Metaphorical User Simulators for Evaluating Task-oriented Dialogue Systems

WEIWEI SUN, Shandong University, China
SHUYU GUO, Shandong University, China
SHUO ZHANG, Bloomberg, United Kingdom
PENGJIE REN, Shandong University, China
ZHUMIN CHEN, Shandong University, China
MAARTEN DE RIJKE, University of Amsterdam, The Netherlands
ZHAOCHUN REN*, Shandong University, China

Task-oriented dialogue systems (TDSs) are assessed mainly in an offline setting or through human evaluation. The evaluation is often limited to single-turn or is very time-intensive. As an alternative, user simulators that mimic user behavior allow us to consider a broad set of user goals to generate human-like conversations for simulated evaluation. Employing existing user simulators to evaluate TDSs is challenging as user simulators are primarily designed to optimize dialogue policies for TDSs and have limited evaluation capabilities. Moreover, the evaluation of user simulators is an open challenge.

In this work, we propose a metaphorical user simulator for end-to-end TDS evaluation, where we define a simulator to be metaphorical if it simulates user's analogical thinking in interactions with systems. We also propose a tester-based evaluation framework to generate variants, i.e., dialogue systems with different capabilities. Our user simulator constructs a metaphorical user model that assists the simulator in reasoning by referring to prior knowledge when encountering new items. We estimate the quality of simulators by checking the simulated interactions between simulators and variants. Our experiments are conducted using three TDS datasets. The proposed user simulator demonstrates better consistency with manual evaluation than an agenda-based simulator and a seq2seq model on three datasets; our tester framework demonstrates efficiency and has been tested on multiple tasks, such as conversational recommendation and e-commerce dialogues.

CCS Concepts: • **Information systems** → *Users and interactive retrieval*; *Evaluation of retrieval results*; • **Computing methodologies** → *Natural language processing*.

Additional Key Words and Phrases: Task-oriented dialogue; Evaluation; User simulation

ACM Reference Format:
Weiwei Sun, Shuyu Guo, Shuo Zhang, Pengjie Ren, Zhumin Chen, Maarten de Rijke, and Zhaochun Ren. 2023. Metaphorical User Simulators for Evaluating Task-oriented Dialogue Systems. *ACM Transactions on Information Systems* 1, 1, Article 1 (May 2023), 28 pages.

*Corresponding author.

Authors' addresses: Weiwei Sun, Shandong University, Qingdao, China, sunnweiwei@gmail.com; Shuyu Guo, Shandong University, Qingdao, China, guoshuyu225@gmail.com; Shuo Zhang, Bloomberg, London, United Kingdom, szhang611@bloomberg.com; Pengjie Ren, Shandong University, Qingdao, China, renpengjie@sdu.edu.cn; Zhumin Chen, Shandong University, Qingdao, China, chenzhumin@sdu.edu.cn; Maarten de Rijke, University of Amsterdam, Amsterdam, The Netherlands, m.derijke@uva.nl; Zhaochun Ren, Shandong University, Qingdao, China, zhaochun.ren@sdu.edu.cn.

1046-8188/2023/5-ART1 $15.00
https://doi.org/

1 INTRODUCTION

Task-oriented dialogue systems (TDSs) assist users in solving a specific task during a conversation [62]. They are being considered in a growing number of information retrieval scenarios, such as conversational information seeking, conversational Q&A, and conversational recommendation [14, 48, 49]. In a TDS, dialogues follow a clearly defined structure that builds on domain knowledge relevant for the task(s) at hand. The evaluation of TDSs is a crucial part of the development process. Recent studies on evaluating TDSs are either through offline evaluation or human evaluation [23, 26]. Offline evaluation evaluates a dialogue system based on test sets, whereas human evaluation reflects the overall performance of the agent through in-field experiments [4, 26] or crowd-sourcing [23]. However, offline evaluation is often limited to single turn assessments, while human evaluation is intrusive, time-intensive, and does not scale [10].

User simulations. User simulation can potentially mitigate the above issues via simulated interactions, and may thus be a viable choice for large-scale automatic evaluation for TDS [10]. The structured nature of task-oriented dialogues allows us to build user simulators as we can exhaustively enumerate user goals to generate human-like conversations for simulated evaluation [68]. Recent studies have employed user simulation for evaluating conversational information access tasks, including complex information seeking [25], conversational item recommendations [63], and multi-step task completion [56]. User simulators have also been successfully applied as a reinforcement learning environment for dialogue policy optimization [53]. However, user simulators as evaluation methods for TDSs are still under-explored [3].

We identify the following challenges when employing user simulators to evaluate TDSs: (i) First, employing existing simulators for evaluating TDSs suffers from limited realism and evaluation capability [3]. E.g., commonly used agenda-based simulators [52] rely on human-curated rules that are domain-specific; thus, their responses are far from human-like [63]. End-to-end simulators [40, 58] have been proposed to improve the domain generalizability of simulators. Still, their evaluation capabilities are limited at the semantic level (e.g., if the system provides a novel entity) [58]. (ii) Besides, there is a lack of automatic methods to assess user simulators' realism and evaluation capabilities. Most existing methods rely on manual evaluation, which is costly and hard to reproduce. Thus, the evaluation of user simulators is still an open challenge [45].

Our proposal. In the face of the challenges identified above, we aim to provide a solution for improving the realism and transferability of user simulators and thus serve as an evaluation method for TDSs. Our solution includes a simulator based on metaphorical user modeling for end-to-end TDS evaluation and an automated evaluation method based on testers.

Specifically, we propose a metaphorical user simulator (MetaSim) that improves the realism of the conversation strategies by simulating the user's analogical thinking. MetaSim is metaphorical because it uses historical strategies as metaphors for an ongoing dialogue: it assists the simulator in the reasoning process by referring to historical dialogue strategies when encountering new items. MetaSim comprises: (i) a preference module that initiates diverse user preferences and updates them during conversations; (ii) a natural language understanding (NLU) module that tracks the dialogue state; (iii) a metaphor module that retrieves similar dialogue strategies related to the current dialogue state; (iv) a policy module that simulates user satisfaction and predicts the user action according to context and retrieved strategies; and (v) a natural language generation (NLG) module that generates a response in natural language.

To evaluate user simulators, we construct a tester-based comparative evaluation framework inspired by the paradigm of evaluating interactive IR systems [25] and conversational recommenders [63]. A tester defines how to make variants for a dialogue system with distinguishable performance and how to evaluate user simulators by interacting with them. We leverage a base

dialogue system based on SOLOIST [43]. Its variants, defined by testers, are constructed by configuring parameters of the context history, retrieval or recommender methods, and dialogue domains. We evaluate the simulators by checking if they can rank the system variants in the same way as humans and respond to new items in a human-like way. A well-built tester can repeatedly evaluate the evaluation capabilities of simulators without the need for manual testing.

Experiments. We conduct experiments on three benchmark datasets, MultiWOZ [13], ReDial [30], and JDDC [7]. We implement MetaSim based on T5 [47], a pre-trained transformer model, and adopt a unified data format that allows it to be generalized to multiple tasks. We optimize each module on the dataset and connect them during testing. We evaluate MetaSim by checking the naturalness of the generated responses by user simulators, the quality of interactions, and consistency between user simulators and humans when comparing the variants of TDSs. Our experiments show that MetaSim consistently outperforms previous comparable simulators in terms of evaluation consistency and human-likeness.

Our experiments demonstrate that (i) metaphorical user simulators (MetaSim) can generate human-like dialogues and achieve evaluation results that are consistent with human expectations; (ii) a tester-based framework is a valid tool for automatic assessment of simulator evaluation capabilities; and (iii) our method generalizes well across domains.

Contributions. The contributions of this paper are:

- We introduce a simulation approach for evaluating task-oriented dialogue systems. In particular, we propose a metaphorical user simulator (MetaSim) that leverages the dialogue records to improve the dialogue generation ability.
- We introduce a tester-based framework for evaluating user simulators;
- We validate the proposed method through automatic and human experiments on three datasets; and
- We release the dataset and code of user simulators at http://github.com/sunnweiwei/MetaSim and the code of tester-based framework at https://github.com/Superbooming/simtester.

2 RELATED WORK

2.1 Task-oriented dialogue systems

Task-oriented dialogue systems (TDSs) aim to assist users in completing tasks through conversations [13]. TDSs are developed either via module-based or end-to-end approaches, differing in whether they manage the sub-steps of dialogue generation with multiple independent modules or not. [20]. Research on the former boils down to advancing the components of TDSs, including natural language understanding (NLU), dialogue state tracking (DST), dialogue policy learning (DPL), and response generation (RG) [36, 37, 51]. The latter generates responses relying on end-to-end neural techniques in dialogue generation, such as the pre-trained language models, which aim to reduce the efforts in component-specific designs [21, 28, 59]. For example, Lei et al. [28] propose the *belief spans* to track dialogue believes for TDSs, which allows an end-to-end seq2seq modeling. Liang et al. [31] introduce an end-to-end trainable framework for TDSs that aggregates supervision from various intermediate dialog system modules.

Recently, pre-trained language models have boosted the end-to-end dialogue modeling, e.g., GPT-2 for all sub-tasks involved in the task-oriented dialogue system [18, 61]. Pre-trained models based on heterogeneous dialogue corpus [17, 43] or leveraging multi-task learning [55] have demonstrated promising performance in scenarios where in-domain annotated data is scarce [32, 34]. TDSs have also been considered in an increasing number of interactive information retrieval scenarios, such as conversational recommendation [14, 30], conversational information seeking [49],

and conversational question answering [48]. For example, question-based user preference elicitation [14] has been proposed to ask about items [71] or attributes [27], and multi-turn conversational recommendation strategies have been explored by [57, 65]. Li et al. [30] construct REDIAL, which is an annotated dataset of conversational recommendation dialogues about movies. Ren et al. [49] propose a module-based framework for conversational information seeking.

2.2 Evaluation of dialogue systems

The typical evaluation methods of dialogue systems include automatic evaluation and human evaluation [10]. Commonly used automatic evaluation methods include component-level evaluation and task-level evaluation. Component-level evaluation is concerned with the assessment of each component. For example, the DST module is conventionally evaluated by joint goal accuracy (i.e., the average accuracy of predicting all slot assignments for a turn correctly), and the accuracy of action prediction is used to evaluate the dialogue policy learning module [13]. To evaluate the item recommendation capability, metrics like recall and NDCG are used to measure the accuracy of recommended items per turn [50]. As for the evaluation of response evaluation, BLEU [41] is employed to measure the word overlap between generated response and ground truth, which draws inspiration from the evaluation of machine translation; Perplexity and Distinct are also used to evaluate the fluency and diversity of text generation, respectively [10].

Task-level evaluation evaluates the overall performance of the dialogue system; commonly used metrics include task success rate, inform success rate, and average turn. Specifically, task success rate validates whether the final recommended items meet the user's needs; inform success rate validates if the system answers the user's question; and average turn evaluates the efficiency of dialogue systems [13, 14, 42].

The above methods based on a static test set are limited to a single turn and do not inform us about the overall usefulness of the system in interaction [63]. Human evaluation can address this drawback [23], and popular human evaluation metrics include Satisfaction, Fluency, Coherence, Task success, Engagingness, etc. However, human evaluation is intrusive, time-intensive, and does not scale [54]. Employing simulation-based evaluation can tackle the above issues and be a viable choice for large-scale automatic evaluation [10].

2.3 User simulation

User simulators are designed to simulate the user's behavior, which can be used either as an environment to train a reinforcement learning-based system or to evaluate a functioning system to find weaknesses (or assess) dialogue quality and potentially replace human evaluation [10]. For the sake of evaluating spoken dialogue systems, Eckert et al. [11] propose the first statistical user simulator. Using a Markov model [15], Cuayáhuitl et al. [8] propose a hidden Markov model for the same purpose. In later work, the agenda-based user simulator [52] has been widely accepted as it elegantly represents the user state as a stack of necessary user actions.

User simulation is not foreign to information retrieval evaluation; its importance has been confirmed in the Sim4IR workshop at SIGIR 2021 [3]. Different methodologies have been proposed for building simulators, e.g., employing a Bayesian procedure [6], cognitive state for interactive information retrieval [35], and for different tasks like search sessions [66], online news recommendation [5], conversational recommender systems via an agenda-based user simulator [63], biases present in the logged data [19]. In addition, Shi et al. [53] investigate the design of user simulators as an reinforcement learning environment. Tseng et al. [58] introduce a reinforcement learning approach based on end-to-end modeling.

Existing simulators mechanically inform the system of the slot, which limits the realism and the evaluation capability [56]. Thus, the main challenge of employing simulation is to build a realistic

user simulator that can mimic natural user behavior to a realistic extent, i.e., "human-likeness." Humans have complicated mental activities and display specific behaviors when interacting with machines [12]. Existing studies have shown that improving human-likeness of either simulators [56, 63, 64] or dialogue systems [12] calls for detailed modeling of *human behavior* in dialogues. Studies show that users build "mental models" in interacting with a machine, and that such mental models are formed through analogical thinking [2, 22]. We aim to prototype an initial mental model in this work, namely the metaphorical user simulator. A metaphorical user simulator has a metaphor model that refers to historical dialogue strategies in the reasoning process. This metaphor model resembles semi-parametric methods in NLP [9], a class of methods that assist a series of knowledge-intensive language tasks (e.g., question answering, fact checking [44]) by retrieving external knowledge. Compared with previous semi-parametric methods in NLP, our proposed metaphorical model further improves the model's capability in user action prediction on the task-oriented dialogue and considers the contextual information about the retrieved content (i.e., the multi-turn dialogue context).

2.4 Evaluation of user simulators

The evaluation of user simulators is an open challenge [45]. Existing evaluation methods of user simulators include the following approaches: (i) Simulators are treated as dialogue systems and then evaluated similarly through text generation metrics (e.g., BLEU, PPL), component-level metrics (e.g., action prediction accuracy), and task metrics (e.g., task success rate) [38]. (ii) Alternatively, assessments are performed based on the realism of simulated dialogues [63]. There, descriptive statistics (e.g., the number of turns, words, slots, actions) of machine-machine dialogues data are compared to data from human-human dialogues, or use humans are asked to perform a Turing test. (iii) In addition, human evaluation is a critical component that assesses the consistency between humans and simulators. Typical metrics include scoring-based methods (e.g., SM-Turn, SM-Dialog) and comparison-based methods (e.g., PW-Turn, PW-Dialog) [54]. (iv) Lastly, some work evaluates simulators by checking the performance of dialogue systems trained with the simulators by reinforcement learning [53, 58].

To directly assess the evaluation capabilities of a simulator, recently, a tester-based evaluation framework has been used in interactive search [25], which has shown good consistency with human expectations while costing less and being more stable than human evaluation. Our work differs from [25] in the following aspects. Testers in interactive search in [25] are constructed by configuring the retrieval methods, changing information needs, or using different document collections. Compared to these testers in interactive search, testers in TDSs require a substantial change in design as dialogue systems involve additional components, such as NLU, NLG, state tracking, etc. We focus more on these characteristics of TDSs, e.g., tracking user intent in multi-turn conversations and ablations on an embedding-based recommendation module when designing testers. Specifically, we control the systems' ability to track a multi-turn conversation by limiting the number of available turns to system; the recommender is ablated by deleting features from item embeddings. In addition, we establish a framework that can be used for multiple conversational tasks, e.g., movie recommendation, restaurant reservations. The proposed framework gets rid of human effort in dialogue evaluation and supports multiple conversational tasks, thus can facilitate reproduction and expansion of the evaluation of task-oriented dialogues.

3 PROBLEM FORMULATION

Most evaluation methods for TDSs follow the PARADISE [60] framework and estimate user satisfaction based on dialogue cost and task success via automatic, simulated, or human evaluation. Simulation-based evaluation methods efficiently evaluate the system with interactions, but their

current capacity is too limited to compensate for human evaluation [3]. Building a human-like user simulator does not mean that we are after a perfect mirror of human behavior, let alone the replacement of humans. The simulator should be good enough to be an assessor that correlates well with human assessment in certain aspects and potentially reduces the reliance on human effort in the loop of evaluation.

Our goal is to build an evaluation framework for comparing task-oriented dialogue systems via user simulation. This framework, first, should be capable of configuring *testers* for a given base dialogue system, i.e., a *base model.*

Definition 1 (Base Model): *The base model is a standard dialogue system that is configurable, thus leading to potentially different capabilities.*

Definition 2 (Tester): *A tester, $\mathbb{T}$, is an instance of a base system, which defines how to configure a base system, e.g., via the choices of the NLG model.*

Definition 3 (Variant): *A system* variant*, S_i, is an instance of $\mathbb{T}$.*

Given a tester system $\mathbb{T}$, any pair of its variants $S_1^{\mathbb{T}}$ and $S_2^{\mathbb{T}}$ should have a sensible difference in components, training data, and/or methods, and thus display distinguishable system performance.

Second, the user simulators can be configured to evaluate dialogue systems on certain aspects by interacting with the systems. Suppose S_1 has a better ability in recommending new items than S_2, and the simulator is expected to measure this difference. The measures by the simulator should be consistent and stable with human assessments, and can be reproduced under this framework.

Lastly, this framework enables us to explore more human-like aspects to see how realistic simulators can be. In this study, we investigate the possibility of building a simulator that responds to new knowledge in a human-like fashion by exploring using a human metaphor. Here, *metaphor* is the module where the user draws on relevant conversational strategies and knowledge in the interaction [24] and is found to be the primary element of the construction of a *mental model* [22]. The simulation framework will be detailed in Sect. 4.

4 FRAMEWORK

We introduce a framework aimed at improving the realism and transferability of user simulators for end-to-end TDS evaluation. This framework includes a metaphorical user simulator (MetaSim) for end-to-end TDS evaluation and an automated evaluation method based on testers. In this section, we first introduce MetaSim (Sect. 4.1), which improves the evaluation capability of the simulator by simulating user's analogical thinking. Then, we introduce the tester-based evaluation framework for evaluating the evaluation capability of the simulators (Sect. 4.2).

4.1 Metaphorical user simulators

We propose a metaphorical user simulator, **MetaSim**, that constructs a metaphorical user model to assist dialogue simulation by referring to prior knowledge. The metaphorical user model simulates the user's analogical thinking in interacting with machines. As shown in Figure 1, the simulator consists of five modules: a preference module, NLU, a metaphor module, a policy module, and an NLG module.

We introduce the modules one-by-one using the following notation: $\{\mathcal{P}, \mathcal{U}, \mathcal{S}, \mathcal{M}, a_{t+1}, u_{t+1}\}$, where $\mathcal{P}$ denotes user preferences that define the user requirements or goal of the conversation; $\mathcal{U} = \{u_1, r_1, \ldots, u_t, r_t\}$ is the dialogue context, in which u_i denotes a user utterance at turn i, and r_i denote the system's response at turn t; $\mathcal{S} = \{a_1, s_1, \ldots, a_t, s_t\}$ is the dialogue state while a_i is the user's action at turn i, and s_i is the dialogue belief state after system's response at turn i;

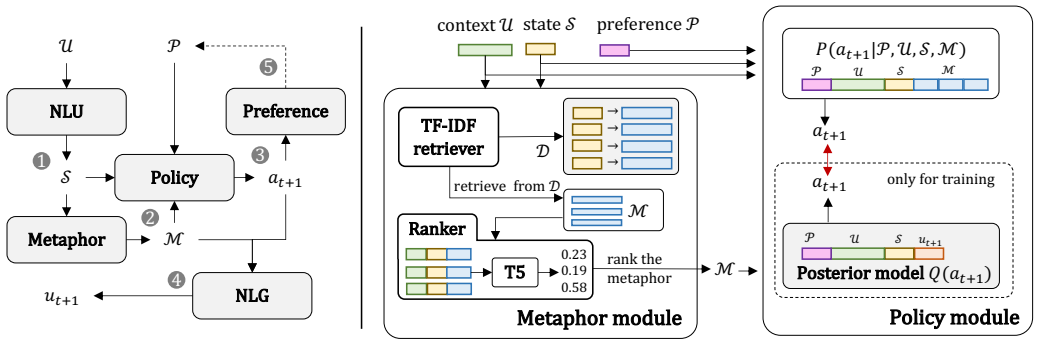

Fig. 1. Architecture of the metaphorical user simulator (MetaSim). The left side shows the five modules of MetaSim (i.e., Preference, NLU, Metaphor, Policy, and NLG) and how they are connected. The right side details the Metaphor module and Policy module, where the Metaphor module retrieves $\mathcal{M}$ from $\mathcal{D}$, and the Policy module decides the next user action.

$\mathcal{M} = \{m_1, \ldots, m_k\}$ is the metaphor strategies and each m_i in $\mathcal{M}$ is a utterance; a_{t+1} is the next user action concatenated with user satisfaction, and u_{t+1} is the next user utterance. Besides, the metaphorical user simulator relies on a database $\mathcal{D}$ that contains dialogue records from which the model retrieves a metaphor $\mathcal{M}$, and we detail it in Sect. 4.1.3.

4.1.1 Preference module. The user preferences $\mathcal{P}$ define the user requirements or the goal of the conversation, and the preference module tells how $\mathcal{P}$ is initialized and updated during the conversations. We represent the user preferences $\mathcal{P}$ as

$$\{(\text{slot}_1, \text{value}_1), \ldots, (\text{slot}_n, \text{value}_n)\},$$

where slot_i denotes an attribute of user requirements and value_i is its value. The initialization of $\mathcal{P}$ fills the initial user preferences before the conversation. For example, the preference for hotel reservation "*want to book a cheap hotel where parking is available*" is initialized as: {(hotel_parking, "*yes*"), (hotel_price, "*cheap*")}. It can assist the model in answering the system's elicitation question at the beginning of the conversation, e.g., answer "What price point would you like for the hotel?" as "cheap". The update of $\mathcal{P}$ tracks the user preference changes during the conversation. For instance, the system may ask a question that is out of the scope of the initialized preference, such as "do you need WIFI." The preference module will add the newly generated user preferences, i.e., (WIFI_needed, "*yes*"), to the initialized preferences when the answer is "yes". As a result, the model can give a consistent answer if the system asks a relevant question in future conversation turns.

Initialization of $\mathcal{P}$. The $\mathcal{P}$ is initialized based on a *item database*, namely *DB*. As shown in Figure 2, the *DB* consists of multiple tables, each table contains items of a specific domain (e.g., restaurant, hotel) with their attributes (e.g., name, area, food, etc.). The initialization of $\mathcal{P}$ can be divided to three steps:

(1) **Domain sampling:** In task-oriented dialogues, the user's preference can span multiple domains [13]. For example, the user may want to find an attraction and then take a taxi there. We refer to a set of domain covered by a user's preferences as a *domain combination*, where multiple domains are typically logically related and will constitute a joint goal. To sample the domain of the user's preference, we calculate the probability of each domain combination (e.g., {hotel} and {hotel, restaurant} denotes two different domain combinations) in the training

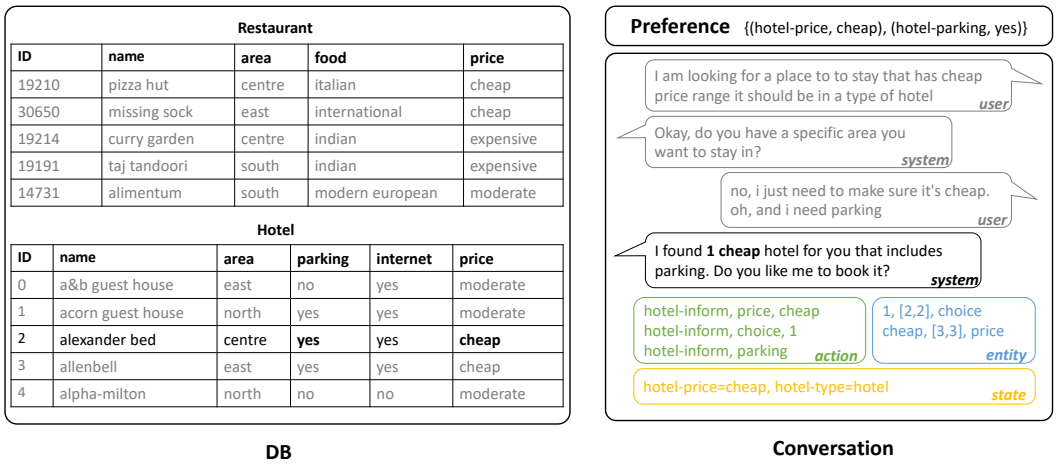

Fig. 2. An example of an item database and conversation annotation in MultiWOZ 2.1.

set. Then, we sample a domain combination based on the distribution of training set. In the example, the sampled domain combination is {hotel}.

(2) **Items sampling:** For each selected domain from *Domain sampling* step, we sample one item from the table following a uniform probability distribution. In the example, the ID of sampled item is 2.

(3) **Attributes sampling:** We calculate the attribute number distribution using the training set, which defined as the number of slots used in the preference. Then, we sample k attributes for each item follow a uniform probability distribution, and k is determined follows the attribute number distribution. In the example, $k = 2$ attributes are selected to be used in preference, that is parking="*yes*" and price="*cheap*".

Update of $\mathcal{P}$. During the conversation, the preference module updates the preference $\mathcal{P}$ according to the following rules:

(1) If the user "informs" the system about a particular slot, then tag its value as "Informed" in $\mathcal{P}$. For example, for $\mathcal{P} = \{(\text{hotel\_parking}, \text{"}yes\text{"}), (\text{hotel\_price}, \text{"}cheap\text{"})\}$, if the user informs the system that the price is *cheap*, then $\mathcal{P}$ is updated to $\mathcal{P} = \{(\text{hotel\_parking}, \text{"}yes\text{"}), (\text{hotel\_price}, \text{"}cheap \mid Informed\text{"})\}$;

(2) If the system "recommends" an item, add it to $\mathcal{P}$. For example, if a newly recommended movie does not appear in $\mathcal{P}$, this module takes the movie as a slot and the user's favor towards it as value and adds it into $\mathcal{P}$.

4.1.2 Natural language understanding. The NLU module keeps track of the states (i.e., the slots have been informed) of the dialogue. We use *actions* to represent the intent of an utterance (of either system or user) in dialogue. We define the format of an action as follows:

$$\{(\text{act}_1, \text{slot}_1, \text{value}_1), \ldots, (\text{act}_n, \text{slot}_n, \text{value}_n)\},$$

where each item $(\text{act}_i, \text{slot}_i, \text{value}_i)$ includes: (i) act_i, the name of action (e.g., *inform*, *bye*); (ii) slot_i, a slot, and (iii) value_i, its value. Note that some actions that do not have slots or values, such as *bye*; in such cases the slots and values are set to NULL. Given user preferences $\mathcal{P}$ and dialogue context $\mathcal{U}$, the NLU module predicts the action s_t of the last system response r_t. Then, we append s_t to the historical actions, which are already predicted in previous turns. These steps form the dialogue state $\mathcal{S}$. Thus the dialogue state $\mathcal{S}$ records all the slots that have been informed from the

system. We optimize the NLU module using supervised learning and minimize the log-likelihood $\mathcal{L}_{NLU} = -\log(P(s_t \mid \mathcal{P}, \mathcal{U}))$.

4.1.3 Metaphor module. The metaphor module is responsible for retrieving similar dialogue records from a database (terms as $\mathcal{D}$). The retrieved dialogue records are used to support the policy module. To construct database $\mathcal{D}$, we index the dialogue state for each utterance in the training data and take the user utterance as the value. Formally, $\mathcal{D} = \{(d_1^k, d_1^v), \ldots, (d_n^k, d_n^v)\}$, where n denotes that $\mathcal{D}$ has n records, d_i^k is the index of the i-th record, and d_i^v is the value of the i-th record.

For any new dialogue, we retrieve k records $\mathcal{M} = \{m_1, \ldots, m_k\}$ (we call $\mathcal{M}$ a *metaphor*) from $\mathcal{D}$, where each m_i is a record retrieved from $\mathcal{D}$.

As illustrated in Figure 1, the retrieval follows a two-stage pipeline:

(1) **Candidate generation**: For any record (d_i^k, d_i^v), we retrieve k candidate records from $\mathcal{D}$ using TF-IDF$(\mathcal{S}, d_i^k)$, which is the TF-IDF score between the index of the record d_i^k and the current dialogue state $\mathcal{S}$. These candidates form $\mathcal{M}$.

(2) **Ranking**: We rank the candidates by a learnable *ranker* for improving the retrieval accuracy. The *ranker* is defined as $P(rel_i \mid m_i, \mathcal{U}, \mathcal{S})$. Specifically, for an utterance m_i in $\mathcal{M}$, we input the dialogue context $\mathcal{U}$ and dialogue state $\mathcal{S}$ to the *ranker*, and estimate the relevance score rel_i of them. In training, we set the relevance score of a ground-truth utterance to 1.0 and the remaining utterances to 0.0, and optimize the *ranker* by maximizing the likelihood of the relevance score:

$$\mathcal{L}_{Metaphor} = -\sum_{i=1}^{k} \log(P(rel_i \mid m_i, \mathcal{U}, \mathcal{S})).$$

In testing, we estimate the relevance scores for each utterance in $\mathcal{M}$ and rank them based on the scores.

4.1.4 Policy module. The policy module is responsible for predicting user satisfaction and the next user action. We define it as $P(a_{t+1} \mid \mathcal{P}, \mathcal{U}, \mathcal{S}, \mathcal{M})$. The inputs of policy module are the user preferences $\mathcal{P}$, dialogue context $\mathcal{U}$, dialogue state $\mathcal{S}$, and metaphor $\mathcal{M}$; the output is the user action a_{t+1}. To predict the user satisfaction and action jointly, we make the user satisfaction a particular slot in the original action following [56]. The annotation of user action and satisfaction is usually incomplete or noisy [16, 56], thus we introduce a *posterior network* $Q(a_{t+1} \mid \mathcal{P}, \mathcal{U}, \mathcal{S}, r_{t+1})$ to facilities the training of the policy module (as shown in Figure 1). The *posterior network* utilizes the ground-truth user utterance; thus, its predictions are more accurate than the policy module. We first optimize the *posterior network* on partially annotated data by a negative log-likelihood (NLL) objective:

$$\mathcal{L}^{Q}_{Policy} = -\log(Q(a_{t+1})),$$

and then optimize the policy module by learning from the *posterior network*. The objective is defined as the Kullback–Leibler divergence between the $Q(a_{t+1})$ and $P(a_{t+1} \mid \mathcal{P}, \mathcal{U}, \mathcal{S}, \mathcal{M})$:

$$\mathcal{L}^{P}_{Policy} = -Q(a_{t+1}) \log\left(\frac{Q(a_{t+1})}{P(a_{t+1} \mid \mathcal{P}, \mathcal{U}, \mathcal{S}, \mathcal{M})}\right).$$

4.1.5 Natural language generation. The NLG is responsible for generating natural language responses that incorporate system requests and individual preferences. We define it as: $P(u_{t+1} \mid \mathcal{P}, \mathcal{U}, \mathcal{M}, a_{t+1})$, where the NLG module inputs the user preferences $\mathcal{P}$, dialogue context $\mathcal{U}$, metaphor $\mathcal{M}$ and next action a_{t+1}, generate the next utterance u_{t+1} in a autoregressive fashion. We optimize it by supervised learning; the training objective of the NLG module is defined as:

$$\mathcal{L}_{NLG} = -\log(P(u_{t+1} \mid \mathcal{P}, \mathcal{U}, \mathcal{M}, a_{t+1})),$$

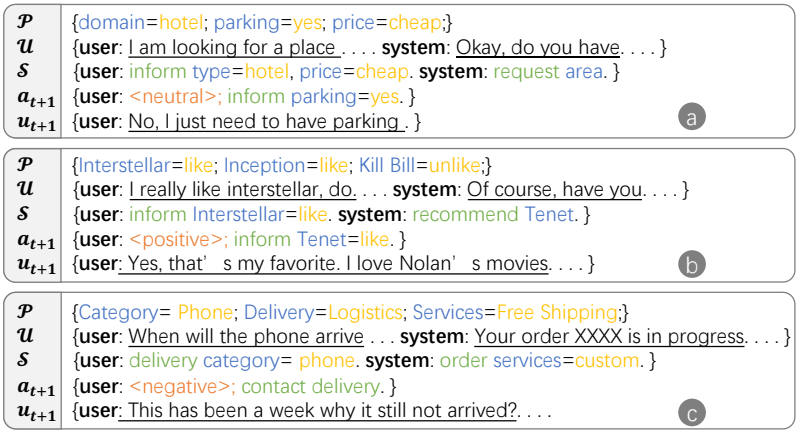

Fig. 3. Examples of processed sequences. We highlight attributes, values, action, satisfaction, utterance, and **speaker**. (a) is in the *hotel* domain from MultiWOZ, (b) is in the *movie* domain from ReDial, and (c) is in the *delivery* domain from JDDC.

where u_{t+1} is the next user utterance, $\mathcal{P}$ is user preference, a_{t+1} is the next user action, and $\mathcal{U}$ is the dialogue context.

4.1.6 Inference. The operation of our model is shown in Figure 1. We first draw a preference $\mathcal{P}$ from the preference module, that represents the initial user needs that lead the dialogue. Then the simulator interacts with the system according to $\mathcal{P}$. For each turn, with a dialogue history U, the model first predicts the system action s_t by the NLU module and composes the dialogue state $\mathcal{S}$. Then the metaphor module retrieves $\mathcal{M}$ from training set dialogue records and ranks the record sentences using the ranker. After that, the policy module predicts the user action and satisfaction based on the metaphor and the dialogue context. Then the NLG module converts the action to a natural language utterance u_{t+1} and responds to the system. Finally, the preference module updates $\mathcal{P}$ by adding new items and removing informed slots. The simulators generate "[END]" as an indicator to end the dialogue.

At the end of the dialogue, the simulator rates the system based on the simulated dialogue. Previous studies [56] show that human assessment of TDS are mainly driven by the success of the task and the quality of system response. Thus we define the calibrated satisfaction by the average of Success (that equals 1 if the system solves the user's problem and equals 0 otherwise) and Satisfaction (that averages the turn-level user satisfaction during the dialogue).

4.1.7 Implementation. We use T5 [47] to instantiate each module. T5 is a sequence-to-sequence model with an encoder-decoder structure. We process the inputs and outputs of each module as a sequence. See Figure 3 for some examples of our sequence processing.

For modules with multiple inputs, we concatenate them in order with a special token <SEP> and clip to the maximum input length of the model. Thus for each module, assuming the input text is $\mathcal{I}$ and the output text is $\mathcal{O} = \{o_1, \ldots, o_m\}$ with m tokens, the probability is defined as:

$$P(\mathcal{O} \mid \mathcal{I}) = \sum_{i=1}^{m} P(o_i \mid \mathcal{O}_{<i}, \mathcal{I}).$$

Taking the metaphor module as an example, we implement the ranker by T5, following techniques introduced by Nogueira et al. [39]. Recall that the metaphor module aims to retrieve similar dialogue

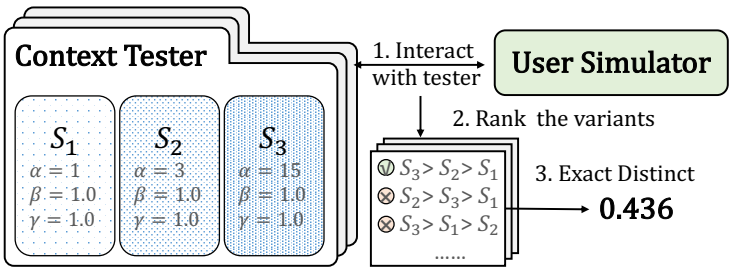

Fig. 4. In tester-based evaluation, the simulator interacts with system variants in the tester, ranks them, and calculates the Exact Distinct score (i.e., to accurately distinguish between the three systems).

records. Here, the learned model based on T5 first predicts the $O =$ "*yes*" for positive records and $O =$ "*no*" for negative records. The relevance score r_i is then defined as $r_i = P(\text{"}yes\text{"} \mid m_i, \mathcal{U}, \mathcal{S})$.

4.2 Tester-based evaluation framework

Existing simulated evaluation methods for a TDS typically rely on third-party systems [63]. This dependency suffers from the following issues: the limited choices of third-party systems and the difficulty of comparing them due to differences in training data and model architecture. Drawing inspiration from [25], we introduce a tester-based evaluation framework for evaluating both dialogue systems and user simulators, see Figure 4. The tester-based framework is constructed to evaluate the simulator automatically—the simulator talks to and evaluates the variants within a tester so that we can see whether a simulator can fairly compare the variants in a tester or not. The tester-based framework has the following benefits in evaluating simulators: (i) the constructed system variants are well-calibrated and thus comparable to each other; (ii) a tester can evaluate any specific capability of a system, leading to a more comprehensive evaluation; and (iii) a well-constructed tester does not require repeated human evaluation and can be reused to evaluate multiple simulators.

4.2.1 Evaluating user simulators by testers. Simulators are built for evaluating systems. Simulators also need to be assessed appropriately. Here, we discuss the latter case. The main idea to evaluate a simulator is to check whether the evaluation results by simulator upon variants are consistent with human expectations. For example, given a simulator US, and three variants S_1, S_2 and S_3, US interacts with S_1, S_2 and S_3 separately and scores them based on the interactions. The performance scores are on certain aspects like recommendation accuracy and domain understanding. A comprehensive assessment requires various testers and variants (see Sect. 4.2.2). We compare these scores against human expectations. A better consistency score indicates a better simulator.

We calculate the ExactDistinct (ED for short) score to measure the consistency between simulators' assessment scores and human expectations. ExactDistinct is defined as the proportion of examples that ranked the same as expectations by the human evaluator (i.e., $S_1 < S_2 < S_3$). An ED score of 0.5 means a rank accuracy of 50%.

4.2.2 Tester bootstrapping. We introduce the base system and three strategies to configure the testers. We leverage a base model based on SOLOIST [43], which employs a sequence-to-sequence model for dialog state tracking and response generation. The testers enable us to configure the base model and make variants based on the needs. Specifically, we can bootstrap testers by changing the context in the dialogue history, exploring different choices for components like the recommender module, leveraging domain-specific training data, and supplying various amounts of supervision

Table 1. Statistics of the training datasets. The top block gives the statistics of the original three datasets, and the bottom block lists the statistics of the USS dataset.

Dataset	MultiWOZ	ReDial	JDDC
Language	English	English	Chinese
Number of dialogues	9,438	10,006	47,389
Number of utterances	147,176	142,502	633,826
Avg number of turns	15.59	14.24	13.37
Number of Action	13	4	19
Number of Slots	15	6,925	23
USS user satisfaction annotation [56]			
Number of dialogues	1,000	1,000	3,300
Number of unsatisfied turns	737	740	4,940
Number of fair turns	11,141	9,623	45,005
Number of satisfied turns	675	1,524	4,572

for training, to name a few. Here, we introduce three strategies that we explore in this work; other options are left for future investigation:

- **Context tester.** Context refers to the dialogue history. The amount of dialogue history that is used for training has an impact on the model performance [14]. The Context tester, $\mathbb{T}_C$, makes variants by giving different context memory capabilities. We define a hyperparameter α indicating the number of sentences in the dialogue history. For example, $S_{\alpha=2}$ represents a model that keeps two sentences of dialogue history.
- **Recommender tester.** The Recommender tester, $\mathbb{T}_R$, consists of systems with different retrieval capabilities. This tester demonstrates how to make variants by changing a system component. The recommender retrieves from an item database using a query that contains a set of keywords. We then control the capacity of the recommender system by removing the keywords in the query. We define a hyperparameter β in the range 0 to 1. $S_{\beta=0.2}$ represents the model that keeps 0.2 of features in query.
- **Domain tester.** Domain tester, $\mathbb{T}_D$, makes variants by training modules that use different amounts of training data so that we can test the system's ability in a particular domain. We control it by removing training data from the training set. We define a hyperparameter γ to indicate the number of dialogues used for training a dialogue system; $S_{\gamma=0.8}$ represents the model that uses 80% of data.

5 EXPERIMENTAL SETUP

Next, we seek to assess the proposed simulators testers. To this end, we first describe three datasets used in our experiments (see Sect. 5.1) and the implementation details of the user simulators and testers (see Sect. 5.2).

5.1 Datasets

We conduct experiments on three datasets:

- **MultiWOZ** [13]: MultiWOZ is a multi-domain task-oriented dialogue dataset spanning seven domains (i.e., Attraction, Hotel, Hospital, Police, Restaurant, Train, and Taxi). It contains about 10K dialogues. Each dialogue can span up to three domains, and multiple domains might be mentioned in a single turn. MultiWOZ provides *action*, *entities* and *dialogue state* annotations for each utterance. Figure 2 shows an example. The *action* provides the action

name, and corresponding slot and value. The *entities* provides the entity span, start and end position in text, and the corresponding slot name. The *dialogue state* provides the (slot, value) pairs that the user has provided up to current turn. We use the MultiWOZ 2.1 [13] version, which fixes noisy state annotations across 40% of the dialogue turns in previous versions.

- **ReDial** [30]: ReDial is conversational recommendation datasets; it consists of over 10K conversations, where users recommend movies to each other. We adopted this dataset and aligned it to task-oriented dialogue to verify the generalizability of the proposed framework. Following [33], we identify entities in the dialogues by the Spacy toolkit[1] and link the entities to a movie knowledge graph provided by DBpedia[1]. In ReDial, the movie mentions and whether the user likes it or not is annotated. There is no action annotation in ReDial, so we constructed action as shown in Figure 3. We then define a user's preferences as a list of movies the user has seen and whether they like them.
- **JDDC** [7]: JDDC is a large-scale, real-world Chinese e-commerce conversation corpus with over 1 million dialogues. We identify entities in the dialogues using the LTP toolkit[2] and link the entities to an e-commerce knowledge graph provided by Aliyun.[3] We define the user's preference as the entities mentioned in the dialogues and design an action based on the informed entity (as shown in Figure 3). Since the original dataset is large and excessive in topics, we sample the dialogues related to the "delivery" domain from JDDC, resulting in 47K dialogues.

We additionally leverage the USS [56] dataset, which provides user satisfaction annotations (on a 5-point scale) both on the exchange level and for entire conversations for the three datasets listed above.

- **USS** [56]: The USS extends a subset of the above datasets with user satisfaction annotations. As per Sect. 4.1.4, we optimize the posterior policy network on partially annotated data and use the posterior network to label the satisfaction on the remaining unlabeled data for the training of the other modules. The posterior network achieved 97.79% accuracy and 88.72% USR on the test set USS. We then leverage these predicted satisfaction labels. We modify the 5-point scale satisfaction annotation of USS (i.e., 1 = Very unsatisfied, 2 = Unsatisfied, 3 = Fair, 4 = Satisfied, 5 = Very satisfied) to a 3-point scale (1 = Unsatisfied, 2 = Fair, 3 = Satisfied). Since USS's satisfaction distribution is unbalanced (most turns express fair satisfaction), we up-sample the non-fair satisfaction turns by increasing the sampling probability of non-fair samples by 10 times.

Summary statics of our training datasets are in Table 1. The processed datasets are available at http://github.com/sunnweiwei/MetaSim.

5.2 Implementations of the evaluation framework

Implemented user simulators. We use T5-base [47] as the backbone of MetaSim for MultiWOZ and ReDial, and optimize the model by AdaFactor with a learning rate of 1e-3. For the Chinese dataset JDDC, we use Mengzi-T5-base [69] and use AdamW with a learning rate of 5e-4. We set the batch size to 64, use a linear lr scheduler with 500 warm-up steps, and train up to 20 epochs. We set the max length to 512 for MultiWOZ and 256 for ReDial and JDDC. We perform greedy decoding for simulators. Due to the characteristics of the data, we replaced the NLU module on the ReDial dataset with a rule-based model that identifies and links the entities in the system utterance to the knowledge graph [1], and serialize the linked subgraphs as system actions.

[1] https://spacy.io/
[2] http://www.ltp-cloud.com/
[3] https://www.aliyun.com/

Table 2. Test set evaluation results of user simulators. Dist. is short for DISTINCT. Boldface indicates the best results in the corresponding metric; significant improvements over the best baseline are marked with * (t-test, $p < 0.05$).

Datasets	MultiWOZ			ReDial			JDDC		
	F1	Dist.	Slot	F1	Dist.	Slot	F1	Dist.	Slot
Agenda [52]	27.25	18.18	27.73	16.58	**24.36**	7.44	13.18	22.42	8.9
AgendaGen [53]	34.51	16.91	24.68	21.06	18.81	7.15	17.32	25.36	10.53
SL+Template [53]	35.54	**20.20**	42.02	27.83	23.27	15.07	12.91	25.28	40.55
Seq2seq [47]	42.02	14.98	49.41	29.75	16.97	11.28	23.45	21.29	37.26
MetaSim	**44.13***	17.29	**57.65***	**33.79***	20.43	**15.72***	**23.85**	**26.37***	**44.00***
MetaSim-Metaphor	43.31	16.83	55.27	32.59	18.94	15.21	23.42	25.41	43.65
MetaSim-Policy	43.11	14.65	49.46	33.67	15.80	13.01	22.35	20.46	39.55
MetaSim-Preference	33.84	12.71	9.76	32.18	14.25	4.35	18.13	19.76	22.32

Table 3. Interaction and human evaluation results of user simulators. Int. is short for Interaction, Dist. is short for DISTINCT, Coh. is short for COHERENCE, and Hum. is short for HUMAN-LIKE. Boldface indicates the best results in the corresponding metric; significant improvements over the best baseline are marked with * (t-test, $p < 0.05$).

Datasets	MultiWOZ				ReDial				JDDC	
	Inter.		Human		Inter.		Human		Inter.	Human
	Success	Dist.	Coh.	Hum.	Success	Dist.	Coh.	Hum.	Dist.	Hum.
Agenda [52]	12.19	14.15	1.96	1.84	4.84	22.97	2.45	1.95	29.37	1.81
AgendaGen [53]	32.17	13.58	2.20	1.98	6.78	**23.04**	2.42	2.10	31.49	1.97
SL+Template [53]	59.78	12.74	2.20	1.75	7.30	19.26	2.64	2.30	28.74	1.86
Seq2seq [47]	76.75	13.76	2.18	2.04	6.18	16.45	2.30	2.35	23.29	2.22
MetaSim	**80.25***	**15.65**	**2.31**	**2.21**	**13.70***	20.97	**2.70**	**2.41**	**34.81***	**2.38**
MetaSim-Metaphor	78.96	15.32	2.25	2.13	12.74	19.36	2.66	2.35	32.65	2.31
MetaSim-Policy	71.54	14.18	2.20	2.17	12.87	17.39	2.35	2.14	30.11	2.14
MetaSim-Preference	0.93	5.20	1.98	1.72	3.43	5.09	2.05	2.05	0.09	1.77

We additionally implement the following baselines on three datasets for comparing against MetaSim:

- **Agenda model** [52] constructs an agenda (that consist of a stack of actions) and pops the top action at each dialogue turn. We build an agenda model by statistically measuring the distribution of action transitions in the training dataset. We retrieve the templates that maximize the override of the slots in action and choose the one with the highest TF-IDF score between the dialogue context.
- **Agenda Gen model** [53] uses the same agenda-based strategy as the Agenda model but generates responses using a generative model instead of retrieved templates. We use a T5-base as the generative model and input the dialogue history and the action drawn from the agenda.
- **SL+Template** [53] predicts employing a retriever in the same way as the Agenda model but using the action predicted by a neural model instead. We use an NLU module and a Policy module to predict the next user action, retrieve a template from the training data, and fill the slot in the template as the response.

- **Seq2seq model** [47] generates response based on dialogue history and preference with a sequence-to-sequence model. We process the data into sequence format and train a T5-base to perform text-to-text generation. Compared to our model, the Seq2seq model doesn't model the policy and metaphor of the user explicitly.

We also compare MetaSim with three modifications:

- **MetaSim-Metaphor** removes the metaphor module and always sets $\mathcal{M}$ to "*none*."
- **MetaSim-Policy** removes the policy module and always sets the user action $a_{t=1}$ to "*none*."
- **MetaSim-Preference** removes the $\mathcal{P}$ and let the model response without constrain.

Implemented testers. We use SOLOIST [43] as the base dialogue system, which is a sequence-to-sequence model based on GPT2 [46]. We implement the base system with hyperparameters of $\{\alpha = 15, \beta = 1, \gamma = 1\}$. Then we set the system variants in context tester as $\mathbb{T}_C = \{S_{\alpha=3}, S_{\alpha=1}\}$; set the system variants in recommender tester as $\mathbb{T}_R = \{S_{\beta=0.4}, S_{\beta=0.1}\}$; and set the system variants in domain tester as $\mathbb{T}_D = \{S_{\gamma=0.1}, S_{\gamma=0.01}\}$. We re-implement the SOLOIST model for ReDail and JDDC.

We use the same data annotation and processing for system implementation as for the user simulator (examples are given in Figure 3). The dialogue system for MultiWOZ and ReDial additionally includes a recommendation module. In MultiWOZ, we generate SQL statements based on belief state. For example, the SQL statements of the belief state in Figure 2 is `select * from Hotel where price=cheap and type=hotel`. The retrieved results are then fill the results back into the responses by string matching following Peng et al. [43]. In ReDial, we implement a recommender as a classifier following [30] (i.e., retrieve the movie by calculating the inner-product of the hidden state produced by the GPT model the movie embeddings). Since the database used in the JDDC dataset is not publicly available [7], we skip the recommendation module in JDDC and do not implement a recommender tester on JDDC.

We optimize the systems using AdamW with a learning rate of 1.5e-4, batch size 32. The systems are trained up to 20 epochs. We set the max length to 512 for MultiWOZ and to 256 for ReDial and JDDC. Following [43], we generate five delexicalized candidates [38] through nucleus sampling, choose the one that contains more slots in the utterance, and back-fill the slots with the retrieval results from the recommender. The code of tester-based framework is available at https://github.com/Superbooming/simtester.

6 EVALUATION AND RESULTS

In this section, we describe how we evaluate the simulators and testers.

- **Evaluating simulators:** To evaluate the realism of the user simulators, we examine the likeness of simulator-generated responses to human responses, and evaluation metrics including word-overlap rate, diversity, entity accuracy, conversation success rate, and human evaluation are employed; see Sect. 6.1 for details.
- **Tester-based evaluation:** To evaluate the capability of simulators to distinguish between systems, we rely on the newly proposed tester-based evaluation framework to test the ability of the simulator to discriminate between dialogue systems with different capabilities; see Sect. 6.2 for details.
- **Evaluating testers:** To validate the reliability of the testers of the tester-based evaluation, we measure the performance of the dialogue system under the tester definition in dialogue benchmarks, interactive tests, and human evaluation; see Sect. 6.3 details.

In addition to this, we conduct a series of analytical experiments, including an in-depth comparison on agenda-based and model-based simulators, efficiency evaluation, human evaluation, and support for evaluating third-party dialogue systems; see Sect. 6.4 for details.

6.1 Evaluating simulators

Evaluation methods. We evaluate the simulators by evaluating the response generated by the simulators and the quality of interactions following three strategies:

- **Test set evaluation** evaluates the simulators on the test set. Using the following metrics, we measure the next user utterance generated based on dialogue context by user simulators: F1[4] is a text generation metric that measures the word overlap between the generated responses and ground-truth responses. DISTINCT [29] estimates the proportion of distinct 3-gram in the data, which indicates the lexical diversity. SLOTACC measures the slot accuracy. SLOTACC equals 1 if all the slots (or entities) in the ground-truth response appear in the generated text, otherwise 0.
- **Interaction evaluation** measures the interaction quality of the simulators. The simulators generate 10K dialogues by interacting with the *base system*, and are evaluated by the SUCCESS rate and DIVERSITY of the simulated dialogues. SUCCESS [13] measures if the recommended item meets all user preferences (we discard SUCCESS for JDDC due to the lack of definition). DISTINCT measures the diversity of simulated dialogues.
- **Human evaluation** evaluates the simulators by the human assessment on simulated dialogues. We ask ten workers to rate each of the simulated dialogues in terms of the COHERENCE and HUMAN-LIKE [53]. Specifically, we randomly sample about 700 dialogues from the simulated data and have people rate a COHERENCE in range 1 to 3, as well as a HUMAN-LIKE score in range 1 to 3 (higher scores are better.).

Results. The experimental results are provided in Table 2 and Table 3. MetaSim consistently outperforms the baseline methods in terms of SLOTACC, which indicates that MetaSim can inform the system about individual requirements more accurately. MetaSim exceeds Seq2seq on SLOTACC with an absolute improvement of +8.24 in MultiWOZ, +4.47 in ReDial and +6.47 in JDDC. The rule-based methods (e.g., Agenda) perform worse than data-driven methods on the test set, especially on JDDC, which indicates that human-curated rules are difficult to cover all the conversation situations. MetaSim also outperforms baselines in terms of F1: the responses generated by MetaSim are more natural than baseline methods. In terms of DISTINCT, MetaSim surpasses all generative methods but it is inferior to the retrieval-based techniques (i.e., Agenda and SL+Template) in MultiWOZ and ReDial. The retrieved responses are more diverse since they are based on templates extracted from human-curated responses from the corpus. Note these responses may not be appropriate for the current conversation.

The experiments on Interaction evaluation indicate that MetaSim achieves a higher task success rate than the baselines, demonstrating better conversational skills (understanding system questions and informing system requirements). MetaSim also exceeds the baselines in terms of diversity except for the agenda model on ReDial data. The human evaluation shows that the simulated dialogue by MetaSim is more coherent to humans and human-like than the baselines.[5] The ablation experiments show that removing modules brings a consistent decrease in performance. Specifically, MetaSim beats MetaSim-Metaphor, indicating a better dialogue ability and diversity via adding the module of Metaphor. Similarly, removing the Policy module leads to a clear performance drop. The MetaSim-Preference is less capable of completing a task-oriented dialogue without preference constraints and a clear task goal.

[4] https://github.com/facebookresearch/ParlAI/blob/master/parlai/core/metrics.py

[5] We do not obtain a significant improvement in the human evaluation due to the limited number of annotations. We consider a large-scale human evaluation as future work.

Table 4. Results of tester-based evaluation. $\mathbb{T}_C$ denotes the Context tester, $\mathbb{T}_R$ denotes the Recommender tester, and $\mathbb{T}_D$ denotes the Domain tester. Human evaluation results are only included for MultiWOZ as detailed in Sect. 6.2.

Datasets	**MultiWOZ**			**ReDial**			**JDDC**	
Tester	$\mathbb{T}_C$	$\mathbb{T}_R$	$\mathbb{T}_D$	$\mathbb{T}_C$	$\mathbb{T}_R$	$\mathbb{T}_D$	$\mathbb{T}_C$	$\mathbb{T}_D$
Agenda	16.93	17.28	19.47	16.41	17.12	17.05	13.76	14.52
Seq2seq	30.84	37.15	32.81	16.52	17.46	19.16	18.47	16.89
MetaSim	**33.23**	**40.11**	**35.21**	**17.48**	**20.61**	**20.43**	**19.17**	**17.47**
Human	43.63	40.54	42.54	–	–	–	–	–

6.2 Tester-based evaluation

Evaluation methods. We conduct experiments with our testers and compare several interactive evaluators (i.e., simulators and humans) in terms of ExactDistinct to evaluate the simulators. Following the strategies described in Sect. 4.2.2, we implement three testers for each of the three datasets except for JDDC (cf. Sect. 5.2). We initialize the preferences from the preference module and let the simulators interact with system variants defined by testers. The simulators give a rating of the system after the interaction. We use user calibrated user satisfaction, i.e., the average of satisfaction and task success (as described in Sect. 4.1.6) as the ratings of models. If the scoring of the two system variants is the same, we further distinguish them by the number of dialogue turns (the one with fewer rounds is better). We then rank the variants based on the ratings and check if the order is the same as expected (i.e., $S_1 < S_2 < S_3$). The simulator will be scored 1 on ExactDistinct if the order given by the simulator is consistent with expectations, otherwise scored as 0. We repeated the testing on 1K different preferences and calculated the averaged ExactDistinct score.

We also conduct a human evaluation on MultiWOZ to validate the reliability, i.e., if the human can distinguish the system variants. Specifically, we develop an annotation system and ask 40 annotators to talk with the system variants to complete a specific goal. We randomly assign systems and goals to users to avoid bias caused by order of operations. After the dialogue, the annotator is asked to rate the system in terms of Satisfaction and Success. We use the averaged Satisfaction and Success as the ratings of the system.

Results. Table 4 shows the results of the tester evaluation. First, MetaSim is more capable of distinguishing the system variants defined in a tester and achieving the best in terms of the ExactDistinct score than Agenda model and Seq2seq model. This might be because the agenda model is limited to domain-specific human-curated rules and difficult to generalize, and end-to-end simulators can only evaluate the systems on the semantic level. Second, the difficulty of the tester varies, as does the difficulty of the tester on different data (e.g., Recommender tester is more difficult for people than others, because the responses generated by system defined by Recommender tester look reasonable and may confuse users). Third, people can better distinguish between different systems compared to simulators (e.g., Human get an ExactDistinct of 43.63, 40.54, and 42.54 for the three testers on MultiWOZ, compared to 33.23, 40.11, and 35.21 for MetaSim). Note there still exists a performance gap between MetaSim and Humans, indicating that building human-like simulators is still a challenge. We leave it for future work of investigating the reasons behind it.

6.3 Evaluating testers

Evaluation methods. To assess the variants of these testers with sensible performance differences, we evaluate their performance in the test set, interaction quality, and human assessment.

Table 5. Evaluation results of variants on the MultiWOZ dataset. Dist denotes Distinct, Sat. denotes Satisfaction, Eff. denotes Efficiency, and Nat. denotes Naturalness.

	Test Set			Interaction		Human			
	BLEU	Success	Slot	Success	Dist	Sat.	Success	Eff.	Nat.
System	20.42	75.00	51.92	80.25	15.32	3.54	39.50	84.57	2.37
$\alpha = 3$	18.84	57.65	51.75	40.20	13.76	3.07	29.09	74.54	2.15
$\alpha = 1$	16.81	46.70	44.59	27.04	15.16	2.32	12.20	51.22	1.66
$\beta = 0.4$	20.49	38.17	51.69	28.60	15.27	3.26	24.53	77.36	2.13
$\beta = 0.1$	20.53	34.90	52.33	7.40	10.17	3.24	13.33	71.11	2.07
$\gamma = 0.1$	16.10	42.53	33.66	31.70	15.84	3.10	16.33	75.51	2.06
$\gamma = 0.01$	13.18	38.54	20.02	25.30	14.51	2.55	2.13	59.57	1.89

Table 6. Evaluation results of variants on the ReDial and JDDC datasets; same conventions as in Table 5. No results are included for the β (recommender tester) for JDDC as JDDC does not have a recommender component due to the unavailability of its database (cf. Sect. 5.2).

Datasets	**ReDial**					**JDDC**			
	Test Set			Interaction		Test Set			Interaction
	BLEU	Success	Slot	Success	Dist	BLEU	Dist	Slot	Dist
System	15.42	8.20	3.74	13.70	20.97	25.45	10.68	8.72	34.81
$\alpha = 3$	14.87	8.20	3.37	11.45	23.01	21.76	9.49	6.80	31.10
$\alpha = 1$	14.34	6.93	2.91	9.62	21.13	14.64	6.84	1.14	29.74
$\beta = 0.4$	15.44	1.86	2.96	6.54	21.43	–	–	–	–
$\beta = 0.1$	14.80	0.75	2.75	2.45	27.14	–	–	–	–
$\gamma = 0.1$	14.91	4.62	3.03	5.95	18.52	22.48	10.05	6.18	33.68
$\gamma = 0.01$	13.99	1.19	1.45	2.27	17.26	17.5	9.94	1.92	34.25

- **Test set evaluation** evaluates the system variants defined by testers on the test set. Specifically, similar to the way to evaluate simulators, we measure the generated utterances using the following metrics: BLEU [41] measures the word-overlap between the generated utterance the ground-truth utterance. Success measures if the recommended item meets all user preferences. SlotAcc measures the slot accuracy.
- **Interaction evaluation** evaluates the system by interacting with MetaSim. We generate 10K dialogue and calculate the Satisfaction and Distinct. Success measures if the system recommend an item that meets all the user needs. Distinct measures the diversity of simulated dialogue.
- **Human evaluation** evaluates the systems by human experiments. Human evaluation was operated on the annotation system we built for MultiWOZ and invited 40 users to talk to the system based on the predefined preference. We collected about 500 dialogues, where the users were instructed to label the following aspects at the end of each dialogue: (i) Success measures if the system solves the user's problem, rating as: 1 = Fail, 2 = Success; (ii) Efficiency measures the efficiency of the system, rating as: 1 = Inefficient, 2 = Efficient; (iii) Naturalness measures the human-likeness of the system, rating as: 1 = Unnatural,

2 = Fair, 3 = Natural; (iv) SATISFACTION is labeled on a 5-point scale: 1 = Very bad, 2 = Bad, 3 = Fair, 4 = Good, 5 = Very good.

Results. Table 5 shows the evaluation results on MultiWOZ and Table 6 shows the evaluation results on ReDial and JDDC. We have three main findings from the results.

First, the differences between system variants can be distinguished by humans. The user satisfaction annotations (as an overall rating for the systems) are consistent with our expectations, while other human evaluation metrics demonstrate the same results but with a focus on the specific capabilities of the system. We also notice a smaller performance gap between the recommender tester-defined human evaluation variants and the other two testers. This is because of human annotators' difficulties judging the goodness of the plausible-looking results recommended by different systems, a finding consistent with that in tester evaluation; see Sect. 6.2.

Second, each tester has focused on a specific capability of the system. For example, the recommender tester largely disrupts the system's ability to make recommendations (low SUCCESS in both test set, interaction, and human evaluation), while keeping the ability to generate natural responses (relatively good BLEU and NATURALNESS). Therefore, we measure a user simulator's capabilities to evaluate the dialogue system's recommendation capabilities by checking whether it can accurately distinguish the differences between the system variants defined by the recommender tester.

Third, the interaction experiments also exhibit results consistent with expectations, but the difference between good and bad systems are more significant. This could be attributed to the accumulation of errors in the interaction amplifies the difference in system performance. For example, in Table 5, we see that the SUCCESS gap between the base system and system variant with $\alpha = 3$ in interaction evaluation is more significant than that of in test set evaluation ($75.00 - 46.70 = 28.30$ compared to $80.25 - 27.04 = 53.21$). We notice that the diversity (DISTINCT) of the variants do not follow the rule that "*weaker variants perform worse*". For example, in Table 5 we see that the system variant with $\alpha = 1$ achieves a higher DISTINCT score (15.16) compared to the system variant with $\alpha = 3$ (13.76)). This is possibly because the designed testers lacked control over response generation diversity.

6.4 Analytical experiments

We conduct further analytical experiments to understand the results better.

Comparison of agenda-based and model-based user simulators. User simulators fall into two main categories, agenda-based user simulators and model-based user simulators [53, 70]. To bootstrap the agenda-based user simulator, we might use human-defined rules, such as a manually designed state transition matrix, for updating dialogue policy. In contrast, model-based user simulators learn dialogue policy from the corpus. In Sect 6.1, we define the state transition matrix of the agenda-based simulators by statistically measuring the distribution of the training dataset. Here, we compare MetaSim with the agenda-based user simulators implemented in ConvLab2 [70], which are state-of-the-art multi-domain agenda-based simulators for MWOZ and have been included as the standard test kits for DSTC competitions[6]. The agenda-based user simulators implemented in ConvLab2 consist of three modules: NLU, policy, and NLG. The NLU module employs a BERT-based model trained on MultiWOZ to identify system actions. The policy module employs sophisticated, manually defined rules to produce user actions. The NLG module converts user actions into responses; two strategies are employed: *Template* handwrite templates for each action, and *Retrieval* use actions as keywords to retrieve relevant responses in MultiWOZ training data and replace entities. We denote the simulators as *ConvLab2+Template* and *ConvLab2+Retrieval*, depending on

[6] https://github.com/thu-coai/ConvLab-2

Table 7. Result comparison between agenda-based and model-based user simulators on MultiWOZ. The top block shows the results of different variants of agenda-based user simulators. The bottom block shows model-based user simulators (MetaSim). The numbers in parentheses represent the number of training dialogues.* indicates statistically significant improvements over baselines (p-value < 0.05).

Datasets	**Test set**			**Interaction**		**Human**	
	F1	Dist.	Slot	Success	Dist.	Coh.	Hum.
Agenda-based models							
ConvLab2+Template [70]	18.26	5.77	18.25	27.10	7.28	1.65	1.74
ConvLab2+Retrieval [70]	18.39	**18.66**	18.70	18.10	**20.47**	1.46	1.84
ConvLab2+Template (oracle) [70]	20.00	5.91	21.39	-	-	-	-
ConvLab2+Retrieval (oracle) [70]	19.73	18.53	21.42	-	-	-	-
Model-based models							
MetaSim	**44.13***	17.29	**57.65***	**80.25***	15.65	**2.51**	**2.61**
MetaSim (1k)	41.68*	14.75	50.56*	59.30*	16.31	2.23	2.51
MetaSim (100)	35.86*	12.30	31.60*	35.90*	13.56	1.80	2.10

their NLG strategies. We encounter *order inconsistencies* between the agenda built automatically by agenda-based simulators and the agenda of the real dialogue in test set evaluation. For example, assume a user plans to complete two tasks in conversation, booking a restaurant and booking a train by order. However, the simulator might automatically build the agenda to book the train first and then book the restaurant. It may be able to accomplish two tasks as well. The order inconsistency of agenda order would degrade the performance of agenda-based simulators in the test set evaluation. Therefore, we also evaluate the agenda-based simulator under *oracle condition*, i.e., we initialize the simulator's agenda with the agenda of the real conversation in the test set evaluation. We did not employ the agenda-based simulators designed by Shi et al. [53] because they only developed rules for the restaurant reservation domain.

We evaluate the performance of ConvLab2 simulators and MetaSim regarding test set, interaction quality, and human assessment. We also include two variants of MetaSim in evaluation: MetaSim (1k) and MetaSim (100), which are MetaSim models trained using 1k annotated dialogues and 100 annotated dialogues, respectively. The results are listed in Table 7. Note that human evaluation in Table 3 and Table 7 are two independent experiments in which the participating annotators, the evaluated dialogues, and the instruction are different; therefore, the absolute values of their results are different. From the results, we have two main findings: (i) The model-based user simulators significantly outperform the rule-based (e.g., agenda-based) simulators. We find that the model-based user simulators can generate more contextual responses and have a higher success rate when interacting with the system. Their interactive dialogues are considered more human-like in human evaluations. To further compare the behavior of agenda-based methods and our proposed method, we listed the number of interaction turns between the base system and user simulators in Figure 5. From the results, MetaSim's interaction rounds are within ten rounds. In comparison, the ConvLab2 Agenda model's interaction rounds have more than 20 rounds, showing the advantage of MetaSim in efficiency. (ii) The model-based simulator performs better when using a small amount of annotated data. For example, MetaSim (100) outperforms agenda-based simulators using 100 annotated dialogues. (iii) Retrieval-based models, such as ConvLab2+Retrieval, outperform generation-based models in terms of diversity. This could be because retrieval-based methods pick human-written responses under the predicted action more randomly from the training. In contrast, generative methods generate similar responses when presented with the same action. Additionally,

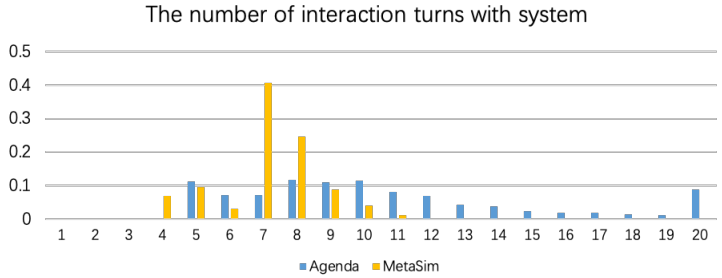

Fig. 5. The number of turn of interaction between base system and user simulators.

Table 8. Evaluation results of MetaSim variants on MultiWOZ. * indicates statistically significant improvements over Seq2seq method (p-value < 0.05).

	Storage	Time	F1	Distinct	Slot	Success	Interact-Distinct
Seq2seq	0.89	409	42.02	14.98	49.41	76.75	13.76
MetaSim	3.56	1,987	44.13*	**17.29***	57.65*	80.25	**15.65**
Variant A	2.67	1,250	**44.56***	16.64*	**58.32***	**81.10**	14.99
Variant B	0.89	657	43.15*	14.76	54.46*	77.40	10.97
Variant C	0.92	471	43.99*	10.57	57.39*	79.60	11.88

using a sampling-based decoding method (like nucleus sampling) can enhance the diversity of the generative model. We plan further to enhance the model generation's diversity in future work.

The efficiency of MetaSim. The proposed MetaSim uses modular modeling, i.e., different parts (NLU, Policy, etc.) are modeled separately using different T5 models. The ablation study in Table 2 shows the model's performance with some of the modules removed. Here, we investigate whether specific modules can be consolidated together to improve efficiency and how these changes affect the model's performance. The tested model variants are: (i) *Variant A: Consolidate policy and NLG modules.* We use one T5 model to sequentially predict the action a_{t+1} and the response r_{t+1}. The action and response are concatenated with a special token </s>. (ii) *Variant B: Consolidate NLU, policy, and NLG modules in one T5 model.* This variant consolidates the NLU, policy, and NLG modules in one T5 model by sequentially generating system action, user action, and user response. The prediction of the metaphor module is removed. (iii) *Variant C: MetaSim with T5-small.* This variant replaces the T5-base model with T5-small.

The model's results on MultiWOZ variants are listed in Table 8. The metrics are the Storage space of the model (GB), inference time (seconds) on the MultiWOZ test set, F1, Distinct, and SlotAcc of the generated response. We can see that variants A and B reduce the model storage space and computational overhead by merging the modules and achieving very competitive results. For example, variant A performs comparably to MetaSim by mildly outperforming in terms of F1 and Slot and lags behind MetaSim in Distinct; this may be because the action prediction task and the response generation task share similar capabilities and can be done using a single model. Variant C, which replaces the T5-base with T5-small, also significantly reduces computational overhead: it is 4× faster than MetaSim with the T5-base model. Variant C also achieves competitive results in terms of F1 and Slot but is ineffective in generating diverse responses (e.g., low Distinct). This may be due to model capacity limitations whereby variant C generates generic responses.

Table 9. Evaluation results of human evaluation. We compare three settings: 3-point scale scoring, 5-point scale scoring, and rank-based evaluation. Score denotes the average scores of the models. Best and Worst measure the number of cases in which the corresponding models perform the best or worst among all the models.

Datasets	**3-Scale Score**			**5-Scale Score**			**Rank**		
	Score	Best	Worst	Score	Best	Worst	Score	Best	Worst
Agenda	1.33	27	255	1.93	15	248	1.13	5	239
Seq2seq	2.31	160	84	3.53	105	52	2.20	78	25
MetaSim	2.72	244	32	4.15	210	16	2.67	187	7
Kappa		34.56			22.78			60.00	

Table 10. Evaluation results of dialogue systems on MultiWOZ.

	Benchmark				Interaction				Human		
	Inform	Success	BLEU	Combine	Success	Distinct	Sat.	Combine	Sat.	Effic.	Natural.
DAMD [28]	76.40	60.40	16.60	85.00	20.90	10.14	2.15	0.392	1.79	1.88	1.87
SOLOIST [43]	85.50	72.90	16.54	95.74	71.00	**14.23**	2.74	0.790	2.40	2.54	2.52
GALAXY [17]	**95.40**	**80.70**	**17.00**	**105.10**	**74.90**	9.59	**2.78**	**0.819**	**2.66**	**2.58**	**2.57**

Human evaluation methods. In previous experiments, we use 3-scale in human evaluation in Table 3. Here we compare the different human evaluation methods, they are: (i)*Score-based evaluation in 3-scale.* Human annotators are asked to rate a score to the responses in range 1-3. (ii) *Score-based evaluation in 5-scale.* Human annotators are asked to rate a score to the responses in range 1-5. (iii) *Rank-based evaluation.* Human annotators are asked to rank multiple responses of different models with respect to the dialogue context. The metrics we use are: *Score*, the average score of the model; *Best*, the number of examples that the model performs the best (or tied for the best) among models; *Worst*, the number of examples that the model performs the worst (or tied for the worst) among models. We also calculate the *Kappa*, i.e., the agreement of the three human annotators of the three evaluation methods. Table 9 lists the results. MetaSim is considered more realistic than baseline models Agenda and Seq2seq across different human evaluation methods: it obtains higher average scores and is considered the best model in more examples. We also find that the agreement of 3-scale scoring is better compared to 5-scale scoring, while the agreement of the rank-based evaluation method is the best.

Test results of third-party dialogue systems. We used SOLOIST as the base dialogue system in the experiments described above. Here, we conduct experiments on some third-party dialogue systems, which are: (i) *DAMD* [67], a modular TDS system where each dialog module is implemented using a neural network, and the whole system is trained in an end-to-end manner; (ii) *SOLOIST* [43], a TDS models pre-trained on heterogeneous dialog corpora; (iii) *GALAXY* [17] a pre-trained TDS model that explicitly learns dialog policy from limited labeled dialogues and large-scale unlabeled dialog corpora via semi-supervised learning. The evaluation results are listed in Table 10. We can see that: (i) GALAXY performs the best in three evaluation methods, while DAMD performs the worst. (ii) User simulators (i.e., MetaSim) have the ability to discriminate between different capability systems, and their predicted model success rate, user satisfaction, and combined scores (i.e., an average of success and satisfaction), all meet expectations. Table 11 illustrates an example

Table 11. Example of dialogues using DAMD, SOLOIST and GALAXY.

Goal	You are planning your trip in Cambridge. You are looking for a place to stay. The hotel should be in the cheap price range and should include free parking. The hotel should include free wifi. You are also looking for a train. The train should arrive by 09:30 and should go to cambridge. The train should leave on monday and should depart from london kings cross.
Model	DAMD
user:	i'm looking for a cheap place to stay that offers free parking.
system:	there are 22 restaurants that fit your criteria . do you have a preference for area ?
user:	no, but it should also have free wifi.
system:	i would recommend the pizza hut city centre . it is in the centre .
user:	could i get the postcode?
system:	pizza hut city centre is a italian restaurant in the centre . it is in the cheap price range .
user:	i think that is good. can i get the post code, please?
system:	the postcode is cb21ab . their phone number is 01223323737 . is there anything else i can help you with ?
user:	...
Model	SOLOIST
user:	i'm looking for a place to stay that includes free wifi and free parking.
system:	there are 29 places that meet that criteria . is there a particular area you would like to stay in ?
user:	no particular area, but i would like it to be cheap.
system:	i have 10 guesthouses that fit that criteria . 5 in the centre and 2 in the east . do you have a preference ?
user:	i would like the one in the east please.
system:	allenbell is a cheap guesthouse in the east with free wifi and parking . would you like to book a room ?
user:	no, but can you give me the postcode?
system:	the postcode for allenbell is cb13js . can i help you with anything else today ?
user:	...
Model	GALAXY
user:	i'm looking for a cheap place to stay that offers free parking.
system:	i have 10 places that fit that criteria . is there a particular area you would like to stay in ?
user:	no, but it should also have free wifi.
system:	i have 10 places that meet that criteria . is there a particular area you would like to stay in ?
user:	i don't have a particular area. which would you recommend?
system:	i would recommend the alexander bed and breakfast in the centre . would you like me to book a room for you ?
user:	i don't need to book a room, but could you tell me the hotel type, phone number, and address please?
system:	it is a guesthouse located at 56 saint barnabas road . the phone number is 01223525725 .
user:	...

Table 12. Evaluation results on unseen domain. **Boldface** indicates significant improvements over baselines with (p-value < 0.05)

Datasets	Hotel			Attraction		
	F1	Dist.	Slot	F1	Dist.	Slot
Seq2seq	34.21	8.16	33.11	37.26	8.50	39.05
MetaSim	**39.73**	**12.41**	**39.79**	**41.45**	**12.69**	**50.12**
MetaSim-Metaphor	35.63	10.70	37.38	38.07	10.58	47.66
MetaSim-Ranking	37.89	11.80	37.00	40.02	12.36	47.72

of a conversation between MetaSim and three dialogue systems, in which we can see that DAMD did not understand the user's needs when they first asked for a postcode. SOLOIST and GALAXY performed well, but GALAXY had some expression repetition problems.

Test results on unseen domains. We further conducted experiments on unseen domains. We removed all samples, including "hotel" or "attraction" domains in the training set, respectively, trained the model, and tested it on the subset of the test set containing "hotel" and "attraction". Table 12 shows the results. We observe that the proposed model achieves the best results. MetaSim-Metaphor,

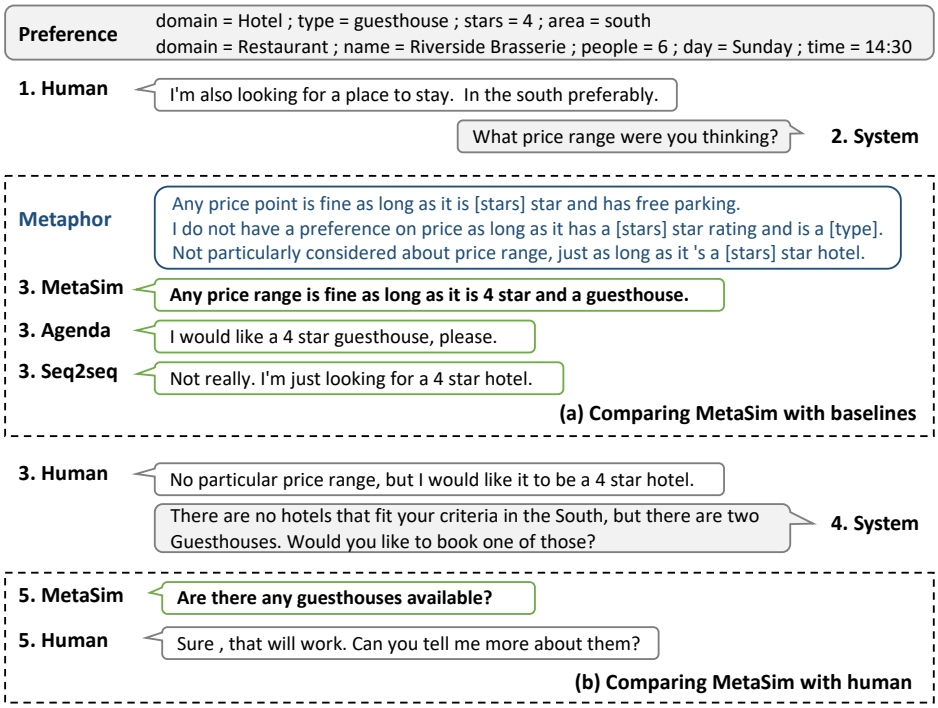

Fig. 6. An example dialogue with responses generated by MetaSim, Agenda, and Seq2seq. In (a), we compare the predictions of MetaSim and baselines for turn 3. In (b), we compare the predictions of MetaSim and Human for turn 5.

a MetaSim variant that does not use the retrieved records, shows a clear drop in effectiveness. MetaSim-Ranking, the variant with the ranking module removed from the metaphor module, had a noticeable drop compared to MetaSim, especially in the Slot accuracy metric. Finally, compared to the Seq2seq model, MetaSim shows a significant improvement, possibly due to its ability to respond more appropriately to items of new domains by retrieving known similar records. For instance, many behaviors in hotel reservations are similar to known restaurant reservation records, and a reasonable association can improve the model's response.

6.5 Case studies

We display examples of responses predicted by MetaSim, Agenda, Seq2seq, and a human annotator (all given the same user preference) for the same dialogue in Figure 6. In Figure 6 (a), we see that the response generated by MetaSim is more natural and coherent to the dialogue than the Agenda model. In Figure 6 (b), we see a mistake by MetaSim, where human asks further information about Guesthouses while MetaSim fails to predict the action.

In Figure 7, we show examples of simulated dialogues of MetaSim with three variants defined by the Recommender tester, as well as the rating of the systems. These variants have sensible capabilities, as witnessed by these examples, i.e., $S_{\gamma=1}$ successfully books a taxi that meets the user's requirements, $S_{\gamma=0.1}$ completes a booking, but the booking time is wrong, and $S_{\gamma=0.01}$ fails in this task. MetaSim is able to distinguish between the three simulated dialogues with clearly different scores.

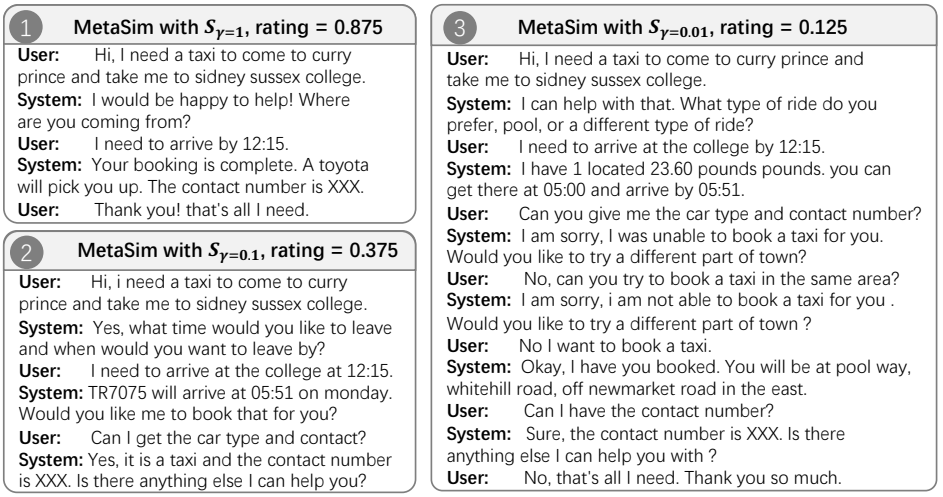

Fig. 7. Simulated dialogues of MetaSim with Domain tester, as well as ratings by the simulator.

The two case studies illustrate that the MetaSim is capable of generating natural and coherent responses and of distinguishing between systems of different quality.

7 CONCLUSIONS

In this paper, we have studied the problem of evaluating task-oriented dialogue systems using user simulators. Based on the identified challenges when employing user simulators in evaluating TDSs, i.e., the lack of automatic methods to evaluate the user simulators, we have proposed a solution target at improving the realism of the current user simulator for better evaluating TDS. Specifically, we have proposed the metaphorical user simulator (MetaSim) for end-to-end TDS evaluation and a tester-based evaluation framework for evaluating the evaluation capability of the simulators.

We have conducted extensive experiments on three benchmark TDS datasets, i.e., MultiWOZ, ReDial, and JDDC. Our experiments have shown that the proposed simulator MetaSim achieves better consistency with manual evaluation on those three datasets and that it is able generate more realistic dialogues than the Agenda-based simulator and a Seq2seq model. We also found that system variants defined by different settings of the tester display meaningful performance differences.

A broader lessons that we arrive at through our study is as follows. Realism is important when constructing user simulators for evaluating dialogue systems. In this work, we have taken important steps towards more realistic user simulators, by constructing mental models. User simulators allowing for a more realistic assessment can reduce the reliance on human effort in the loop of evaluation. Moreover, a test-based evaluation framework reduces the dependence on third-party agents, and can assess user simulators appropriately.

Recall that in Sect. 4.1.4, we predict the user action and satisfaction jointly. But the combination of user simulation and satisfaction assessment is still preliminary as we simply perform a sequence prediction and add the results of both (see Sect. 4.1.6). Also, our simulator design with multiple modules is computationally intensive and thus limits the inference efficiency. Moreover, our research on establishing mental models is still in its early stages, with limited evaluation of its effectiveness in multi-domain and cross-task performance.

In future work, we would like to explore more effective combination of user simulation and satisfaction assessment. In addition, we would like to improve tester-based evaluation strategies and

simulator design so that it can be more efficient, for example, reducing the number of interactions between tester and simulators and employing end-to-end modeling [28]. Moreover, we would like to extend the tester framework to support a broader range of simulator evaluation demands, such as conversational question answering and knowledge-grounded dialogue.

DATA AND RESOURCES

This paper only made use of publicly available datasets. The code used to produce the results in this paper is available at http://github.com/sunnweiwei/MetaSim. The code of tester-based framework is available at https://github.com/Superbooming/simtester.

REFERENCES

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*. Springer, 722–735.

[2] Krisztian Balog. 2021. Conversational AI from an Information Retrieval Perspective: Remaining Challenges and a Case for User Simulation. In *DESIRES 2021*.

[3] Krisztian Balog, David Maxwell, Paul Thomas, and Shuo Zhang. 2021. Sim4IR: The SIGIR 2021 Workshop on Simulation for Information Retrieval Evaluation. In *SIGIR 2021*. 2697–2698.

[4] Alan W. Black, Susanne Burger, Alistair Conkie, H. Hastie, Simon Keizer, Oliver Lemon, Nicolas Merigaud, Gabriel Parent, Gabriel Schubiner, Blaise Thomson, J. Williams, Kai Yu, Steve J. Young, and Maxine Eskénazi. 2011. Spoken Dialog Challenge 2010: Comparison of Live and Control Test Results. In *SIGDIAL 2011*. 2–7.

[5] Dimitrios Bountouridis, Jaron Harambam, Mykola Makhortykh, Mónica Marrero, Nava Tintarev, and Claudia Hauff. 2019. SIREN: A Simulation Framework for Understanding the Effects of Recommender Systems in Online News Environments. In *FAccT 2019*. 150–159.

[6] Ben Carterette, Evangelos Kanoulas, and Emine Yilmaz. 2011. Simulating Simple User Behavior for System Effectiveness Evaluation. In *CIKM 2011*. 611–620.

[7] Meng Chen, Ruixue Liu, Lei Shen, Shaozu Yuan, Jingyan Zhou, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. The JDDC Corpus: A Large-Scale Multi-Turn Chinese Dialogue Dataset for E-commerce Customer Service. In *LREC 2020*. 459–466.

[8] Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. 2005. Human-computer Dialogue Simulation using Hidden Markov Models. *IEEE Workshop on ASRU 2005* (2005), 290–295.

[9] Rajarshi Das, Patrick Lewis, Sewon Min, June Thai, and Manzil Zaheer (Eds.). 2022. *Proceedings of the 1st Workshop on Semiparametric Methods in NLP: Decoupling Logic from Knowledge*.

[10] Jan Deriu, Álvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. 2020. Survey on Evaluation Methods for Dialogue Systems. *Artificial Intelligence Review* 54 (2020), 755–810.

[11] Wieland Eckert, Esther Levin, and Roberto Pieraccini. 1997. User Modeling for Spoken Dialogue System Evaluation. In *IEEE Workshop on ASRU 1997*. 80–87.

[12] Jens Edlund, Joakim Gustafson, Mattias Heldner, and Anna Hjalmarsson. 2008. Towards Human-like Spoken Dialogue Systems. *Speech Commun.* 50 (2008), 630–645.

[13] Mihail Eric, Rahul Goel, Shachi Paul, Adarsh Kumar, Abhishek Sethi, Anuj Kumar Goyal, Peter Ku, Sanchit Agarwal, and Shuyang Gao. 2020. MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines. In *LREC 2020*. 422–428.

[14] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2021. Advances and Challenges in Conversational Recommender Systems: A Survey. *AI Open* 2 (2021), 100–126.

[15] Kallirroi Georgila, James Henderson, and Oliver Lemon. 2005. Learning User Simulations for Information State Update Dialogue Systems. In *INTERSPEECH 2005*.

[16] Ting Han, Ximing Liu, Ryuichi Takanobu, Yixin Lian, Chongxuan Huang, Dazhen Wan, Wei Peng, and Minlie Huang. 2021. MultiWOZ 2.3: A Multi-domain Task-Oriented Dialogue Dataset Enhanced with Annotation Corrections and Co-Reference Annotation. In *NLPCC*. 206–218.

[17] Wanwei He, Yinpei Dai, Yinhe Zheng, Yuchuan Wu, Zheng Cao, Dermot Liu, Peng Jiang, Min Yang, Fei Huang, Luo Si, Jian Sun, and Yongbin Li. 2021. GALAXY: A Generative Pre-trained Model for Task-Oriented Dialog with Semi-Supervised Learning and Explicit Policy Injection. In *AAAI 2022*.

[18] Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A Simple Language Model for Task-Oriented Dialogue. *ArXiv* abs/2005.00796 (2020).

[19] Jin Huang, Harrie Oosterhuis, Maarten de Rijke, and Herke van Hoof. 2020. Keeping Dataset Biases out of the Simulation: A Debiased Simulator for Reinforcement Learning based Recommender Systems. In *RecSys 2020*.

[20] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2021. A Survey on Conversational Recommender Systems. *ACM Computing Surveys (CSUR)* 54 (2021), 1–36.
[21] Xisen Jin, Wenqiang Lei, Zhaochun Ren, Hongshen Chen, Shangsong Liang, Yihong Eric Zhao, and Dawei Yin. 2018. Explicit State Tracking with Semi-Supervisionfor Neural Dialogue Generation. In *CIKM 2018*. 1403–1412.
[22] Natalie A Jones, Helen Ross, Timothy Lynam, Pascal Perez, and Anne Leitch. 2011. Mental Models: an Interdisciplinary Synthesis of Theory and Methods. *Ecology and Society* 16 (2011), 46.
[23] Filip Jurcícek, Simon Keizer, Milica Gasic, François Mairesse, Blaise Thomson, Kai Yu, and Steve J. Young. 2011. Real User Evaluation of Spoken Dialogue Systems Using Amazon Mechanical Turk. In *INTERSPEECH 2011*.
[24] Anna Kaal. 2012. *Metaphor in Conversation*. Ph.D. Dissertation. Vrije Universiteit Amsterdam.
[25] Sahiti Labhishetty and ChengXiang Zhai. 2021. An Exploration of Tester-based Evaluation of User Simulators for Comparing Interactive Retrieval Systems.. In *SIGIR 2021*.
[26] Lori Lamel, Sophie Rosset, Jean-Luc Gauvain, Samir Bennacef, Martine Garnier-Rizet, and Bernard Prouts. 2000. The LIMSI ARISE system. *Speech Commun.* 31 (2000), 339–353.
[27] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-Action-Reflection: Towards Deep Interaction Between Conversational and Recommender Systems. In *WSDM 2018*.
[28] Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying Task-oriented Dialogue Systems with Single Sequence-to-Sequence Architectures. In *ACL 2018*. 1437–1447.
[29] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In *NAACL 2016*. 110–119.
[30] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Christopher Joseph Pal. 2018. Towards Deep Conversational Recommendations. *ArXiv* abs/1812.07617 (2018).
[31] Weixin Liang, Youzhi Tian, Chengcai Chen, and Zhou Yu. 2020. MOSS: End-to-End Dialog System Framework with Modular Supervision. In *AAAI 2020*.
[32] Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul A. Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. 2021. Leveraging Slot Descriptions for Zero-Shot Cross-Domain Dialogue StateTracking. In *NAACL 2021*.
[33] Wenchang Ma, Ryuichi Takanobu, and Minlie Huang. 2021. CR-Walker: Tree-Structured Graph Reasoning and Dialog Acts for Conversational Recommendation. In *EMNLP 2021*. 1839–1851.
[34] Andrea Madotto and Zihan Liu. 2020. Language Models as Few-Shot Learner for Task-Oriented Dialogue Systems. *ArXiv* abs/2008.06239 (2020).
[35] David Maxwell and Leif Azzopardi. 2016. Agents, Simulated Users and Humans: An Analysis of Performance and Behaviour. In *CIKM 2016*.
[36] Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Pei hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2015. Multi-domain Dialog State Tracking using Recurrent Neural Networks. In *ACL 2015*.
[37] Nikola Mrksic, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. 2017. Neural Belief Tracker: Data-Driven Dialogue State Tracking. In *ACL 2017*.
[38] Tomás Nekvinda and Ondrej Dusek. 2021. Shades of BLEU, Flavours of Success: The Case of MultiWOZ. In *Workshop on GEM 2021*. 34–46.
[39] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy J. Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *EMNLP 2020*. 708–718.
[40] Alexandros Papangelis, Yi-Chia Wang, Piero Molino, and Gökhan Tür. 2019. Collaborative Multi-Agent Dialogue Model Training Via Reinforcement Learning. In *SIGdial 2019*. 92–102.
[41] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *ACL 2002*. 311–318.
[42] Surendra Pathak, Sheikh Ariful Islam, Honglu Jiang, Lei Xu, and Emmett Tomai. 2022. A Survey on Security Analysis of Amazon Echo Devices. *High-Confidence Computing* (2022).
[43] Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Lidén, and Jianfeng Gao. 2021. SOLOIST: Building Task Bots at Scale with Transfer Learning and Machine Teaching. *TCAL* 9 (2021), 807–824.
[44] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vassilis Plachouras, Tim Rocktaschel, and Sebastian Riedel. 2021. KILT: A Benchmark for Knowledge Intensive Language Tasks. In *NAACL 2021*. 2523–2544.
[45] Olivier Pietquin and Helen Hastie. 2012. A Survey on Metrics for the Evaluation of User Simulations. *The Knowledge Engineering Review* 28 (2012), 59–73.
[46] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. In *OpenAI blog*. 9.

[47] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *JMLR* 21 (2020), 1–67.

[48] Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A Conversational Question Answering Challenge. *TCAL* 7 (2019), 249–266.

[49] Pengjie Ren, Zhongkun Liu, Xiaomeng Song, Hongtao Tian, Zhumin Chen, Zhaochun Ren, and Maarten de Rijke. 2021. Wizard of Search Engine: Access to Information Through Conversations with Search Engines. In *SIGIR 2021*.

[50] Zhaochun Ren, Zhi Tian, Dongdong Li, Pengjie Ren, Liu Yang, Xin Xin, Huasheng Liang, Maarten de Rijke, and Zhumin Chen. 2022. Variational Reasoning about User Preferences for Conversational Recommendation. In *SIGIR 2022*.

[51] Lina Maria Rojas-Barahona, Milica Gasić, Nikola Mrksic, Pei hao Su, Stefan Ultes, Tsung-Hsien Wen, Steve J. Young, and David Vandyke. 2017. A Network-based End-to-End Trainable Task-oriented Dialogue System. In *EACL 2017*.

[52] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve J. Young. 2007. Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System. In *NAACL 2007*.

[53] Weiyan Shi, Kun Qian, Xuewei Wang, and Zhou Yu. 2019. How to Build User Simulators to Train RL-based Dialog Systems. In *EMNLP 2019*. 1990–2000.

[54] Eric Michael Smith, Orion Hsu, Rebecca Qian, Stephen Roller, Y-Lan Boureau, and Jason Weston. 2022. Human Evaluation of Conversations is an Open Problem: Comparing the Sensitivity of Various Methods for Evaluating Dialogue Agents. *ArXiv* abs/2201.04723 (2022).

[55] Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2021. Multi-Task Pre-Training for Plug-and-Play Task-Oriented Dialogue System. *ArXiv* abs/2109.14739 (2021).

[56] Weiwei Sun, Shuo Zhang, Krisztian Balog, Zhaochun Ren, Pengjie Ren, Zhumin Chen, and Maarten de Rijke. 2021. Simulating User Satisfaction for the Evaluation of Task-oriented Dialogue Systems. In *SIGIR 2021*.

[57] Yueming Sun and Yi Zhang. 2018. Conversational Recommender System. In *SIGIR 2018*.

[58] Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. 2021. Transferable Dialogue Systems and User Simulators. *ArXiv* abs/2107.11904 (2021).

[59] Oriol Vinyals and Quoc Le. 2015. A Neural Conversational Model. *ArXiv* abs/1506.05869 (2015).

[60] Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. PARADISE: A Framework for Evaluating Spoken Dialogue Agents. In *ACL 1997*. 271–280.

[61] Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2021. UBAR: Towards Fully End-to-End Task-Oriented Dialog Systems with GPT-2. In *AAAI 2021*.

[62] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. POMDP-Based Statistical Spoken Dialog Systems: A Review. In *Proceedings of the IEEE*. 1160–1179.

[63] Shuo Zhang and Krisztian Balog. 2020. Evaluating Conversational Recommender Systems via User Simulation. In *SIGKDD 2020*.

[64] Shuo Zhang, Mu-Chun Wang, and Krisztian Balog. 2022. Analyzing and Simulating User Utterance Reformulation in Conversational Recommender Systems. In *SIGIR 2022*.

[65] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Bruce Croft. 2018. Towards Conversational Search and Recommendation: System Ask, User Respond. *CIKM 2018* (2018).

[66] Yinan Zhang, Xueqing Liu, and ChengXiang Zhai. 2017. Information Retrieval Evaluation as Search Simulation: A General Formal Framework for IR Evaluation. In *SIGIR 2017*. 193–200.

[67] Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Task-Oriented Dialog Systems that Consider Multiple Appropriate Responses under the Same Context. In *AAAI*.

[68] Zheng Zhang, Ryuichi Takanobu, Minlie Huang, and Xiaoyan Zhu. 2020. Recent Advances and Challenges in Task-oriented Dialog System. *ArXiv* abs/2003.07490 (2020).

[69] Zhuosheng Zhang, Hanqing Zhang, Keming Chen, Yuhang Guo, Jingyun Hua, Yulong Wang, and Ming Zhou. 2021. Mengzi: Towards Lightweight yet Ingenious Pre-trained Models for Chinese. *ArXiv* abs/2110.06696 (2021).

[70] Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. 2020. ConvLab-2: An Open-Source Toolkit for Building, Evaluating, and Diagnosing Dialogue Systems. In *ACL*.

[71] Lixin Zou, Long Xia, Yulong Gu, Xiangyu Zhao, Weidong Liu, Xiangji Huang, and Dawei Yin. 2020. Neural Interactive Collaborative Filtering. In *SIGIR 2020*. 749—-758.