domain, pick ontology $\mathcal{O}_j$
2. Generate target goal $\mathcal{G}_i$
   Select in-context dialogues $\mathcal{D}_{s,i}$ from seed $\mathcal{D}_s$ dataset
3. Prompt GPT-3 with $\mathcal{G}_i$ and $\mathcal{D}_{s,i}$
   - Each entry in $\mathcal{D}_{s,i}$ has a goal and dialogue
   - Task description



Post-LLM w/ Prompting: **DIALOGIC**

Methodology:

1. For specified domain, pick ontology $\mathcal{O}_j$
2. Generate target goal $\mathcal{G}_i$
   Select in-context dialogues $\mathcal{D}_{s,i}$ from seed dataset $\mathcal{D}_s$
3. Prompt GPT-3 with $\mathcal{G}_i$ and $\mathcal{D}_{s,i}$
4. GPT-3 generates dialogue $\mathcal{C}_i$
5. Apply automatic verification and revision
   - Control GPT-3 predictions due to reliability issues
   - Way to manipulate over- and under- generation in the belief state



Post-LLM w/ Prompting: **DIALOGIC**

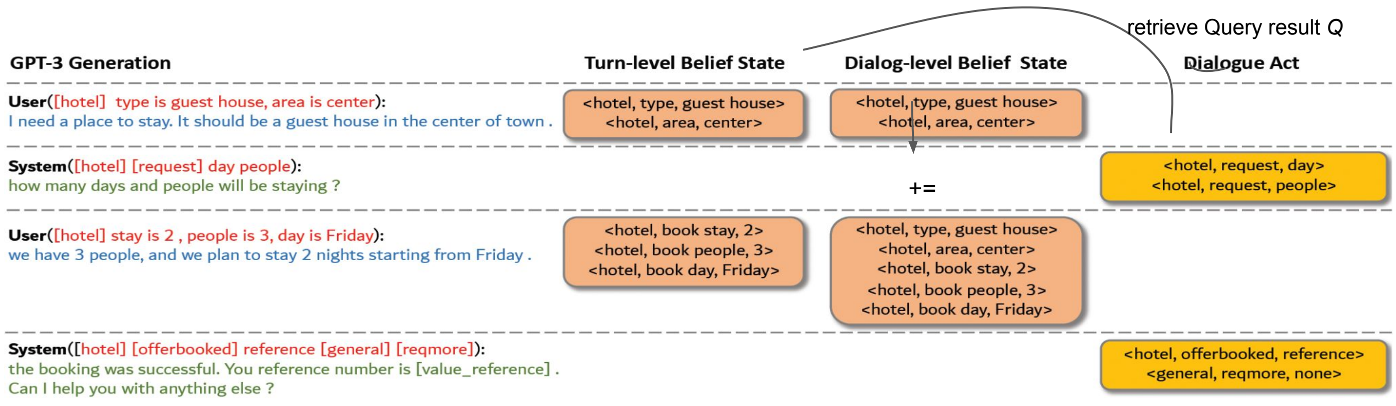
Methodology:

1. For specified domain, pick ontology $\mathcal{O}_j$
2. Generate target goal $\mathcal{G}_i$
3. Select in-context dialogues $\mathcal{D}_{s,i}$ from seed dataset $\mathcal{D}_s$
4. Prompt GPT-3 with $\mathcal{G}_i$ and $\mathcal{D}_{s,i}$
5. GPT-3 generates dialogue $\mathcal{C}_i$
6. Apply automatic verification and revision
7. DST Task: keep track of the accumulation of belief states

Post-LLM w/ Prompting: **DIALOGIC**

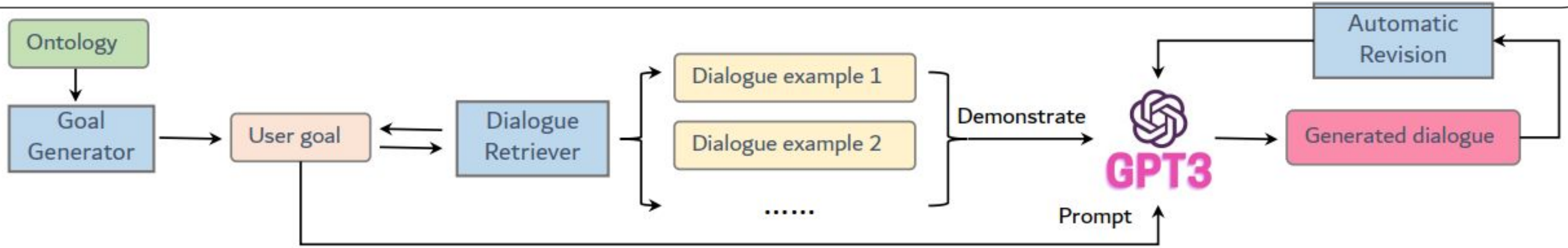
- Inputs:
  - Ontology $\mathcal{O}$ (for each domain; slots and possible values), database DB
  - Small seed dataset $\mathcal{D}_s$
- At each turn, the pipeline generates:

  User utterance *U* with annotations, → Belief state *B*, → Query result *Q*, → Dialog act *A*, → System response *S* with annotations



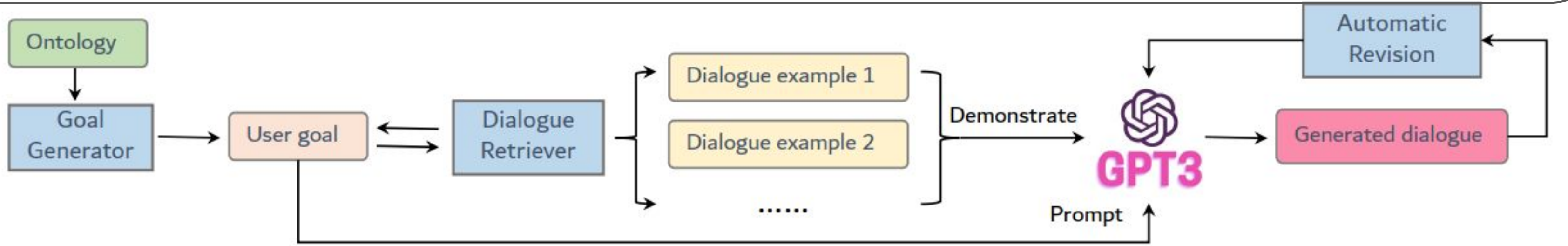
Post-LLM w/ Prompting: **DIALOGIC**

- Datasets:
  - MultiWOZ, MultiWOZ 2.3
  - Simulate low-resource setting by using 1/5/10% of the training dataset (86/422/843 dialogues)
  - Cost comparison:
    - MultiWOZ ~ 30k$
    - DIALOGIC data construction ~0.006$/training sample and 8,438 samples
      - 1% training dataset -> 0.3k + 50$ ~ 0.3k



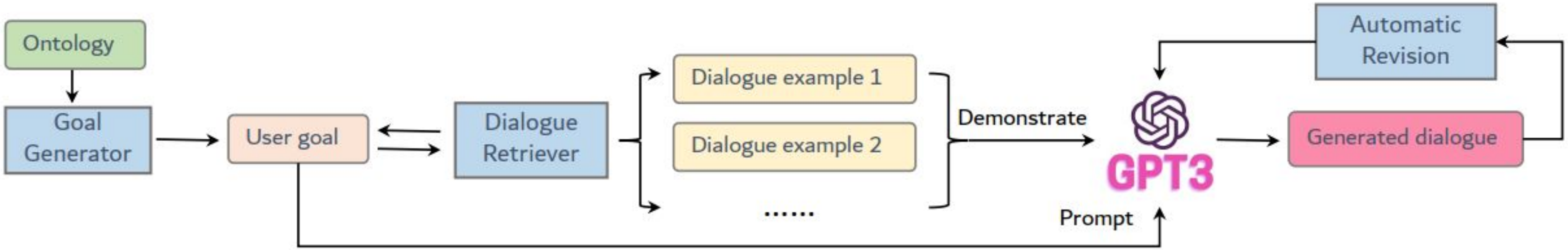
Post-LLM w/ Prompting: **DIALOGIC**

- DIALOGIC vs ICL-US:
  - Similarities:
    - Prompting, generation, simulation, GPT-3
    - Goal generator, Prompt builder, Dialogue Evaluation and Revision
    - In-context learning with k-shot dialogue samples based on Jaccard similarity
  - Differences:
    - DIALOGIC goal generator and dialogue retriever are in parallel; in ICL-US it's sequential

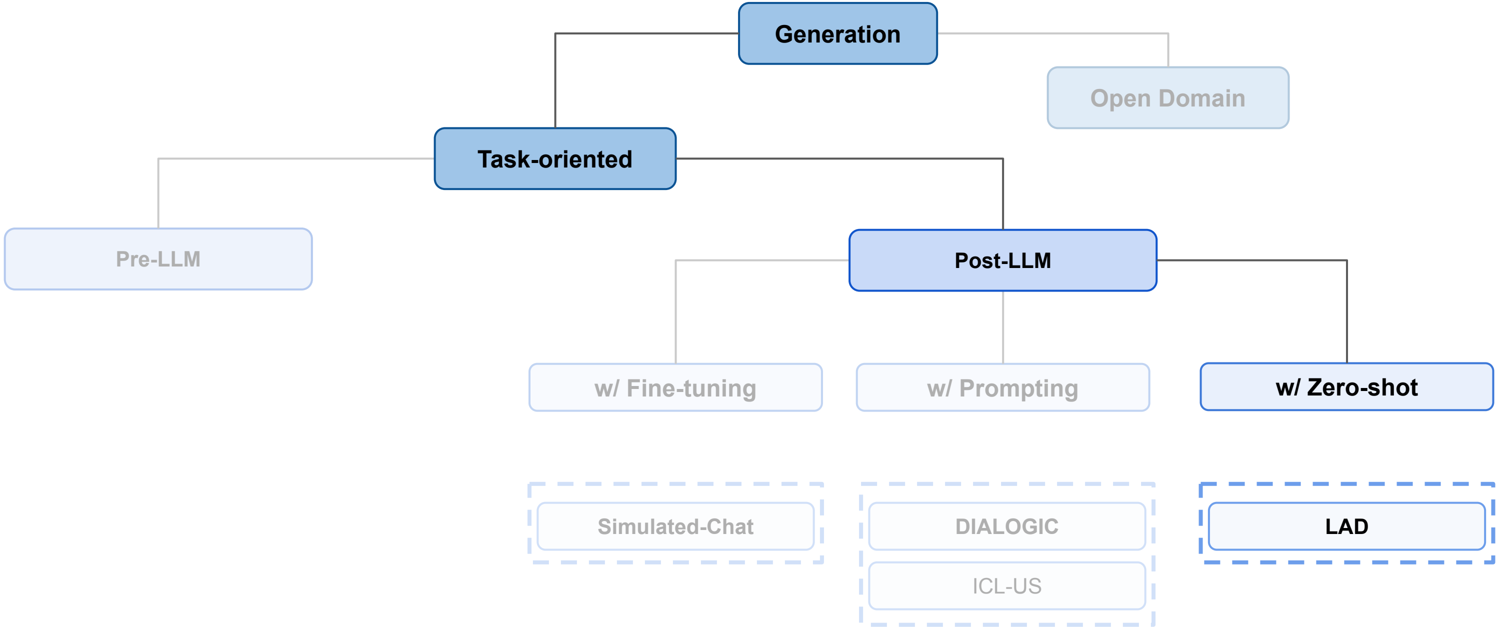


Post-LLM w/ Prompting: Challenges

- Providing a few in-context examples (less than a dozen as in DIALOGIC) is not enough to encapsulate domain constraints
- Plus, it brings biases by hand-designing a prompt that artificially best fits what we want from the model



Overview


Generation
Open Domain
Task-oriented
Pre-LLM
Post-LLM
w/ Fine-tuning
w/ Prompting
w/ Zero-shot
Simulated-Chat
DIALOGIC
ICL-US
LAD

Post-LLM w/ Zero-shot: **LAD**

- LLMs (GPT-3) with zero-shot generalization
- Tasks: intent prediction, slot filling, next action prediction
- Human Involvement is minimal: create a *schema* for encapsulating new domain/task constraints (they go back to a *schema-based* approach)
- Contains a generation validation step (same as DIALOGIC)

Post-LLM w/ Zero-shot: **LAD**

- Tasks: intent prediction, slot filling, next action prediction
  - **Intent prediction** (utterance-level): a model $\mathcal{M}_{\mathcal{I}}$ maps utterances to their goal
  - **Slot filling** (utterance and span-level): a model $\mathcal{M}_{\mathcal{S}}$ maps similar slot values $w_{i:i+k}$ to similar representations
  - **Next action prediction** (utterance and span-level): a model $\mathcal{M}_{\mathcal{A}}$ maps the set of instructions I and slots *S* from the current dialogue history to an action a through a policy: $a = policy(\mathcal{I}_{\mathcal{D}}, \mathcal{S}_{\mathcal{D}})$; therefore, this task requires solving **Intent prediction, Slot filling** and finding the **policy function**

Post-LLM w/ Zero-shot: **LAD**

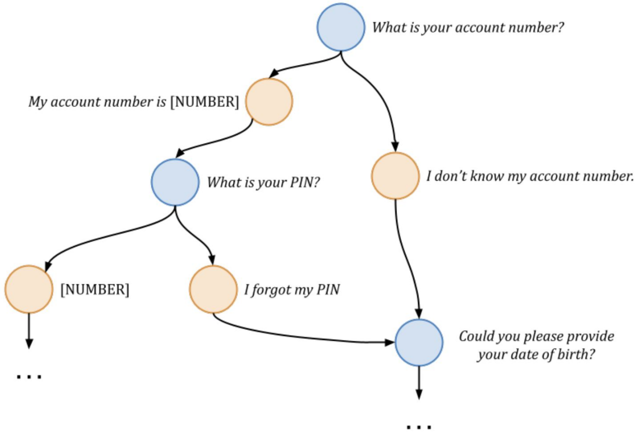
- Schemas: intent prediction, slot filling, next action prediction
  - **Intent prediction** (utterance-level): one utterance per intent
  - **Slot filling** (utterance and span-level): one utterance per intent AND one utterance per slot type + using multiple slot values
  - **Next action prediction** (utterance and span-level): the previous + graph representation

  (Mosig et al., 2020; Mehri and Eskenazi 2021)

Post-LLM w/ Zero-shot: **LAD**

- Schemas: intent prediction, slot filling, next action prediction
  - **Intent prediction** (utterance-level): one utterance per intent
  - **Slot filling** (utterance and span-level): one utterance per intent AND one utterance per slot type + using multiple slot values
  - **Next action prediction** (utterance and span-level): the previous + graph representation

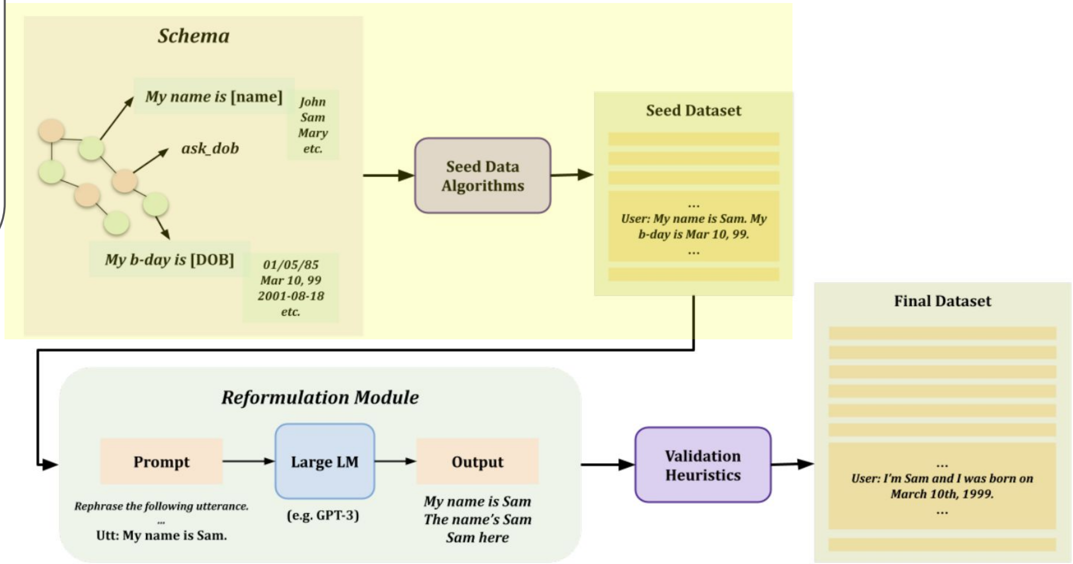
(Mosig et al., 2020; Mehri and Eskenazi 2021)



Post-LLM w/ Zero-shot: **LAD**

- Methodology:
  1. Seed Data Creation

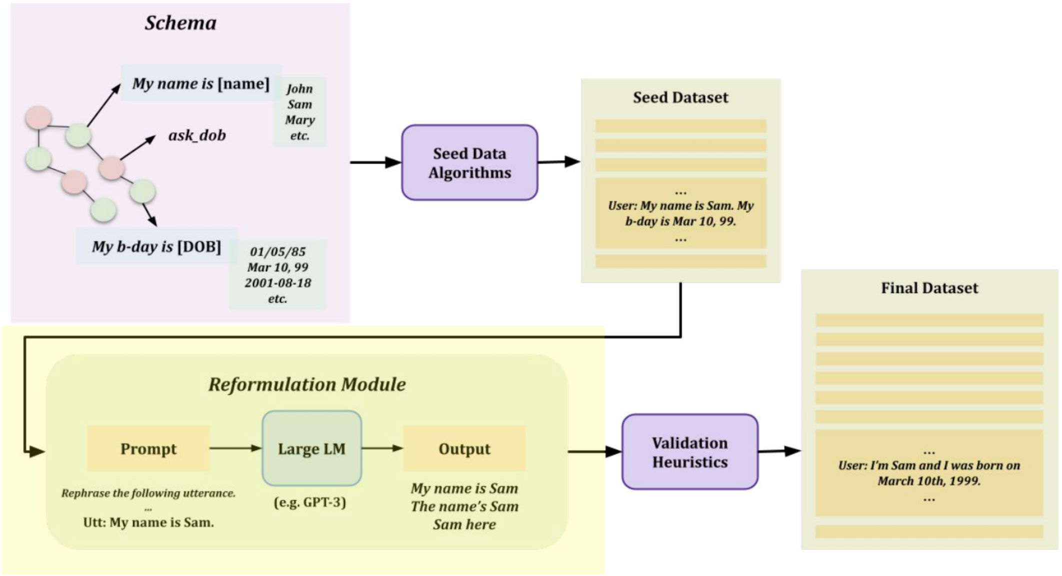
     Traverse *schema* to generate seed utterances and form initial dataset



Post-LLM w/ Zero-shot: **LAD**

- Methodology:
  1. Seed Data Creation
  2. Reformulation

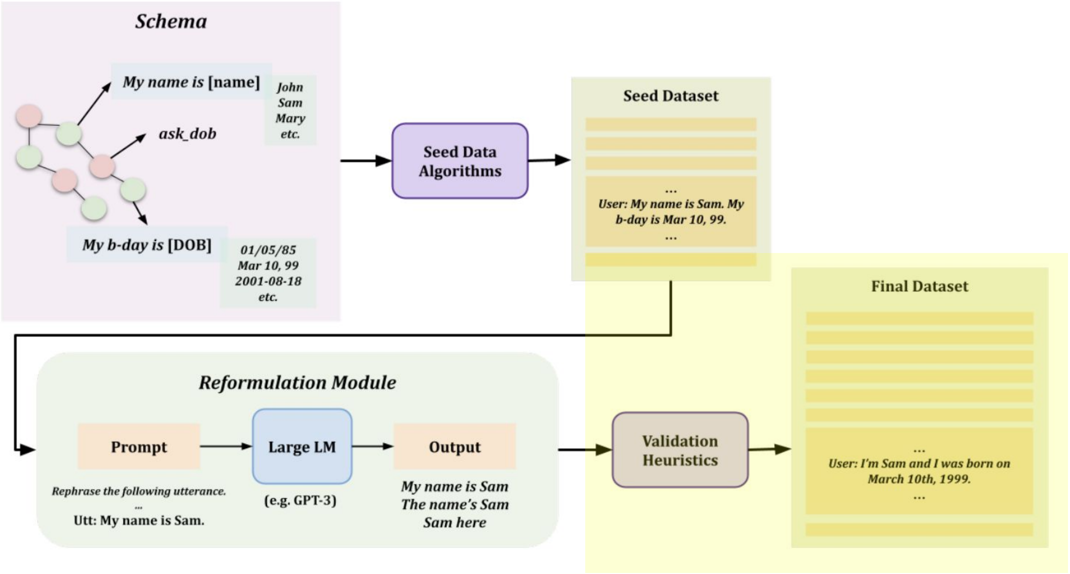
     Induce linguistic diversity by rephrasing the seed dataset entries into multiple version of the same utterance



Post-LLM w/ Zero-shot: **LAD**

- Methodology:
  1. Seed Data Creation
  2. Reformulation
  3. Validation

     Similarly to DIALOGIC, ensure the structural constraints are kept by ensuring all *slot values* present in the original utterance are also present in the reformulated one



Post-LLM: Can we compare these models?

- SimulatedChat:
  - Experiment with: add the augmented data
    - MultiWOZ 2.0 (experiment with two e2e models):
      - Soloist (init with GPT2-small): transformer auto-regressive model
      - **MinTL-T5 (init with T5-small): transformer**
    - PersonaChat:
      - GPT2-small (with augmentation)
  - Baselines:
    - MultiWoz 2.0:
      - DAMD (non-augmentation recent baseline)
      - DAMD-MADA (dialog-states based augmentation)
      - PARG-TSCP
    - PersonaChat:
      - GPT2-small (wo augmentation)
- DIALOGIC:
  - Experiment with: same models, but add the augmented data
  - Baselines:
    - MultiWOZ 2.3:
      - SimpleTOD (init with GPT2-small)
      - **MinTL (init with T5-small)**
      - PPTOD (init with T5-small)
- LAD:
  - Experiment with: same models, but add the augmented data
  - Baselines:
    - Intent Prediction:
      - ConvBERT (CBEO)
    - Slot Filling:
      - GenSF
    - Next Action Prediction:
      - SAM

Post-LLM: Can we compare these models?

MODELS	5%				20%				100%			
	B	I	S	C	B	I	S	C	B	I	S	C
MinTL-T5-Small (Lin et al., 2020b)	12.5	50.9	33.9	55.7	15.8	63.5	48.8	72.0	17.4	80.1	**64.7**	**89.8**
MinTL-T5-Small (Sim. Aug)	13.1	57.6	36.1	60.0	**16.0**	68.0	55.1	76.6	**18.5**	79.5	57.1	86.8

SimulatedChat; MultiWOZ 2.0

Seed data	**Augmented data**	**MinTL-T5**			
		I	S	B	C
1% (85)	Base	56.81	40.38	12.16	60.76
	+Orig.(85)	64.93	50.20	12.37	70.13
	+Sim.(85)	**69.44**	**50.30**	**12.46**	**72.33**
5% (422)	Base	74.05	60.42	14.71	82.70
	+Orig.(422)	72.24	60.42	**14.91**	81.24
	+Sim.(422)	**77.45**	**64.93**	13.98	**85.17**
10% (843)	Base	72.24	60.42	14.91	81.24
	+Orig.(843)	78.76	68.74	**15.92**	89.67
	+Sim.(843)	**79.96**	**69.84**	15.41	**90.31**
100% (8438)	Base	80.06	72.85	17.87	94.33
	+Sim.(422)	79.46	73.45	18.52	94.98
	+Sim.(843)	**80.76**	**74.15**	**18.72**	**96.18**

DIALOGIC; MultiWOZ 2.3