# Data Augmentation for Conversational AI

**The Web Conference 2024**

Tutorial website

# Presenters

**Heydar Soudani**
PhD Candidate
Radboud University
heydar.soudani@ru.nl

**Evangelos Kanoulas**
Full Professor
University of Amsterdam
e.kanoulas@uva.nl

**Roxana Petcu**
PhD Candidate
University of Amsterdam
r.m.petcu@uva.nl

**Faegheh Hasibi**
Assistant Professor
Radboud University
f.hasibi@cs.ru.nl

# Part 2: Conversation Generation - Task Oriented

Duration: 45 min

Presenter: Roxana Petcu

# Task-Oriented Dialogue (TOD) System

## Definition

- **Structured interactions** focused on **completing a specific task** and **reaching the user goal**.

## Examples of tasks

- Booking a flight, reserving a restaurant table, asking a chatbot about available non-dairy products at an online supermarket

## Challenges

- Constraints on the task and domain
  - Example: making a restaurant reservation requires adhering to constraints: location availability, matching user's cuisine, table must fit party size
- Diverse user goals
- Lack of specialized datasets

# TOD example

User: Book a restaurant in Orlando for 4 people.

System: What type of food and price range should I look for?

User: I'd like a moderately priced taiwanese restaurant.

System: How about the Formosan Garden restaurant? And at what time do you want the reservation?

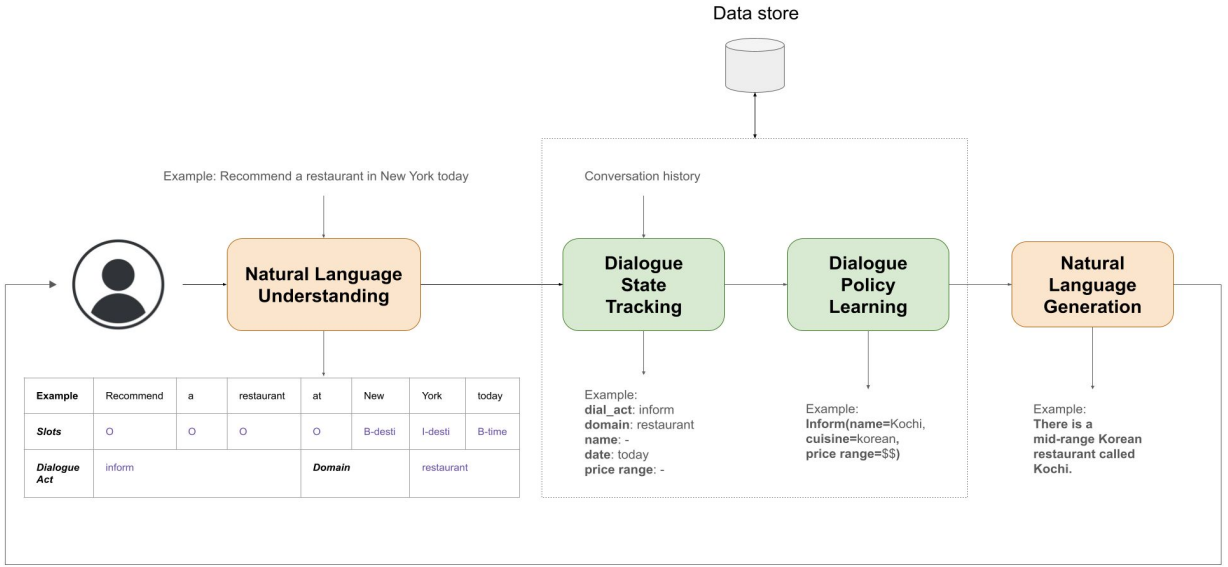
User: Formodan Garden is perfect! Make the reservation at 8PM.

System: I just made a reservation for Formodan Garden in the Centre of Orlando at 8PM. The reservation is made for 4 people.

# TOD Data Generation

## Essential components

- Roughly 4 modules; they can be (1) in an end-to-end fashion, or (2) modular approach (see Figure)

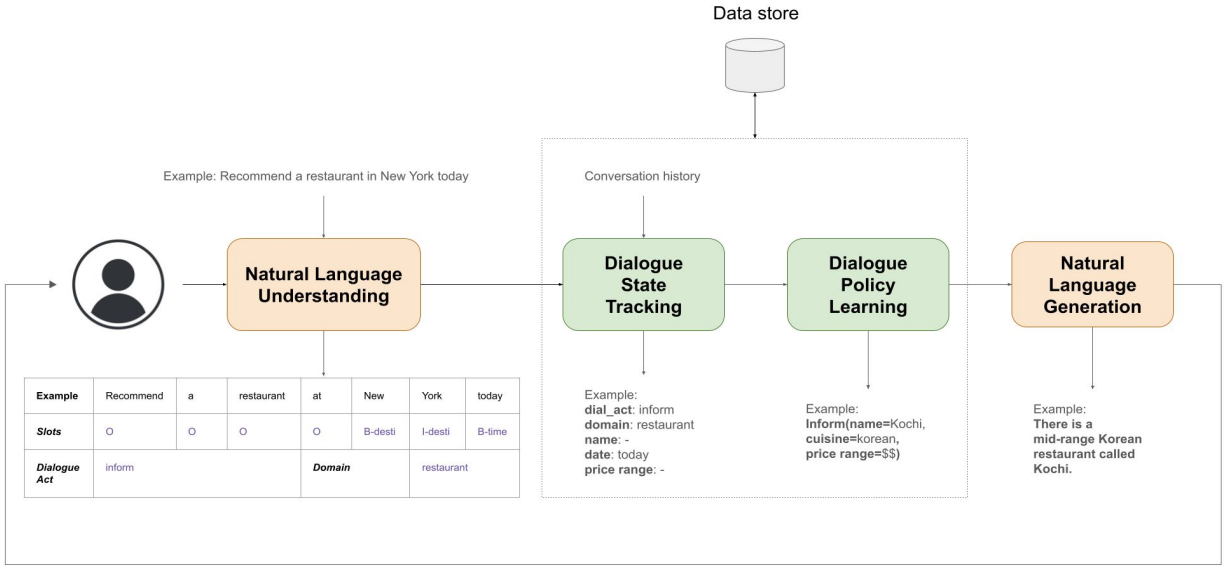


# TOD Data Generation

## Essential components

- Roughly 4 modules; they can be (1) in an end-to-end fashion, or (2) modular approach (see Figure)

**Natural Language Understanding (NLU):** this module receives as input a conversational turn in natural language form. The goal is to process the input and extract intents, slots and values for the identified slots.

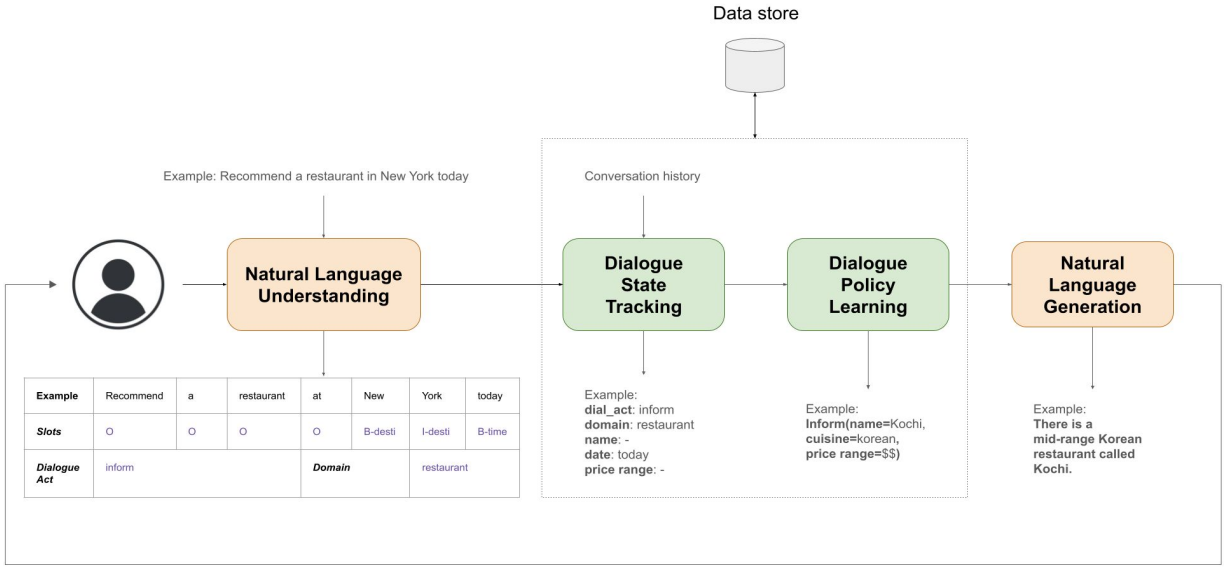


# TOD Data Generation

## Essential components

- Roughly 4 modules; they can be (1) in an end-to-end fashion, or (2) modular approach (see Figure)

**Dialogue State Tracking (DST):** this module receives as input the conversation history and output of the NLU module (which corresponds to the current turn of the dialog) and produces the necessary slots that should be filled to approach the user goal.

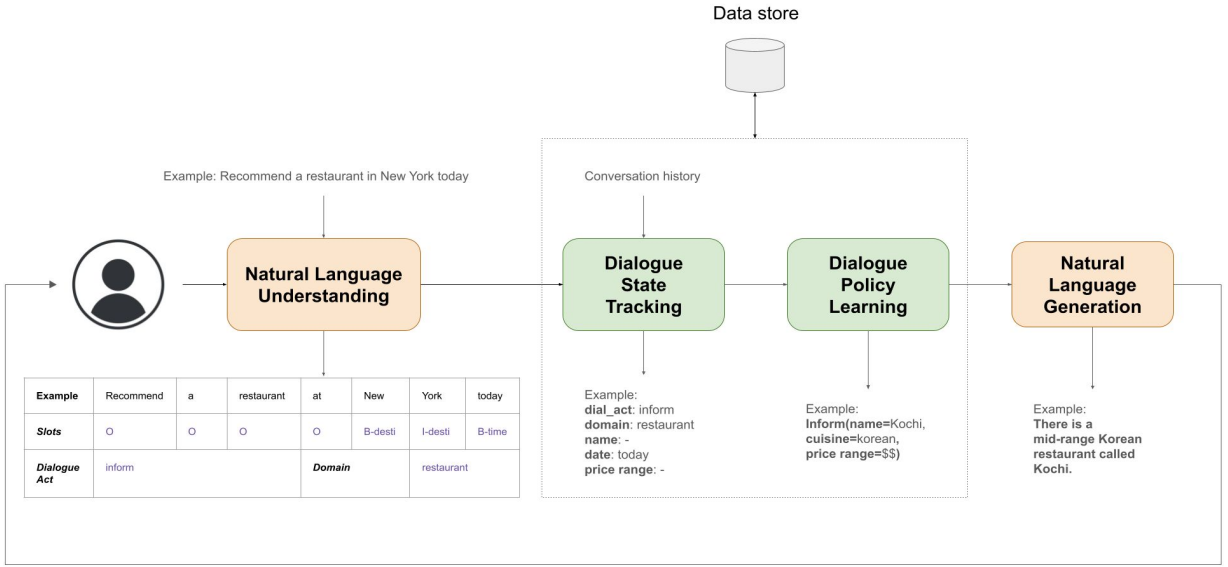


# TOD Data Generation

## Essential components

- Roughly 4 modules; they can be (1) in an end-to-end fashion, or (2) modular approach (see Figure)

**Dialogue Policy Learning (DPL):** receives as input the slots that must be filled in, and outputs values that would be satisfactory next actions based on the current dialogue state.

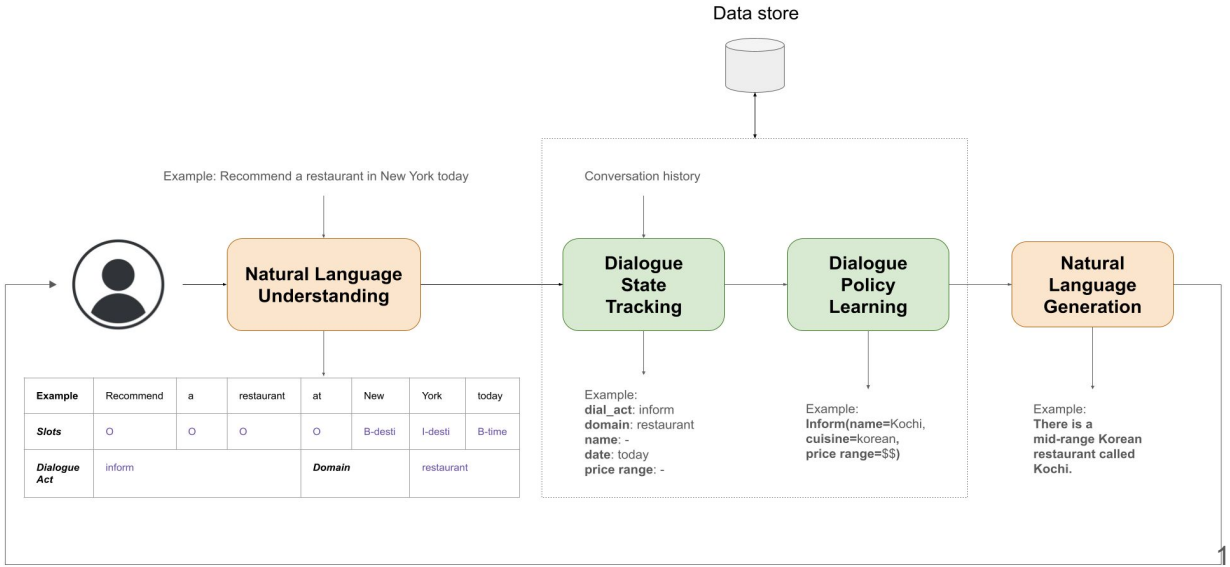


# TOD Data Generation

## Essential components

- Roughly 4 modules; they can be (1) in an end-to-end fashion, or (2) modular approach (see Figure)

**Natural Language Generation:** receives as input the DPL output, and converts it into natural language representation.



# TOD Data Generation - Training

Rule-based systems

Training approaches

- Supervised training
  - Offline training
  - Needs a lot of annotated data (data scarcity problems)
- Reinforcement learning
  - Enables real-time dialogue generation
  - Requires less data
  - **Simulates** real-world interactions

# TOD Data Generation - Simulation

## Simulation

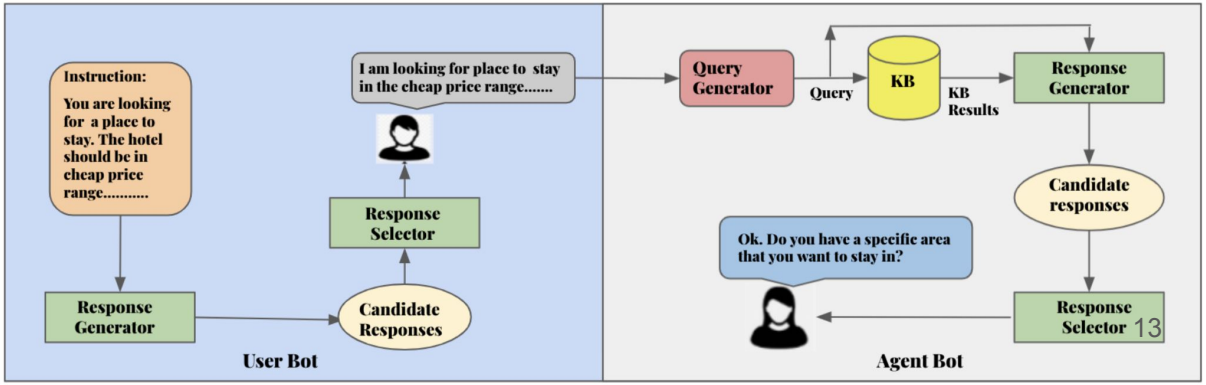
- A conversation inherently involves 2 participants (at least)
- Concept of simulation: have something akin a user to produce part of the dialogue and interact with the dialog system
- A Simulator can be:
  - Pre-trained (One-sided simulation)
  - Co-trained alongside the dialog system (Two-sided simulation)

# TOD Data Generation - Simulation

## Simulation

- Two-sided simulators are usually referred to as:
  - *User Bot* and *Agent Bot* (see Simulated-Chat example)
  - *User Bot* and *System Agent*
  - *User Bot* and *Dialogue System*
  - *User Simulator* and *Dialogue System*
  - ….

(Mohapatra et al., 2021)



# How to split TOD Generation systems?

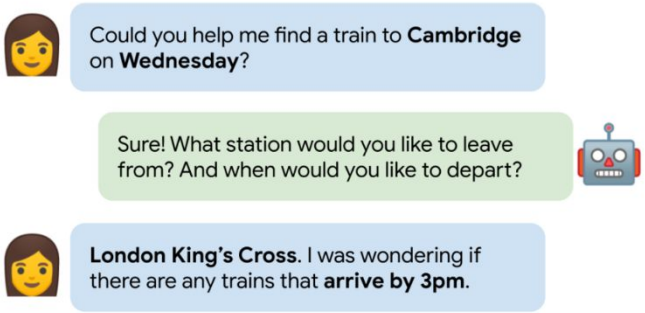
## **Where** to get the input?

slots and values

| Slot Description | Value |
|---|---|
| Train destination | **Cambridge** |
| Train departure | **London King's Cross** |
| Time the train should arrive by | **3pm** |
| Time the train should leave by | (unspecified) |
| Day the train should run | **Wednesday** |

Provided?
Discovered?

## **What** to generate?

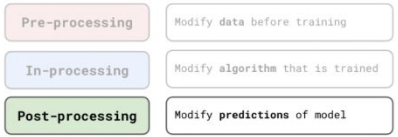



Utterance?
Dialog?

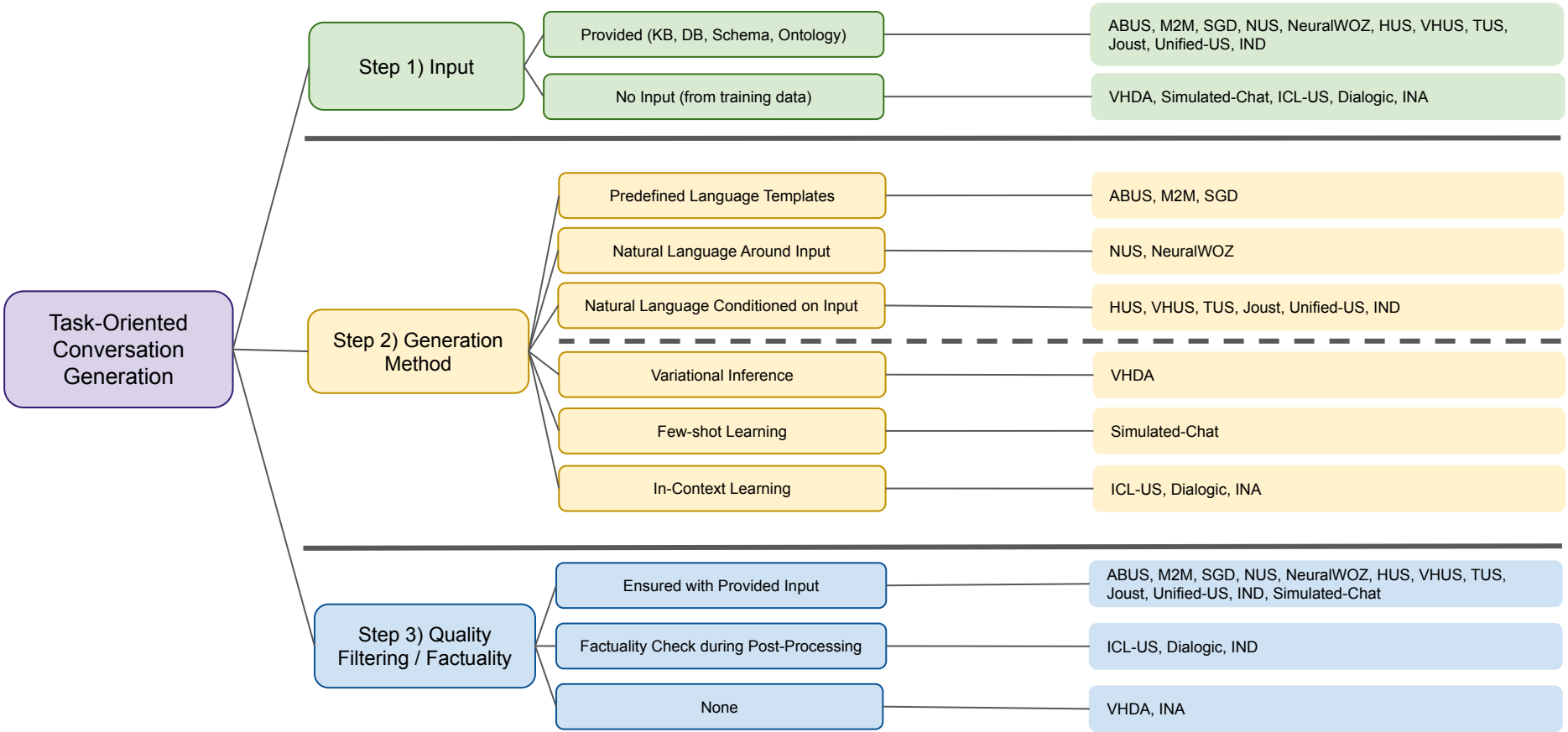
## **How** to verify?

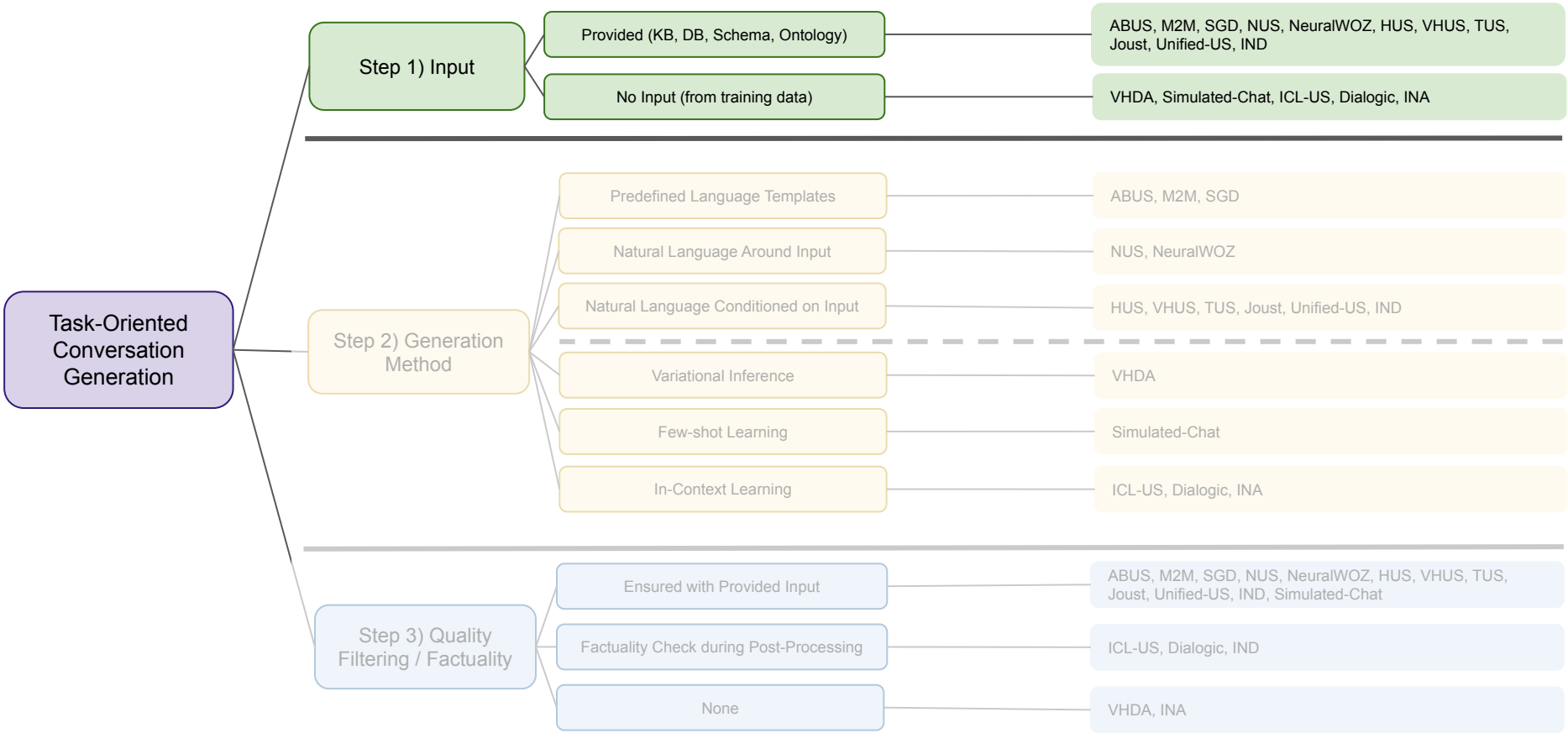
With Input?

Post processing?




None?


Task-Oriented Conversation Generation
Step 1) Input
Provided (KB, DB, Schema, Ontology)
ABUS, M2M, SGD, NUS, NeuralWOZ, HUS, VHUS, TUS, Joust, Unified-US, IND
No Input (from training data)
VHDA, Simulated-Chat, ICL-US, Dialogic, INA
Step 2) Generation Method
Predefined Language Templates
ABUS, M2M, SGD
Natural Language Around Input
NUS, NeuralWOZ
Natural Language Conditioned on Input
HUS, VHUS, TUS, Joust, Unified-US, IND
Variational Inference
VHDA
Few-shot Learning
Simulated-Chat
In-Context Learning
ICL-US, Dialogic, INA
Step 3) Quality Filtering / Factuality
Ensured with Provided Input
ABUS, M2M, SGD, NUS, NeuralWOZ, HUS, VHUS, TUS, Joust, Unified-US, IND, Simulated-Chat
Factuality Check during Post-Processing
ICL-US, Dialogic, IND
None
VHDA, INA


Task-Oriented Conversation Generation
Step 1) Input
Provided (KB, DB, Schema, Ontology)
ABUS, M2M, SGD, NUS, NeuralWOZ, HUS, VHUS, TUS, Joust, Unified-US, IND
No Input (from training data)
VHDA, Simulated-Chat, ICL-US, Dialogic, INA
Step 2) Generation Method
Predefined Language Templates
ABUS, M2M, SGD
Natural Language Around Input
NUS, NeuralWOZ
Natural Language Conditioned on Input
HUS, VHUS, TUS, Joust, Unified-US, IND
Variational Inference
VHDA
Few-shot Learning
Simulated-Chat
In-Context Learning
ICL-US, Dialogic, INA
Step 3) Quality Filtering / Factuality
Ensured with Provided Input
ABUS, M2M, SGD, NUS, NeuralWOZ, HUS, VHUS, TUS, Joust, Unified-US, IND, Simulated-Chat
Factuality Check during Post-Processing
ICL-US, Dialogic, IND
None
VHDA, INA

# Component 1: Input

**TOD systems are constructed around:**

- **Entities** like *Restaurant, Customer, or Movie*
- Based on the entity, there are:
  - **Slots** like *Cuisine, Party Size, Date, Time*
  - **Slot Values** like *French, 2 people, January 25th, 19:00*
- **Entities, Slots,** and **Slot Values** are usually extracted from some **predefined knowledge** that contains information that *Cuisine* can be *French* but cannot be *Metallic*; or that *Time* can be *19:00* but cannot be *25:00*
- **Predefined knowledge** is usually represented in graphical structures such as:
  - **Schema / Ontology**
  - **Knowledge Graph / Database**

# Component 1: Input

**Schema** / Ontology:

- Contains **entities**, and **slots**

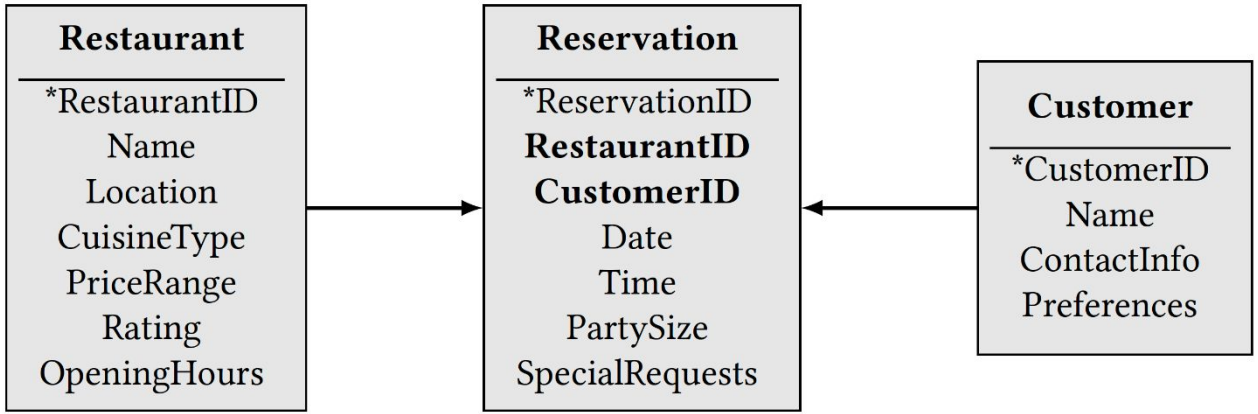



Fig. 4. Example of a schema for the restaurant reservation task; each table represents a class (entity) with its attributes (slots); * indicates the primary key (mandatory for each class), and **boldface** indicates the foreigner key used to connect two classes.

# Component 1: Input

**Schema / Ontology:**

- Contains **entities**, **slots**, and the **relationship between entities** (ex: *Reservation "is made by" Customer*)

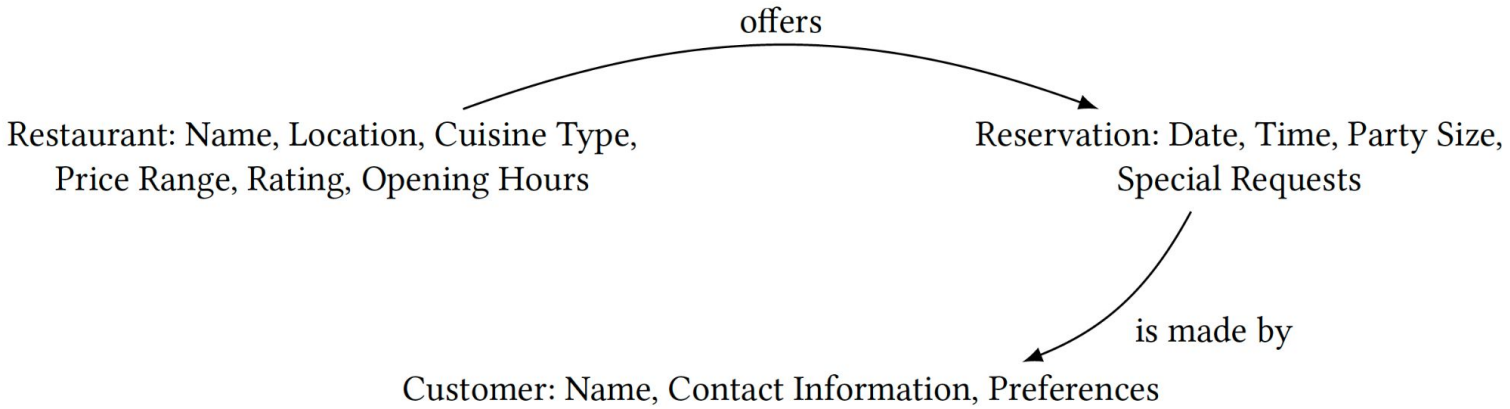



Fig. 5. Example of an ontology for the restaurant reservation task.

# Component 1: Input

## Schema / Ontology:

- Goal: The dialog system can use these general structures to ask relevant questions. They contain information about the **semantics** of the dialogue and **not** about **instantiations** of entities.
- Limitation: General structures do not contain real-world data or restrictions on the possible slot values. For data generation, this means that a dialogue may evolve around combinations of slot values that do not exist, e.g. a restaurant called *Moeders* that specializes in *japanese cuisine*.

# Component 1: Input

**Database** / Knowledge Graph **:**

- Contains **entities**, **slots,** and **values**

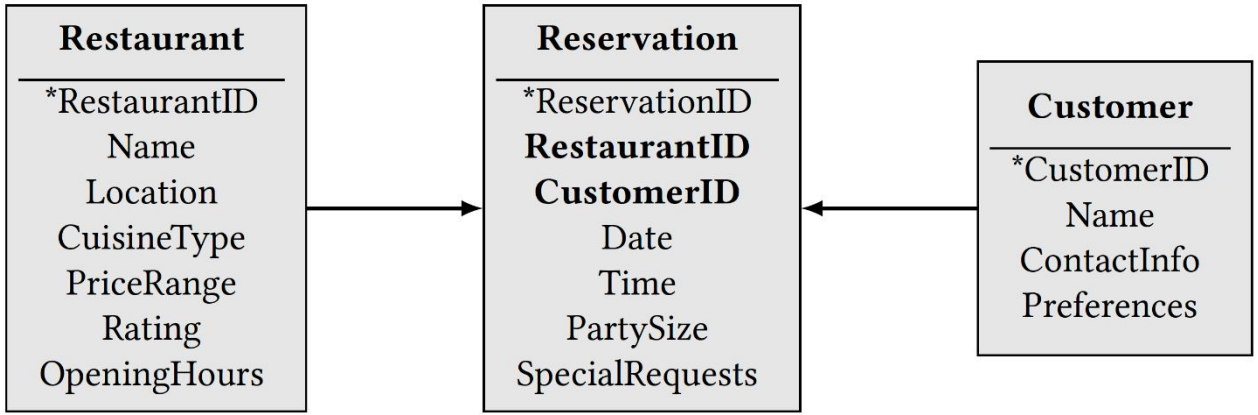
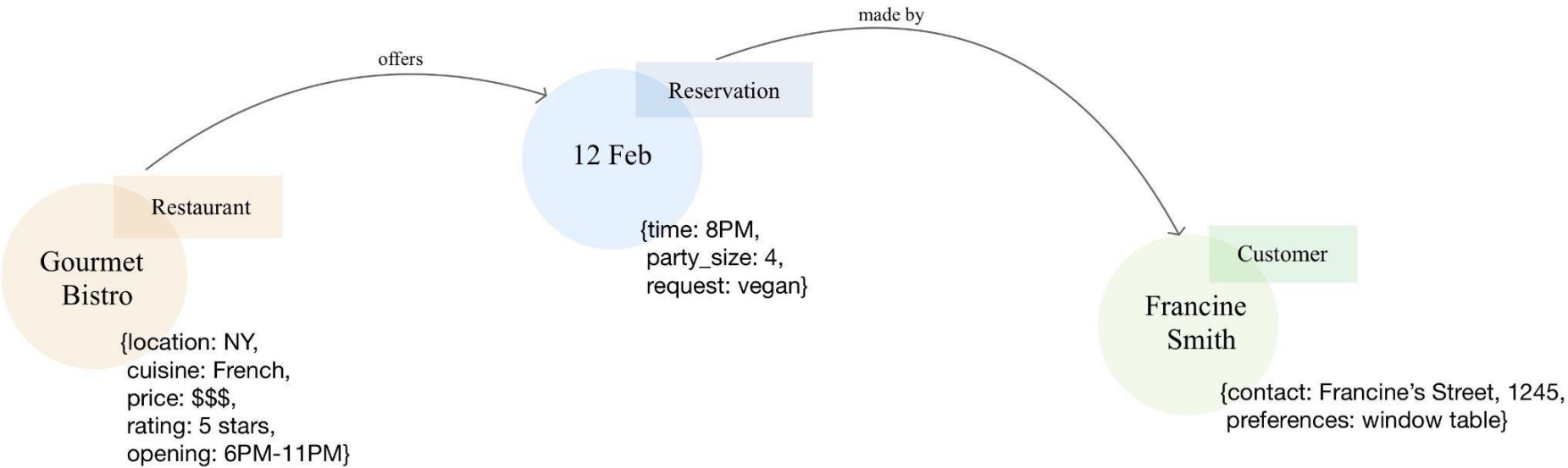



Fig. 4. Example of a schema for the restaurant reservation task; each table represents a class (entity) with its attributes (slots); * indicates the primary key (mandatory for each class), and **boldface** indicates the foreigner key used to connect two classes.

# Component 1: Input

**Database / Knowledge Graph :**

- Contains **entities**, **slots,** the **relationship between entities,** and **values**



# Component 1: Input

## Database / Knowledge Graph :

- Goal: Links to real entities and is updated in real-time.
- Limitation: Difficult to build for every problem.

**NOTE:**

- DB is an instantiation of a Schema
- KG is an instantiation of an Ontology

# Component 1: Input

## TOD key terms:

- **Intent**
- **Dialogue Act**
- **(User) Goal**
- **Dialogue Frame**

  *Inform*<date="tomorrow", time="8PM", restaurant="LaCongerie", cuisine="french">

- **Belief State / Dialogue State**

  *Inform*<date="tomorrow", time="8PM", restaurant="LaCongerie", cuisine="french">, *Request*<party_size>

User: Book a restaurant in Orlando for 4 people.

System: What type of food and price range should I look for?

User: I'd like a moderately priced taiwanese restaurant.


```
"user_intents": ["BOOK_RESTAURANT"],

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

"user_acts": [
        { "type": "INFORM" }],

"user_goal": [
    "domain": "restaurant",
    "user_intent": ["BOOK_RESTAURANT"],
    {"act": "inform",
        {"slot": "location",        "value": "orlando"},
        {"slot": "price_range",     "value": "moderately priced"},
        {"slot": "category",        "value": "taiwanese"}.
    },
    {"act": "request",
        {"slot": "price_range"},
        {"slot": "category"},
    } ]

"dialog_frame": [
    {"act": "request"},
    {"slot": "date"},
    {"slot": "time"}}]

"belief_state": [
    {"act": "inform",
        {"slot": "location",        "value": "orlando"},
        {"slot": "price_range",     "value": "moderately priced"},
        {"slot": "category",        "value": "taiwanese"}.
    },
    {"act": "request",
        {"slot": "date"},
        {"slot": "time"}
    }]
```

# **Input Provided** vs **No Input**

## Provided:

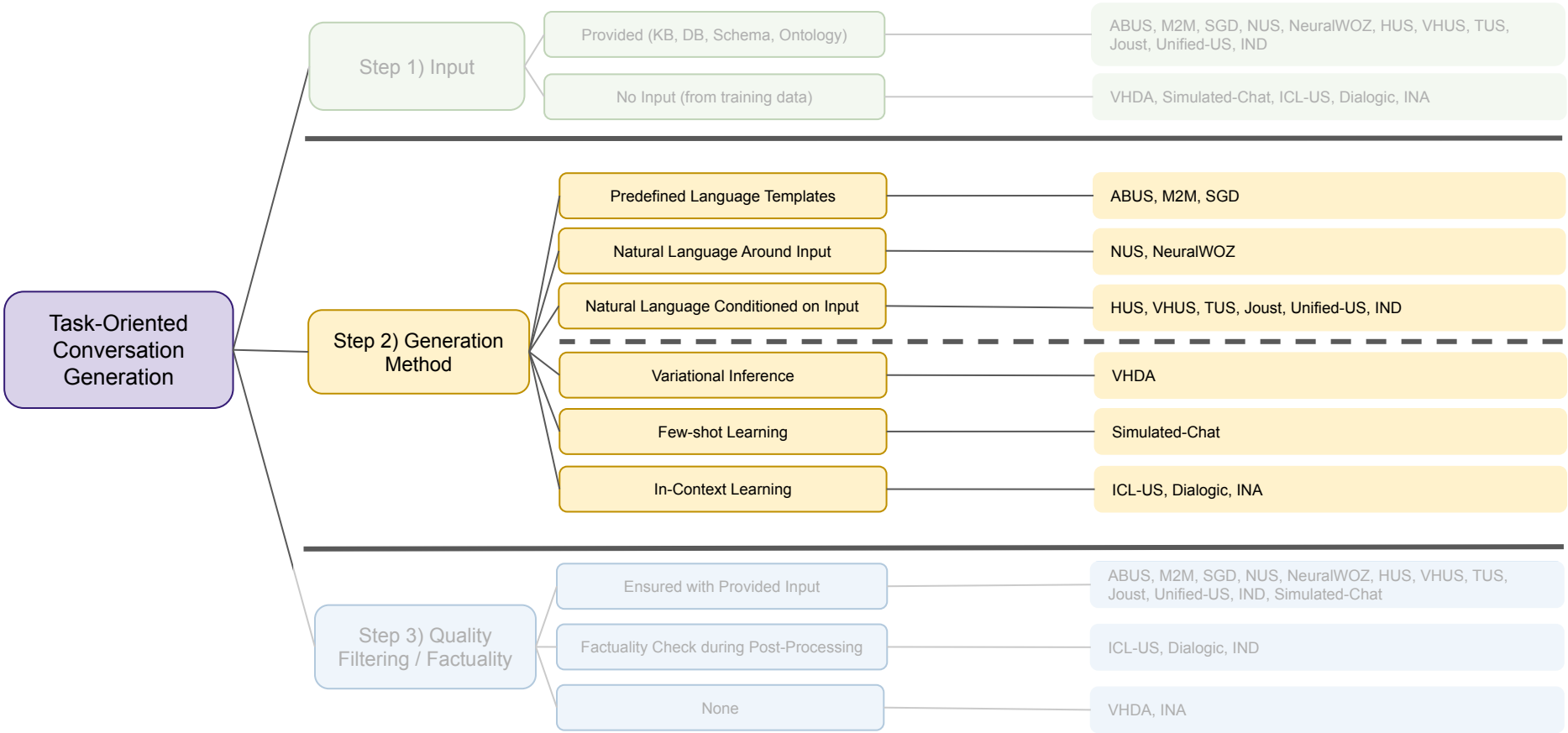
- If provided, slots and slot values are plugged into the dialogue system and natural language are generated around/conditioned on them
- Guarantees factuality

ABUS, M2M, SGD, NUS, NeuralWOZ, HUS, VHUS, TUS, Joust, Unified-US, IND

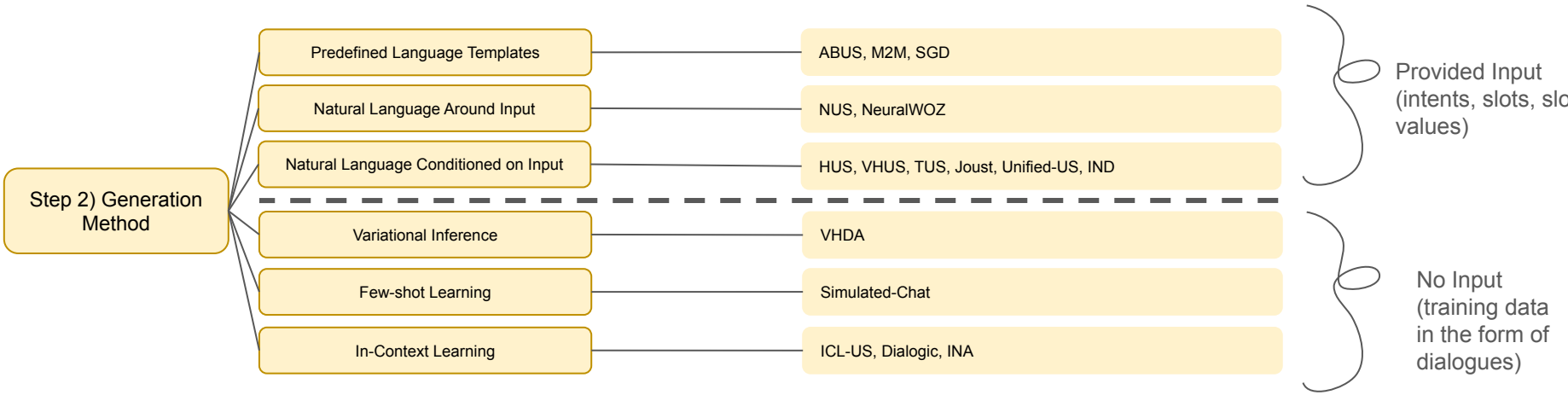
## Not Provided:

- It not provided, the dialogue system must learn them through training
- Does NOT guarantee factuality

VHDA, Simulated-Chat, ICL-US, Dialogic, INA


Task-Oriented Conversation Generation
Step 1) Input
Provided (KB, DB, Schema, Ontology)
ABUS, M2M, SGD, NUS, NeuralWOZ, HUS, VHUS, TUS, Joust, Unified-US, IND
No Input (from training data)
VHDA, Simulated-Chat, ICL-US, Dialogic, INA
Step 2) Generation Method
Predefined Language Templates
ABUS, M2M, SGD
Natural Language Around Input
NUS, NeuralWOZ
Natural Language Conditioned on Input
HUS, VHUS, TUS, Joust, Unified-US, IND
Variational Inference
VHDA
Few-shot Learning
Simulated-Chat
In-Context Learning
ICL-US, Dialogic, INA
Step 3) Quality Filtering / Factuality
Ensured with Provided Input
ABUS, M2M, SGD, NUS, NeuralWOZ, HUS, VHUS, TUS, Joust, Unified-US, IND, Simulated-Chat
Factuality Check during Post-Processing
ICL-US, Dialogic, IND
None
VHDA, INA

# Component 2: Generation Method



# Generation Method - Predefined Language Templates

- Access to intents, slots and slot values that are plugged-in predefined templates
- Referred to as **agenda-based** approaches
- **Task-dependent!**
- Follow a predetermined set of templates (**outlines**) for generating dialog turns
- Example:
  - the **intent** <*book_movie*> can be associated with template *"Book movie with [**name**="value"] and [**date**="value"]"*
  - for Inform<**intent**=*book_movie*, **name**=*Inside Out*, **date**=*tomorrow*>
  - The template is filled and generates the turn *"Book movie with **name** Inside Out and **date** is tomorrow."*
- Paraphrasing can be added to generate more diverse human-like turns:
  - *"I want to buy tickets for Inside Out for tomorrow"*

# Generation Method - Predefined Language Templates

ABUS (Li et al., 2017)

- Input: agenda and example dialogues
  - agenda is used as a stack-like representation for user states
  - example dialogues are used for training a simulator
- Simulator: using RL policy
- Challenge addressed: training dialogue systems to respond accurately and in-real time

# Generation Method - Predefined Language Templates

M2M (Shah et al., 2018)

- Input: multiple agendas and task specification (it has access to multiple APIs, each API has a task-dependent agenda)
- Simulator: using RL policy
- Challenge addressed: enhances generalizability by allowing to scale to new tasks and domains if provided a new API

# Generation Method - Predefined Language Templates

SGD (Rastogi et al., 2020)

- Input: multiple agendas and task specification (it has access to multiple APIs, each API has a task-dependent agenda)
- Simulator: using RL policy
- Challenge addressed: in the real world, multiple services have overlapping functionality. The authors build a single unified model for all services by having dynamic APIs that allow for sharing knowledge between services.
- Spans over 26 services, 16 domains, resulting in a 16k dialogue dataset
- They use crowdsourcing for paraphrasing

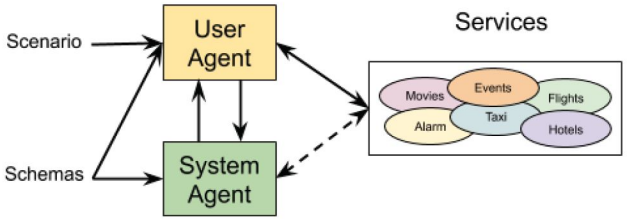



Figure 2: The overall architecture of the dialogue simulation framework for generating dialogue outlines.

# Generation Method - Natural Language Around Input

- Access to intents, slots and slot values that are plugged-in **generated** natural language utterances
- No language templates
- Natural Language Generation -> generations are more versatile
- Requires less human-involvement

NUS / NeuralWOZ (Kreyssig et al., 2018, Kim et al., 2021)

- Eliminates **hand-crafted templates,** but still uses API calls
- Corpus-driven
- (NUS) Dynamic goal generation: the system can dynamically change the goal, assuming the user would want to shift their goal mid-conversation

# Generation Method - Natural Language Conditioned on Input

- Access to intents, slots and slot values that are used as **input** to **generate** natural language utterances

HUS (Gür et al., 2021)

- Same family as ABUS and NUS
- Employs a multifaceted encoding scheme: it encodes different features in different vector representations (the user goal, the current dialogue turn, the dialogue history)

VHUS (Gür et al., 2021)

- HUS but created more human-like generations
- How? HUS is deterministic, while VHUS introduces variability through variational inference
- VHUS models the dialog latent space without affecting the slots and values extracted from a KB

# Generation Method - Natural Language Conditioned on Input

TUS (Lin et al., 2021)

- Similarly to VHUS, TUS maps different inputs to different representations in the feature space
- BUT it is domain-agnostic
- By adding a *domain and slot index feature* representation that can be changed

JOUST (Tseng et al., 2021)

- Simulator: two pre-trained agents, fine-tuned using RL
- Novelty is added by fine-tuning on multi-domain dialogues

# Generation Method - Natural Language Conditioned on Input

JOUST (Tseng et al., 2021)

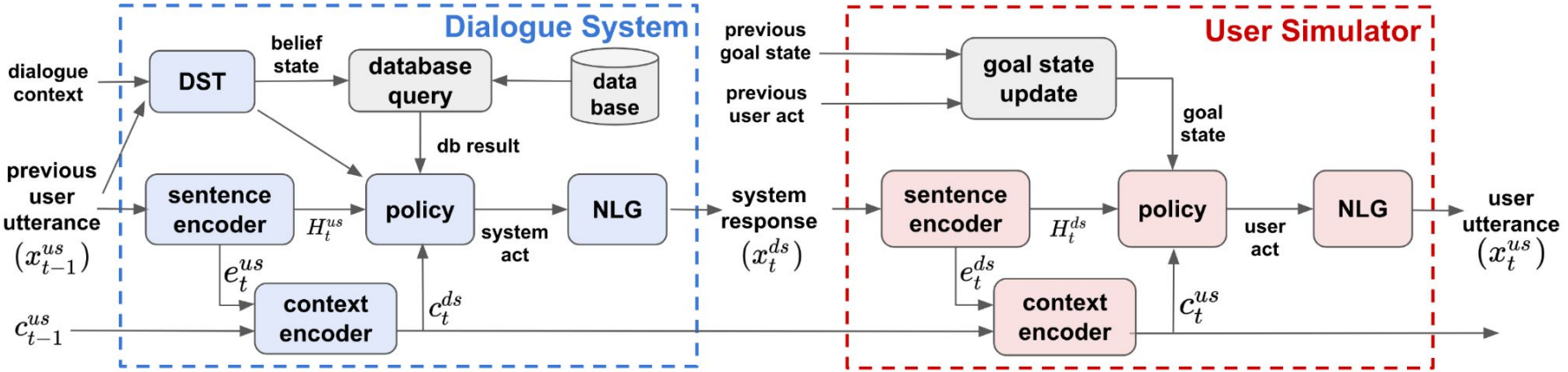



Figure 1: Overall architecture of the proposed framework, where the dialogue system (DS) and user simulator (US) discourse with each other. $t$ denotes dialogue turn index. The context encoder is shared between the two agents.

# Generation Method - Natural Language Conditioned on Input

INA (Ahmad et al., 2023)

- Simulator: two pre-trained agents, fine-tuned using RL
- Negotiation in a win-win manner, meaning each party must understand the other's needs and goal is mutual satisfaction
- Generates a Negotiation Dialogue dataset using negotiation-specific intents
- Novelty: adds negotiation-inteints such as *Negotiate-Price-Decrease*, *Add-X*, ..
- Data correction with human-in-the-loop for quality check
- Uses GPT-J for generation
- Challenge: negotiation strategies are highly context-dependent, so it adds a layer of complexity compared to the previous approaches

# Generation Method - Variational Inference

VHDA (Yoo et al., 2020)

- **NO predefined knowledge**
- Input: human-generated dialogues
- Models latent variables over all dialogue aspects similar to VHUS, and TUS, but this time also for learning **intents**, **slots** and **slot values**
- Allows for the model to generate attributes beyond the training data
- However, there is no guarantee these generations are valid (we will discuss this more in part 3 of this section)

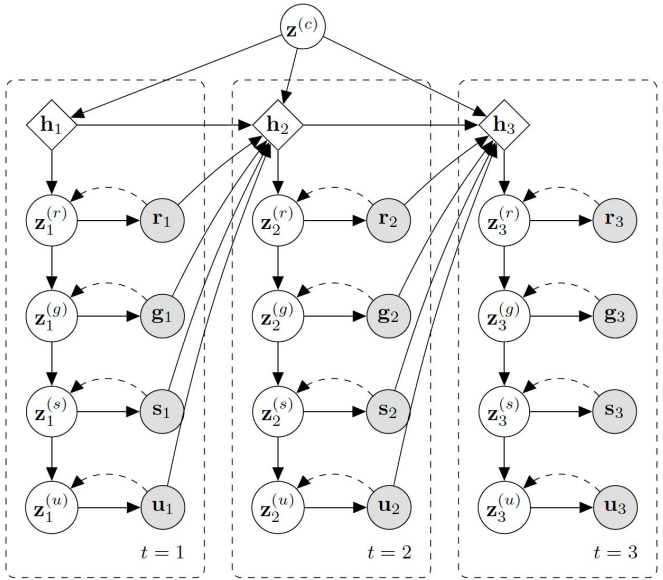



Figure 1: Graphical representation of VHDA. Solid and dashed arrows represent generation and recognition respectively.

# Generation Method - Few-shot learning

Simulated-Chat (Mohapatra et al., 2021)

- **NO predefined knowledge**
- Few-shot learning: the ability of a model to generalize when provided a very small dataset for **training** or **fine-tuning**
- Input: set of instruction based on which an LLM can generate dialogues
- Uses GPT-2 and Longformer
- First receives human-generated dialogues, then self-generated simulated dialogues

# Generation Method - In-context learning

ICL-US (Terragni et al., 2023)

- **NO predefined knowledge**
- In-context learning: the ability of a model to generalize when provided a very few examples in the input prompt without explicitly **training** or **fine-tuning**
- Input: set of instruction based on which an LLM can generate dialogues and example dialogues

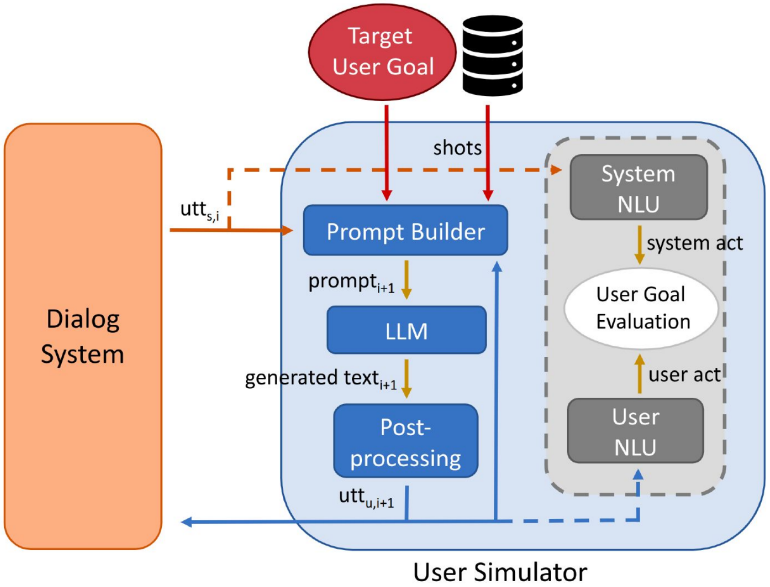



Figure 1: System and user simulator architecture sketch.

# Generation Method - In-context learning

Dialogic (Li et al., 2023)

- **NO predefined knowledge**
- Input: set of instruction based on which an LLM can generate dialogues and example dialogues
- In-context learning: the ability of a model to generalize when provided a very few examples in the input prompt without explicitly **training** or **fine-tuning**

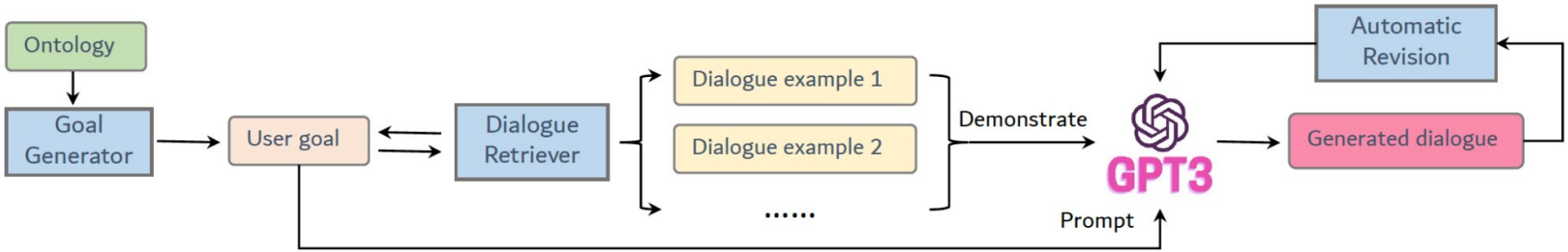
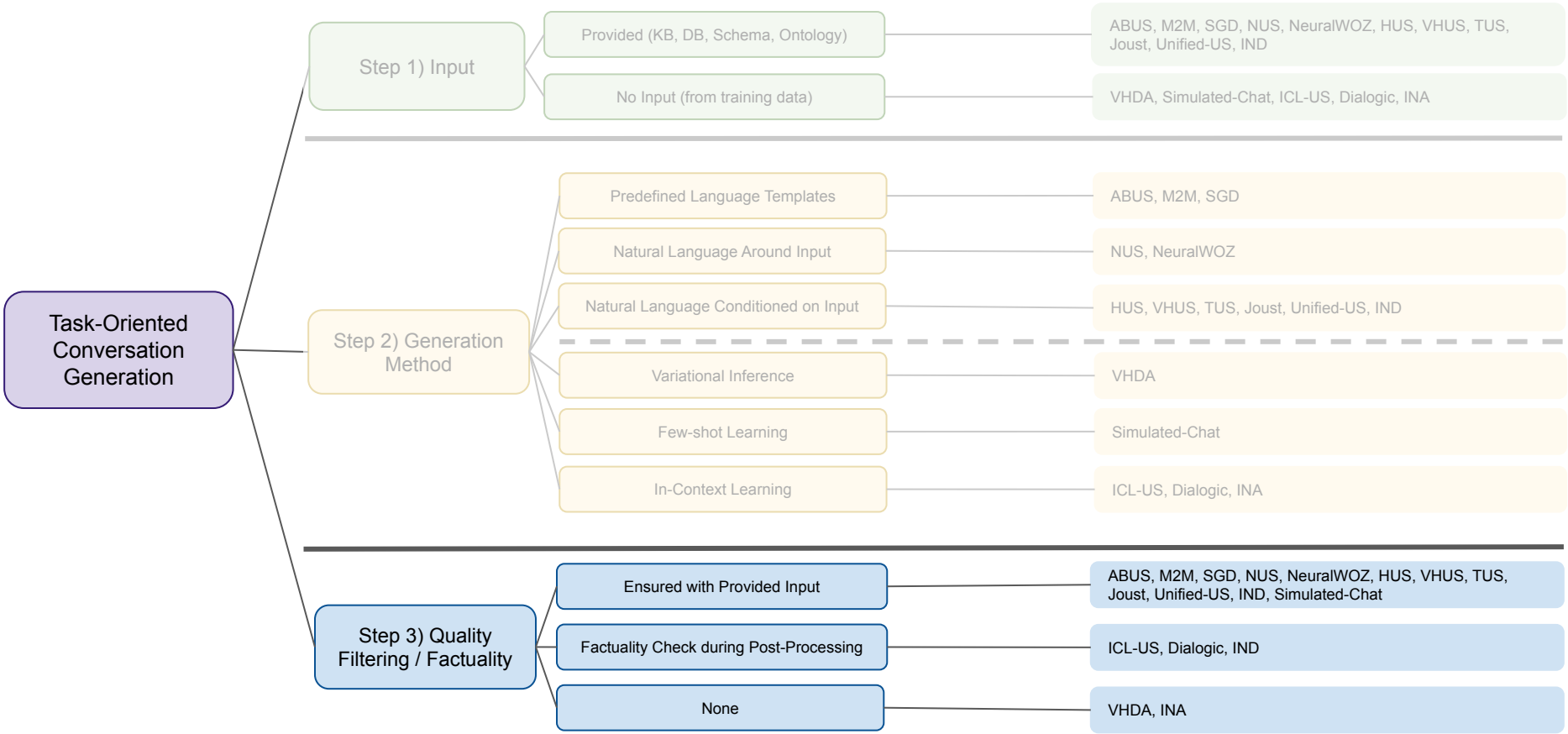



Figure 2: Overview of the proposed method.


Task-Oriented Conversation Generation
Step 1) Input
Provided (KB, DB, Schema, Ontology)
ABUS, M2M, SGD, NUS, NeuralWOZ, HUS, VHUS, TUS, Joust, Unified-US, IND
No Input (from training data)
VHDA, Simulated-Chat, ICL-US, Dialogic, INA
Step 2) Generation Method
Predefined Language Templates
ABUS, M2M, SGD
Natural Language Around Input
NUS, NeuralWOZ
Natural Language Conditioned on Input
HUS, VHUS, TUS, Joust, Unified-US, IND
Variational Inference
VHDA
Few-shot Learning
Simulated-Chat
In-Context Learning
ICL-US, Dialogic, INA
Step 3) Quality Filtering / Factuality
Ensured with Provided Input
ABUS, M2M, SGD, NUS, NeuralWOZ, HUS, VHUS, TUS, Joust, Unified-US, IND, Simulated-Chat
Factuality Check during Post-Processing
ICL-US, Dialogic, IND
None
VHDA, INA

# Component 3: Quality Filtering

**Ensured with Provided Input**

- When extracting slots and slot values from Ontology, Schema, KG and DB, factuality is granted
- ABUS, M2M, SGD, NUS, NeuralWOZ, HUS, VHUS, Joust, Unified-US

**None**

- Although uncommon, approaches such as VHDA ensure semantic logic of dialogue turn but does not constrict, or edit generations given lack of factuality or lack of plausible interactions.

# Component 3: Quality Filtering

## Factuality Check during post-processing

- Methods that discover slots and slot values in the latent space
- Dialogic has a step called automatic revision, where it corrects for potential errors by comparing GPT-3 generated belief states with the current utterance; The errors can be either due to de-generation or over-generation
- ICL-US adds an evaluation step by comparing all dialogue act extracted from the generated system and User NLU competent at each turn

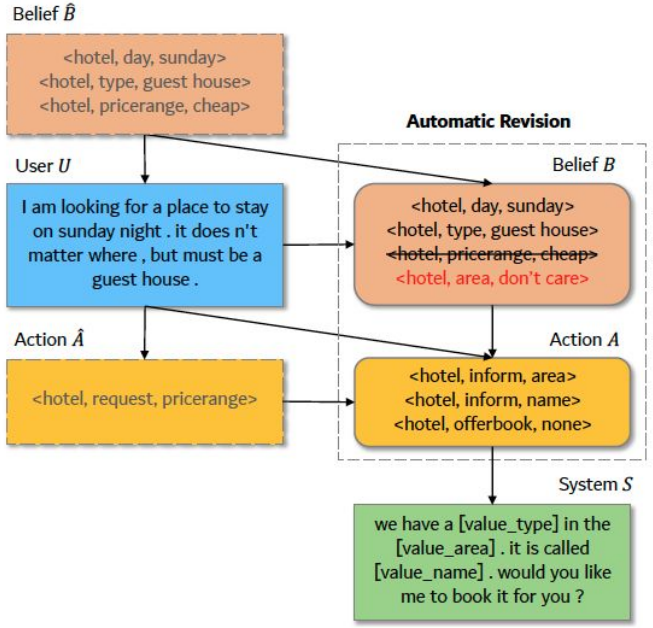



Figure 5: Illustration of the controllable generation process of a dialogue turn. An example of the generation process of a complete dialogue is shown in Appendix C.1 as Table 9.