Rethinking Task-Oriented Dialogue Systems: From Complex Modularity to Zero-Shot Autonomous Agent

Anonymous ACL submission

Abstract

Task-oriented dialogue (TOD) systems are predominantly designed to be composed of several functional modules (e.g. dialogue state tracker, dialogue policy, natural language generation) whether they are pipeline or end-to-end architectures. However, this modular design not only heavily relies on massive fully-annotated data, but also suffers from many intrinsic drawbacks, such as serious error accumulation, poor generalization ability, high customization cost, and low fault tolerance rate. In this paper, we rethink the architecture of the task-oriented dialogue systems and propose a novel fully zero-shot autonomous TOD agent, named AutoTOD, where all the delicate modules in traditional TOD systems are deprecated and all it needs is a general-purpose instruction-following language model (e.g. GPT-4). AutoTOD only leverages a simple instruction schema consisting of the description of tasks and external APIs, and can autonomously decide to what to do at each dialogue turn, including asking for information, calling APIs, summarizing API results, and correcting previous mistakes. Moreover, we propose a simulation-based evaluation framework to better validate the abilities of TOD models in real-life scenarios. Extensive experiments conducted on the MultiWOZ and SGD datasets show the superior task completion ability and flexible language skills of AutoTOD.[1]

1 Introduction

Task-oriented dialogue (TOD) systems have gained increasing attention in recent years both in the research community and the industry (Valizadeh and Parde, 2022; Hudeček and Dušek, 2023). They are designed to help users complete specific tasks through natural language interactions, such as querying flight tickets and booking restaurant tables (Budzianowski et al., 2018; Rastogi et al., 2020). Traditional TOD systems are mostly designed as a pipeline of several separate modules, including natural language understanding, dialogue state tracker, dialogue policy, and natural language generation (Zhang et al., 2020). These modules are trained separately and work one by one to generate the dialogue response to the user (Su et al., 2022). Later, end-to-end TOD systems emerged where the separate modules are combined and built on a single pretrained language model (He et al., 2022a; Yang et al., 2021). Thus the whole system can be trained end-to-end with annotated task dialogues. Examples of these two kinds of TOD systems are shown in Figure 1 (a, b). Nevertheless, both the pipeline and end-to-end models are essentially in the same modular architecture.

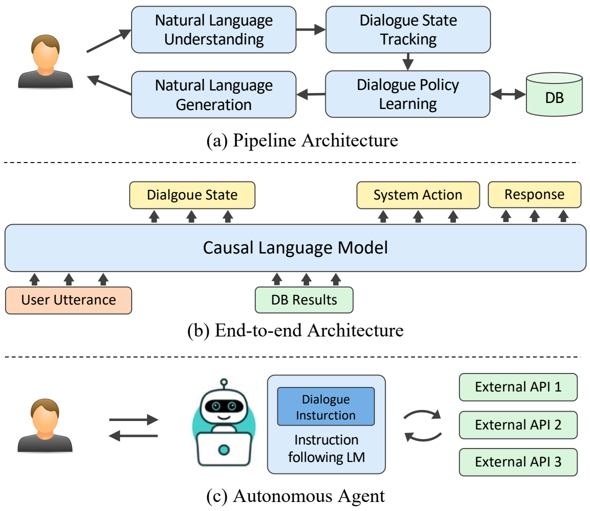

Figure 1: Different architectures of TOD systems. The pipeline architecture has several individually designed and trained modules. The end-to-end architecture combines all the modules into a causal language model. The autonomous agent uses an instruction-following language model to autonomously call external APIs and communicate with the user without any training.

[1]The code is provided as the supplementary material for the reviewers, and will be released publicly after the paper is accepted.

The classic modular TOD framework has many intrinsic drawbacks in it. Firstly, the modules are connected sequentially. Any mistake in a module will propagate to the subsequent ones. Although there are later works to merge some adjacent modules, the error propagation problem has not been fundamentally solved (Zhang et al., 2020). Secondly, the training of these systems requires a large number of fully-annotated task dialogues. At the same time, the capabilities of such TOD systems are also severely limited to the training data, which makes it difficult for them to extend to new dialogue scenarios (Mi et al., 2022). Thirdly, when encountering dialogue tasks in new forms, the module architecture usually needs to be redesigned, which makes building and maintaining a long-running TOD system a challenging and costly endeavour (Su et al., 2022). Finally, the fault tolerance ability of the systems is quite poor. When facing inappropriate input utterances or misunderstandings in previous turns, the systems struggle to lead the dialogue back to the correct path (Kim et al., 2022).

To tackle the above problems, in this paper, we rethink the architecture of the task-based dialogue systems and propose a novel fully zero-shot autonomous TOD agent, named AutoTOD. AutoTOD gives up all the delicate functional modules in the traditional pipeline and end-to-end TOD models, and the only thing it needs is a general-propose instruction-following language model, e.g. GPT-4 (OpenAI, 2023a) and Llama 2 (Touvron et al., 2023) (shown in Figure 1 (c)). We propose an instruction schema to tell the base language model what tasks it will deal with and how it should do. The instruction schema is simple enough to easily apply to various dialogue tasks, thus the generalization ability is greatly improved and the customization cost is greatly reduced. AutoTOD can intelligently and autonomously decide to call an external API with proper parameters and summarize the API results into the final responses, which greatly flexes the rigid process of state tracking and database querying in traditional TOD models. To the best of our knowledge, AutoTOD is the first TOD system that completely deprecates the traditional fragile modular design and has the real zero-shot capability.

To better evaluate the performance of AutoTOD, we also propose a simulation-based evaluation framework to validate the abilities of TOD models in real-life scenarios. We use an instruction-following language model to act as the user simulator, which has the task goals in its prompts and tries to achieve the goals via talking with the dialogue system. The dialogue system has no access to the user goals and can only do its best to complete the user's requests. Then, we propose a novel TOD evaluation approach to validate the system's ability to complete user goals where another instruction-following language model is used to extract the key information that the system provides to the user from the dialogues. By comparing with the goals for the user simulator, it's easy to conclude whether the TOD system completes the user goals successfully. We conduct extensive experiments on the MultiWOZ (Budzianowski et al., 2018) and SGD (Rastogi et al., 2020) datasets and evaluate the system performance by both our proposed simulation-based framework and the traditional TOD evaluation approach (Mehri et al., 2019). The results show the superior task completion ability and fluent language skills of AutoTOD. Furthermore, AutoTOD demonstrates great robustness when facing various dialogue scenarios.

2 Related Work

2.1 Task-Oriented Dialogue Systems

Task-oriented dialogue (TOD) systems have been studied for decades. Traditional approaches are fundamentally built in a pipeline architecture, consisting of components including natural language understanding, dialogue state tracking, dialogue policy learning, and natural language generation (Wu et al., 2019; Peng et al., 2018). Later, end-to-end TOD systems emerged, where all the modules in the pipeline modules are combined into a single model and trained end-to-end with fully annotated dialogue data (Wen et al., 2017; Wang et al., 2020). Despite the apparent simplification, the end-to-end architecture still necessitates large fully-annotated dialogue datasets for training and retains the modular nature of the traditional TOD systems.

Recently, in order to reduce the training data requirement and the cost of transferring to new dialogue scenarios, several zero-shot TOD systems have been proposed. AnyTOD (Zhao et al., 2023) adopts a neuro-symbolic approach to facilitate generalization onto unseen dialogue tasks without further training. ZS-ToD (Mosharrof et al., 2023), a zero-shot end-to-end TOD model, leverages domain schemas for robust generalization to unseen domains. However, neither AnyTOD nor ZS-ToD jumps out of the classical modular design framework and retains the traditional pipeline or end-to-end architecture. Moreover, neither of them is strictly zero-shot as they both require training data on some known tasks, and their performance is largely influenced by the quality of the collected data.

2.2 Instruction-Following Language Models and Autonomous Agents

Recently, the advent of instruction-following language models, such as ChatGPT (OpenAI, 2022) and Llama 2 (Touvron et al., 2023), has opened up a new avenue in the realm of intelligent assistants. These models show impressive capabilities in understanding user intents, generating human-like responses, and providing insights on a vast array of subjects (OpenAI, 2023a). However, their application has traditionally been limited to single-step tasks, and they lack the ability to autonomously manage multi-step processes (Nakano et al., 2022). This limitation has opened the door for the emergence of autonomous AI agents, a new class of AI applications that breaks down complex tasks into manageable subtasks and manages the execution of these subtasks in a coordinated and autonomous manner. Examples of such agents include ReAct (Yao et al., 2023), AgentGPT (ReworkdAI, 2023), AutoGPT (Richards, 2023), BabyAGI (Nakajima, 2023), and Microsoft's Jarvis (HuggingGPT) (Shen et al., 2023), each of which combines the power of LLMs with the ability to store and retrieve information, access external resources, and manage multi-step tasks. These AI agents offer a more autonomous execution of complex tasks and open a new way to re-design the task-oriented dialogue systems.

3 The AutoTOD Agent

AutoTOD is a fully zero-shot autonomous task-oriented dialogue (TOD) agent where all the delicate modules in traditional TOD systems are deprecated and all it needs is a general-purpose instruction-following language model (e.g. GPT-4). The ability of AutoTOD comes from two aspects: the language knowledge from the base language model, and the dialogue ability from the instruction schema we proposed for the dialogue tasks. The instruction schema is composed of three parts: scenario description, task information, and output format (a brief example is shown in Figure 2). The detailed introduction of each part is given below.

3.1 Scenario Description

The scenario description is at the beginning of the instruction schema and it's a brief and comprehensive description of the dialogue tasks as well as the characters of the agent. It describes the common features of the tasks and the principles that the agent needs to obey. The example scenario description for the tasks in the MultiWOZ (Budzianowski et al., 2018) dataset is shown in Figure 2 (top), where the agent is asked to act as a travel guide in Cambridge and help users to complete several querying and booking tasks.

3.2 Task Information

The task information is the main part of the instruction schema. It provides detailed information on each task separately. The task information for each task consists of three components: task description, task APIs, and task logic. An example of the restaurant task in the MultiWOZ (Budzianowski et al., 2018) dataset is shown in Figure 2 (left), and the detailed description is as follows.

The task description is a sentence briefly introducing what the task is and how the agent can help the user. The task APIs part lists all the external APIs that can be invoked in the dialogue. The agent can use them to obtain external information or interact with the external world. The information of each API first begins with the API name and a brief description, and then follows the input format definition. As shown in Figure 2, the API input format is defined as a JSON string. At last, there is a text specifying which parameters are required. It should be noticed that the language model is a pure text-in and text-out model, thus the output of APIs must be in text format. Moreover, it's a good practice to make the APIs output readable messages when receiving invalid inputs, so that the agent can adjust the inputs according to the error message, which improves the model robustness greatly.

The task logic is an optional part that gives further action guidance to the agent. The requirements for the agent can be itemized in this part. Moreover, the task logic part can also be treated as a customization area which makes it possible for the designer to adjust the agent behaviors according to the performance in real production scenarios.

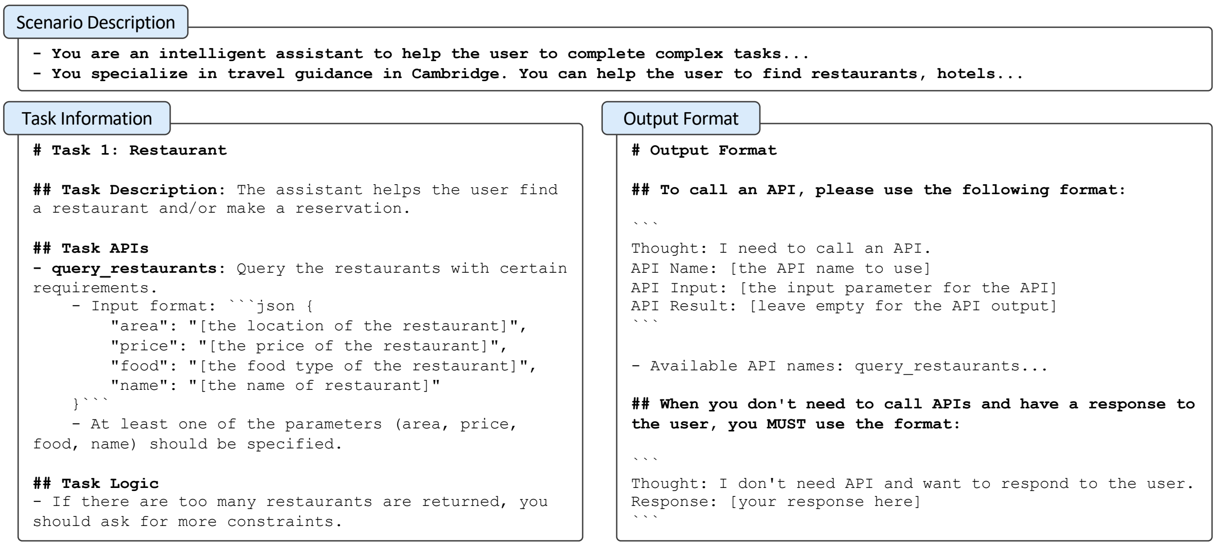

Figure 2: A demonstration of the instruction schema for the MultiWOZ dataset. The instruction schema is composed of three parts: scenario description, task information, and output format, where the task information describes the tasks for the agent. The task information of the restaurant task with one API is shown in the figure. The full instruction schema is presented in the Appendix.

3.3 Output Format

The output format specifies the output pattern of the agent. We choose the Reasoning and Acting pattern in the ReAct (Yao et al., 2023) paper where the language model first thinks about what it should do (reasoning) and then takes the corresponding action (acting). The output format for AutoTOD is shown in Figure 2 (right). We define two kinds of thoughts: calling an API and generating the response. In a dialogue turn, AutoTOD thinks about whether the information it has is enough to respond to the user. If not, it will choose the required API and generate the input parameters. The returned content of the API is appended after "`API Result:`" in the instruction. The process of calling an API may happen many times until AutoTOD thinks it should reply to the user.

4 The Simulation-based Evaluation Framework

The AutoTOD agent communicates with the user in an end-to-end manner where no dialogue state or system action is generated as in the traditional TOD systems. Besides, AutoTOD directly outputs the natural language responses rather than the delexicalized utterances without real slot values. However, the classical TOD evaluation approach (Budzianowski et al., 2018) used by almost all previous TOD systems is deeply coupled with the traditional modular TOD system architecture, and can not be used for AutoTOD directly. Therefore, we propose a new simulation-based evaluation framework that has no assumption about the architecture of the TOD system. The framework we proposed consists of a user simulator and a dialogue evaluator. The detailed introduction is presented below.

4.1 User Simulator

Thanks to the powerful understanding and generating ability of current large language models, the user simulator is implemented only by an instruction-following language model. The user simulator is driven by its instruction prompt, which is the most important part in it is the user goals. The simulator tries to complete all the goals in its mind via talking with the TOD system. We also present the grounding dialogue corresponding to the user goals in the dataset, which acts as the guideline for the simulator so it can imitate the language style and produce more reliable dialogues. The example of the prompt of the user simulator is presented in the Appendix.

4.2 Dialogue Evaluator

How to accurately evaluate the performance of TOD systems has always been a challenging problem. The core of the dialogue evaluator we proposed is to answer the question: *Does the TOD system help the user complete the goals?* As the user goals are usually composed of multiple sub-goals,

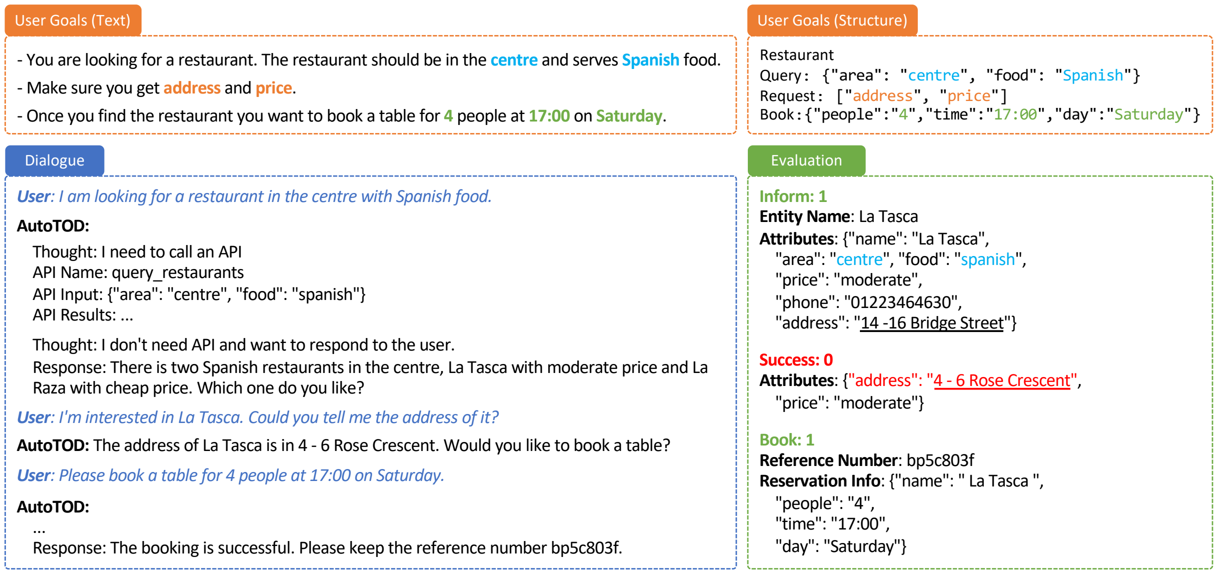

Figure 3: An example of user goals, dialogue, and evaluation results. The text user goals are used for the prompt of the user simulator. The dialogue is generated by the user simulator and TOD system (AutoTOD). The dialogue evaluator produces the evaluation results by extracting the key information from the dialogue and comparing it with the structure user goals.

the evaluation results are also composed of multiple metrics. Similar to the traditional evaluation, we define two kinds of metrics: **Inform** and **Success**, where Inform measures whether the system finds the right entity for the user, and Success measures whether the system provides all the required entity attributes. For the MultiWOZ (Budzianowski et al., 2018) dataset, we also define the particular metric **Book**, which measures whether the system makes the reservation successfully for the user. Unlike the traditional evaluation, all the metrics we defined are computed directly by the generated dialogues without any intermediate state. An evaluation example is shown in Figure 3.

4.2.1 Inform

The Inform metric is to check whether the system finds the right entity for the user. As shown in Figure 3, the system uses the API to query the database and recommends the found entity names to the user. To this end, we use a general pretrained language model to extract the entity name (the primary key in the database) from the dialogue, and then obtain the complete entity attributes from the database. Thus, the Inform metric can be computed by comparing the constraints in the user goals and the attributes of the recommended entity. The pretrained language model is powerful enough and it's only used to extract the minimal information (primary key) from the dialogue, thus the probability of failed extraction can be quite low.

4.2.2 Success

The Success metric is to check whether the system provides all the required entity attributes for the user. We also use a pretrained language model to extract the provided entity attributes from the dialogue. The Success metric passes if and only if 1) the found entity is right (Inform is passed), 2) the system provides all the attributes the user wants, and 3) all the attribute values are correct. As shown in Figure 3, the value of the address attribute is not matched with that in the database, so the Success metric is not passed in that example.

4.2.3 Book

In the MultiWOZ dataset, the user often makes a reservation after finding the wanted venue (e.g. restaurant, hotel). However, whether the booking is successful is completely ignored in the traditional TOD evaluation. Therefore, we define the Book metric to count the success rate of booking. In MultiWOZ, the system will return a reference number after making a reservation, thus we use a pretrained language model to extract the reference number in the dialogue. Then the complete reservation information can be obtained from the reservation database. The Book metric is computed by comparing the booking constraints in the user goals and the database record.

At last, we define a comprehensive metric **Com-**

Model	Domain Level				Dialogue Level			
	Inform	Success	Book	Combine	Inform	Success	Book	Combine
SimpleTOD* (Hosseini-Asl et al., 2020)	32.5	29.4	-	23.6	18.8	22.0	-	14.9
UBAR* (Yang et al., 2021)	40.8	33.3	-	28.7	24.0	26.8	-	18.7
GALAXY* (He et al., 2022b)	44.4	35.1	-	31.0	26.4	28.8	-	20.4
Mars* (Sun et al., 2023)	42.7	34.4	-	30.0	25.9	27.5	-	19.8
Mars (5% few-shot) (Sun et al., 2023)	28.9	26.3	-	21.0	16.2	14.0	-	11.6
TOATOD* (Bang et al., 2023)	45.3	36.7	-	31.8	27.8	26.9	-	20.6
AutoTOD (GPT-3.5)	62.5	52.7	51.4	57.3	43.0	46.2	48.4	45.8
AutoTOD (GPT-4)	**85.2**	**59.1**	**86.7**	**79.1**	**80.2**	**46.9**	**82.0**	**72.3**
AutoTOD (Llama 2 70B)	54.3	42.6	44.2	48.9	32.7	30.5	31.9	32.0
AutoTOD (Llama 2 13B)	37.1	28.5	31.8	33.6	28.6	23.1	27.3	26.9

Table 1: Goal completion evaluation results on MultiWOZ 2.0. All the models are evaluated with our proposed simulation-based evaluation framework. Models marked with an asterisk (*) are trained with all the training data of MultiWOZ 2.0 while AutoTOD models are fully zero-shot. All the baseline models don't have the booking ability thus their Book scores are none.

bine to indicate the overall performance of the TOD system, which is formulated as: Combine $= 0.5 \cdot \text{Inform} + 0.25 \cdot (\text{Success} + \text{Book})$.

5 Experiments

5.1 Datasets

We implement AutoTOD for two widely used datasets: MultiWOZ (Budzianowski et al., 2018) and SGD (Rastogi et al., 2020). MultiWOZ is a large-scale multi-domain TOD dataset with multiple revised versions. As only the task ontology and user goals are used by AutoTOD, we just take the MultiWOZ 2.0 version for wide baseline models. SGD is a schema-guided TOD dataset spinning over 26 services and each service is accompanied by a schema that describes the APIs and slots in it. The schemas are used in the instruction of AutoTOD and we collect the user actions in each dialogue to form the user goals for the user simulator. We implement the API backend with DB support for both two datasets in order to conduct real TOD dialogues.

5.2 Experimental Settings

We build AutoTOD with several popular and powerful instruction-following language models, including the closed-source models GPT-3.5 (`gpt-3.5-turbo-0613`) (OpenAI, 2023b), GPT-4 (`gpt-4-0613`) (OpenAI, 2023a) and open-source models Llama 2 (`llama-2-chat 13B`, `70B`) (Touvron et al., 2023). The model used for the user simulator and dialogue evaluator is GPT-3.5. We use OpenAI API[2] for using the OpenAI series models and Replicate API[3] for using Llama 2 series models.

5.3 Results on MultiWOZ

Goal Completion The evaluation results of goal completion ability on MultiWOZ 2.0 are shown in Table 1. The scores are calculated from the domain level and dialogue level. A dialogue is counted as completed only if all the domains in it are completed. We can see that the AutoTOD models outperform the baselines with a large margin even the full-shot trained models. It also indicates the previous TOD models cannot work well in real dialogue scenarios and our user-simulator based evaluation is necessary. For AutoTOD with different base models, the OpenAI series models perform much better than the Llama series models. There is still a gap between the closed-source and open-source large language models. AutoTOD with GPT-4 performs far better than all the other models, while the 13B llama model performs worst.

Language Diversity It's not sufficient to only evaluate the goal completion ability for TOD systems. Here we use some language diversity metrics to evaluate the quality of model responses (Nekvinda and Dušek, 2021). The metrics include some statistics about words, n-gram and information entropy (number of n-grams, Shannon Entropy (SE), Conditional bigram Entropy (CE), Mean Segmental Type-Token Ratio (MSTTR), Measure of Textual Lexical Diversity (MTLD) and Hypergeometric Distribution Function (HDD)) (Terragni et al., 2023). The results are

[2]OpenAI API: `https://openai.com/blog/openai-api`

[3]Replicate API: `https://replicate.com`

Model	#Uni	#Bi	#Tri	SE	CE	MSTTR	MTLD	HDD
SimpleTOD* (Hosseini-Asl et al., 2020)	683	2057	3388	7.13	2.00	59.32	31.28	76.91
UBAR* (Yang et al., 2021)	760	2424	3658	7.20	2.05	61.21	34.33	77.21
GALAXY* (He et al., 2022b)	791	3287	4160	7.45	2.16	62.55	34.19	80.23
Mars* (Sun et al., 2023)	849	3315	4781	7.61	2.14	64.98	40.15	83.43
Mars (5% few-shot) (Sun et al., 2023)	711	2110	3101	7.02	1.91	60.12	32.17	73.30
TOATOD* (Bang et al., 2023)	898	3829	5047	7.65	2.20	68.23	42.84	80.25
AutoTOD (GPT-3.5)	1722	6201	10188	8.11	2.62	76.61	65.09	**86.07**
AutoTOD (GPT-4)	**2031**	**7391**	**13181**	**8.63**	**2.91**	**80.85**	**80.93**	85.71
AutoTOD (Llama 2 70B)	1482	5281	7149	7.71	2.59	72.12	58.82	81.17
AutoTOD (Llama 2 13B)	1037	4121	6843	7.54	2.31	69.50	49.13	78.67

Table 2: Language diversity evaluation results on MultiWOZ 2.0. #Uni/#Bi/#Tri stands for the number of uni-grams/bi-grams/tri-grams in system responses. SE, CE, MSTTR, MTLD, and HDD stand for Shannon Entropy, Conditional bigram Entropy, Mean Segmental Type-Token Ratio, Measure of Textual Lexical Diversity, and Hypergeometric Distribution Function separately.

Model	Inform	Success	BLEU	Combine
SimpleTOD*	83.4	69.1	14.8	91.0
UBAR*	**94.9**	80.3	18.0	105.6
GALAXY*	93.5	**84.9**	**20.8**	**110.0**
Mars*	89.9	81.3	18.6	104.2
Mars (5% shot)	56.7	42.3	12.4	61.9
AutoTOD				
• GPT-3.5	87.2	82.8	9.3	94.3
• GPT-4	91.7	84.4	10.4	98.5
• Llama 2 70B	73.3	69.8	7.8	79.4

Table 3: Traditional TOD evaluation results on MultiWOZ 2.0. All the models are evaluated by feeding with grounding user utterances regardless of dialogue consistency.

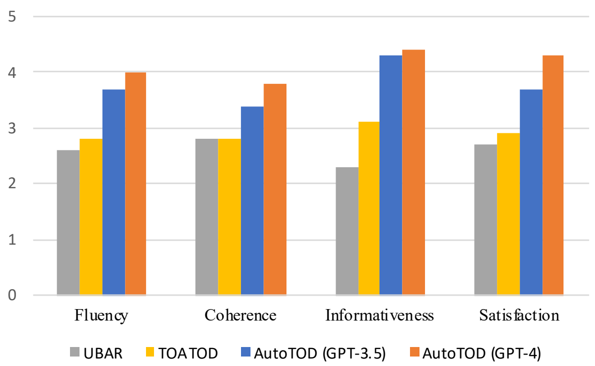

Figure 4: Human evaluation results on MultiWOZ dialogues. Human evaluators are asked to rate the randomly sampled dialogues from 4 aspects, each with a max score of 5.

shown in Table 2. We can see that all the AutoTOD models show better language diversity than the trained baselines. AutoTOD based on GPT-4 gains the highest scores on almost all metrics.

5.4 Results of Traditional TOD Evaluation

we also conduct the traditional TOD evaluation (Budzianowski et al., 2018) on the baselines and AutoTOD models. Following the previous works, the TOD models are fed with the grounding user utterances in each turn regardless of the consistency of the dialogue flow. We track the Inform, Success, BLEU, and Combine metrics in the traditional TOD evaluation. For AutoTOD, the generated API parameters are used as the dialogue state to calculate the Inform score, and the entity attributes in the responses are used to calculate the Success score. The evaluation results on the MultiWOZ datasets are shown in Table 3. We can see that AutoTOD variants have competitive Inform and Success scores with the baseline models, which indicates the powerful language understanding and task completion abilities of AutoTOD. The weakest metric for AutoTOD is BLEU, which is not surprising since AutoTOD does not see any grounding utterances in the dataset and has no prior about the language style of ground truth responses.

5.5 Human Evaluation

We conduct human evaluation to supplement the limitations of automated metrics. Two full-shot baseline models (UBAR (Yang et al., 2021), TOATOD (Bang et al., 2023)) and two AutoTOD variants (GPT-3.5, GPT-4) are taken into account. We random sample 100 dialogues from their dialogue logs separately and ask for 5 graduate students to

Model	Service Level		Dialogue Level		Diversity				
	Inform	Success	Inform	Success	SE	CE	MSTTR	MTLD	HDD
SimpleTOD* (Hosseini-Asl et al., 2020)	13.1	10.4	9.3	9.5	6.88	1.95	69.22	51.44	76.19
ZS-TOD* (Mosharrof et al., 2023)	24.9	11.2	18.3	10.1	7.11	2.04	71.16	53.23	80.01
AutoTOD (GPT-3.5)	46.8	21.0	35.0	20.0	8.44	2.45	73.87	58.91	86.87
AutoTOD (GPT-4)	**52.4**	**25.9**	**48.1**	**24.8**	**9.58**	**2.51**	**78.21**	**63.02**	**89.66**
AutoTOD (Llama 2 70B)	41.9	15.0	36.1	13.8	8.31	2.30	71.12	58.05	84.98
AutoTOD (Llama 2 13B)	35.0	11.0	32.4	9.9	7.85	2.25	70.18	56.13	82.20

Table 4: Goal completion and language diversity evaluation results on SGD. Models marked with an asterisk (*) are trained with all the training data of SGD while AutoTOD models are fully zero-shot.

User Simulator	Dialog Goal			Diversity		
	Inf.	Suc.	Book	SE	CE	MSTTR
TOD System: UBAR						
Agenda	20.4	11.4	-	6.15	1.82	57.9
TUS	18.2	12.7	-	6.31	1.97	61.8
GenTUS	19.5	13.3	-	6.88	2.13	62.7
Ours	**24.0**	**26.8**	-	**7.12**	**2.41**	**62.9**
TOD System: AutoTOD (GPT-3.5)						
Agenda	22.7	13.3	16.1	6.31	1.86	59.9
TUS	20.6	17.1	19.0	6.63	2.11	65.0
GenTUS	19.5	18.5	28.6	7.26	2.61	71.5
Ours	**43.0**	**46.2**	**48.4**	**7.36**	**2.70**	**73.5**

Table 5: Comparison of different user simulators on MultiWOZ 2.0. Inf. and Suc. represent for Inform and Success separately.

rate from 4 aspects: language fluency (Fluency), dialogue coherence (Coherence), information accuracy in responses (Informativeness), and overall satisfaction (Satisfaction). The results are shown in Figure 4. We can see that the AutoTOD models achieve comparable results with the two full-shot TOD models, where the GPT-4 variant in particular achieves the highest scores on all four metrics. Both the two AutoTOD models have significantly high scores on the Information metric. An important reason is that AutoTOD has a better ability to summarize the results returned by external APIs to the user.

5.6 Results on SGD

The evaluation results for the SGD dataset are presented in Table 4. The goal completion metrics are summarized into service level and dialogue level assessments. Notably, the booking aspect is treated as a special Inform task in SGD, thus there is no Book metric presented. The results demonstrate that all AutoTOD models surpass the baseline models in terms of both goal completion and language diversity. In particular, Among different AutoTOD variants, the OpenAI series models exhibit superior performance compared to the Llama models.

5.7 User Simulators

The user simulator plays a pivotal role in the evaluation of task-oriented dialogue (TOD) systems within real-world scenarios. An effective user simulator should not impede the capabilities of TOD systems and should possess clear goal expression, as well as flexible dialogue strategies and diverse languages. In order to assess the performance of our proposed user simulator, we conducted experiments to compare it with several SoTA user simulators, namely Agenda (Schatzmann et al., 2007), TUS (Lin et al., 2021) and GenTUS (Lin et al., 2022). These simulators were constructed with a dialogue policy incorporating either template-based or neural language generation modules, and were employed to engage in dialogues with various TOD systems using the MultiWOZ dataset. The results are presented in Table 5, illustrate that our proposed user simulator enables TOD systems to achieve higher goal completion scores, while simultaneously offering greater linguistic diversity.

6 Conclusion

In this paper, we propose AutoTOD, a novel fully zero-shot autonomous agent for task-oriented dialogues. AutoTOD deprecates all the delicate modules in traditional TOD systems and only uses an instruction-following language model to autonomously call external APIs and communicate with the user, which greatly reduces the construction cost and improves the generalization ability. We also propose a simulation-based evaluation framework to evaluate TOD systems in more real scenarios. Extensive experiments demonstrate the superior task completion ability and flexible language skills of AutoTOD.

7 Limitations

AutoTOD is the first fully zero-shot task-oriented dialogue (TOD) agent as far as we know, although it still exhibits certain limitations that warrant future improvements. (1) Due to the API cost, AutoTOD has only been implemented on a limited number of large language models (LLMs) (GPT-3.5, GPT-4, and Llama 2). However, it is crucial to conduct a more comprehensive comparison by including other well-known LLMs such as Claude (Anthropic, 2023) and PaLM (Chowdhery et al., 2022). (2) There is a need for more comprehensive evaluations of TOD. Regarding goal completion, the metrics employed should strive for enhanced efficiency and accuracy. For the language aspect, better automatic evaluation methods need to be developed to assess language quality beyond diversity. (3) It would be better to assess AutoTOD on a broader range of datasets and employ a greater variety of instruction prompts. (4) Few-shot methods for LLMs-based TOD agents need to be investigated and the agent performance would be further improved.

References

Anthropic. 2023. Introducing claude. `https://www.anthropic.com/index/introducing-claude`. Accessed 2023-08-13.

Namo Bang, Jeehyun Lee, and Myoung-Wan Koo. 2023. Task-optimized adapters for an end-to-end task-oriented dialogue system. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7355–7369, Toronto, Canada. Association for Computational Linguistics.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Wanwei He, Yinpei Dai, Min Yang, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. 2022a. Space-3: Unified dialog model pre-training for task-oriented dialog understanding and generation.

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. 2022b. Galaxy: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):10749–10757.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191. Curran Associates, Inc.

Vojtěch Hudeček and Ondřej Dušek. 2023. Are llms all you need for task-oriented dialogue?

Takyoung Kim, Yukyung Lee, Hoonsang Yoon, Pilsung Kang, Junseong Bang, and Misuk Kim. 2022. Oh my mistake!: Toward realistic dialogue state tracking including turnback utterances. In *Proceedings of the Towards Semi-Supervised and Reinforced Task-Oriented Dialog Systems (SereTOD)*, pages 1–12, Abu Dhabi, Beijing (Hybrid). Association for Computational Linguistics.

Hsien-chin Lin, Christian Geishauser, Shutong Feng, Nurul Lubis, Carel van Niekerk, Michael Heck, and Milica Gasic. 2022. GenTUS: Simulating user behaviour and language in task-oriented dialogues with generative transformers. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 270–282, Edinburgh, UK. Association for Computational Linguistics.

Hsien-chin Lin, Nurul Lubis, Songbo Hu, Carel van Niekerk, Christian Geishauser, Michael Heck, Shutong Feng, and Milica Gasic. 2021. Domain-independent user simulation with transformers for task-oriented dialogue systems. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 445–456, Singapore and Online. Association for Computational Linguistics.

Shikib Mehri, Tejas Srinivasan, and Maxine Eskenazi. 2019. Structured fusion networks for dialog. In

Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue, pages 165–177, Stockholm, Sweden. Association for Computational Linguistics.

Fei Mi, Yasheng Wang, and Yitong Li. 2022. Cins: Comprehensive instruction for few-shot learning in task-oriented dialog systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11076–11084.

Adib Mosharrof, M. H. Maqbool, and A. B. Siddique. 2023. Zero-shot generalizable end-to-end task-oriented dialog system using context summarization and domain schema.

Yohei Nakajima. 2023. Babyagi. https://github.com/yoheinakajima/babyagi.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback.

Tomáš Nekvinda and Ondřej Dušek. 2021. Shades of bleu, flavours of success: The case of multiwoz.

OpenAI. 2022. Introducing chatgpt. https://openai.com/blog/chatgpt. Accessed 2023-08-13.

OpenAI. 2023a. Gpt-4 technical report.

OpenAI. 2023b. Models - gpt-3.5. https://platform.openai.com/docs/models/gpt-3-5. Accessed 2023-08-13.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. 2018. Deep Dyna-Q: Integrating planning for task-completion dialogue policy learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2182–2192, Melbourne, Australia. Association for Computational Linguistics.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8689–8696.

ReworkdAI. 2023. Agentgpt. https://github.com/reworkd/AgentGPT.

Toran Bruce Richards. 2023. Auto-gpt. https://github.com/Significant-Gravitas/Auto-GPT.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152, Rochester, New York. Association for Computational Linguistics.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face.

Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2022. Multi-task pre-training for plug-and-play task-oriented dialogue system. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4661–4676, Dublin, Ireland. Association for Computational Linguistics.

Haipeng Sun, Junwei Bao, Youzheng Wu, and Xiaodong He. 2023. Mars: Modeling context & state representations with contrastive learning for end-to-end task-oriented dialog. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11139–11160, Toronto, Canada. Association for Computational Linguistics.

Silvia Terragni, Modestas Filipavicius, Nghia Khau, Bruna Guedes, André Manso, and Roland Mathis. 2023. In-context learning user simulators for task-oriented dialog systems.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Mina Valizadeh and Natalie Parde. 2022. The AI doctor is in: A survey of task-oriented dialogue systems for healthcare applications. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6638–6660, Dublin, Ireland. Association for Computational Linguistics.

Kai Wang, Junfeng Tian, Rui Wang, Xiaojun Quan, and Jianxing Yu. 2020. Multi-domain dialogue acts and response co-generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7125–7134, Online. Association for Computational Linguistics.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449, Valencia, Spain. Association for Computational Linguistics.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, Florence, Italy. Association for Computational Linguistics.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2021. Ubar: Towards fully end-to-end task-oriented dialog systems with gpt-2.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Zheng Zhang, Ryuichi Takanobu, Qi Zhu, Minlie Huang, and Xiaoyan Zhu. 2020. Recent advances and challenges in task-oriented dialog system.

Jeffrey Zhao, Yuan Cao, Raghav Gupta, Harrison Lee, Abhinav Rastogi, Mingqiu Wang, Hagen Soltau, Izhak Shafran, and Yonghui Wu. 2023. Anytod: A programmable task-oriented dialog system.

A Appendix

A.1 Case Study

We present a dialogue fragment between AutoTOD (GPT-3.5) and the user simulator in Table 8. The user simulator is fed with the user goals, located at the top of the table, and asked to achieve the goals step by step via talking with AutoTOD. AutoTOD has no access to the goals and can only leverage the APIs given to it to complete the user's requests.

A.2 The Instruction Schema for AutoTOD

The full version of the instruction schema for AutoTOD, specifically designed for the restaurant domain within the MultiWOZ dataset, is provided in Listing 1. The instruction schema comprises three distinct components: scenario description, task information, and output format. During execution, the task information component encompasses all five domains (restaurant, hotel, attraction, train, taxi) available in the MultiWOZ dataset.

Domains	Inform	Success	Book	Overall
Restaurant	98.8	95.2	97.6	97.7
Hotel	98.3	97.5	100.0	98.5
Attraction	100.0	96.2	-	98.4
Train	-	96.8	100.0	98.6
Taxi	95.8	100.0	-	97.3
Overall	97.6	97.7	99.0	97.3

Table 6: The accuracy of the information extractor within the dialogue evaluator. The results are obtained through manual verification 100 randomly selected dialogues generated by AutoTOD.

A.3 The Prompt for the User Simulator

An example of the user simulator prompt for MultiWOZ tasks is presented in Listing 2. The pivotal component of the prompt is the user goals, which serve as the driving force behind the interactions between the user simulator and the TOD agent. Additionally, the prompt includes the corresponding grounding dialogue extracted from the dataset. To ensure the appropriate behavior of the simulator, specific instructions are provided, such as the requirement for the simulator to conclude the dialogue by outputting "Dialogue Ends" when all the goals have been successfully completed.

A.4 The Information Extractor in Dialogue Evaluator

In the simulation-based evaluation framework, the dialogue evaluator employs a pretrained large language model, (GPT-3.5 in our experiments) to extract essential information from the generated dialogues. To illustrate this process, we provide an example prompt and its corresponding output in Listing 3. The prompt encompasses multiple components, including user goals, the generated dialogue, specific questions regarding the desired information, and the specified answer format in JSON. To optimize the API cost, all the necessary information in one domain is extracted at once within a single prompt.

Given the powerful capabilities of pretrained large language models, the extraction accuracy can be quite high. To quantify this accuracy, we conducted a manual verification process on a randomly selected subset of 100 dialogues from the MultiWOZ test set. The extraction accuracy results are presented in Table 6. The overall accuracy is determined to be over 97%. Notably, the Inform

Schema	Service Level		Dialogue Level	
	Inform	**Success**	**Inform**	**Success**
SGD (v0)	46.0	20.9	34.6	20.0
Variant v1	46.5	22.5	34.5	20.4
Variant v2	44.9	20.9	33.9	18.8
Variant v3	47.7	21.0	36.1	19.1
Variant v4	46.1	20.5	35.5	19.0
Variant v5	45.6	22.5	33.7	20.7

Table 7: Evaluation results of AutoTOD (GPT-3.5) on the service schemas of SGD and the 5 variants in SGD-X. The same 100 dialogues are sampled from the test set and equipped with different service schemas separately.

information in the taxi domain presents the most challenging extraction task, achieving an accuracy of 95.8%. The overall accuracy achieves 97.3%, indicating the effectiveness of the language model-based information extractor.

A.5 Influence of Task Information

To assess the robustness of AutoTOD concerning various task information, we conducted experiments utilizing the SGD-X dataset, which is an extension of SGD comprising 5 variants of service schemas. These variants exhibit semantic similarity while displaying diverse stylistic characteristics. Consequently, we transformed these diverse schemas into task information for AutoTOD. Using the GPT-3.5 version of AutoTOD, we independently ran experiments on 100 randomly sampled dialogues from MultiWOZ. The results, presented in Table 7, indicate that the performance difference between the original SGD schema and its five variants is not prominent. This outcome demonstrates the robustness of AutoTOD, as it is not easily influenced by different task descriptions. To further quantify the significance of the differences in scores, we performed a statistical significance test, which confirmed that the observed variations were not statistically significant ($p < 0.05$).

<table>
<tr><td>Goals</td><td>• You are looking for information in Cambridge.
• You are looking for places to go in town. The attraction should be in the centre and should be in the type of entertainment.
• If there is no such attraction, how about one that is in the type of cinema.
• Make sure you get address, postcode, and phone number.
• You are also looking for a train. The train should go to Peterborough and should leave on Sunday.
• The train should leave after 13:45 and should depart from cambridge.
• Make sure you get train ID, travel time, and price.</td></tr>
<tr><td>User</td><td>I'm looking for entertainment places in the centre of town.</td></tr>
<tr><td>AutoTOD</td><td>API Name: query_attractions
API Input: {"area": "centre"}
API Result: | id | name | type | area | address | postcode | phone | pricerange | entrance_fee | openhours |
| 2 | adc theatre | theatre | centre | park street | cb58as | 01223300085 | ? | ? | ? |
43 more records ...<hr>Response: I found a theatre in the centre of town with address: Park Street, CB5 8AS. Please let me know if you need more information or if there's anything else I can help with!</td></tr>
<tr><td>User</td><td>How about a cinema?</td></tr>
<tr><td>AutoTOD</td><td>API Name: query_attractions
API Input: {"type": "cinema", "area": "centre"}
API Result: | id | name | type | area | address | postcode | phone | pricerange | entrance_fee | openhours |
| 75 | vue cinema | cinema | centre | the grafton centre, east road | cb11ps | 08712240240 | ? | ? | ? |<hr>Response: There is a cinema called Vue Cinema located in the centre of town. You can contact them at 08712240240 for more information about pricing and opening hours.</td></tr>
<tr><td>User</td><td>Yes please, I would like the address, postcode, and phone number.</td></tr>
<tr><td>AutoTOD</td><td>Response: The address of Vue Cinema is The Grafton Centre, East Road, and the postcode is CB1 1PS. The phone number is 0871 224 0240.</td></tr>
<tr><td>User</td><td>I am also looking for a train that will go to Peterborough and leave on a Sunday. Can you locate one for me?</td></tr>
<tr><td>AutoTOD</td><td>API Name: query_trains
API Input: {"destination": "peterborough", "day": "sunday"}
API Result: | id | arriveBy | day | departure | destination | duration | leaveAt | price | trainID |
| 1041 | 05:56 | sunday | cambridge | peterborough | 50 minutes | 05:06 | 13.20 pounds | TR8293 |
| 1042 | 06:24 | sunday | cambridge | peterborough | 50 minutes | 05:34 | 13.20 pounds | TR2913 |
35 more records ...<hr>Response: Sure! Here are some trains that go to Peterborough and leaves on a Sunday. They leave Cambridge at 05:06, 05:34, 06:06. Please let me know if you would like more information about any of these trains.</td></tr>
<tr><td>User</td><td>Is there a train that leaves after 13:45?</td></tr>
<tr><td>AutoTOD</td><td>API Name: query_trains
API Input: {"destination": "peterborough", "day": "sunday", "leaveAt": "13:45"}
API Result: | id | arriveBy | day | departure | destination | duration | leaveAt | price | trainID |
| 1059 | 14:56 | sunday | cambridge | peterborough | 50 minutes | 14:06 | 13.20 pounds | TR0283 |
| 1060 | 15:24 | sunday | cambridge | peterborough | 50 minutes | 14:34 | 13.20 pounds | TR3577 |
17 more records ...<hr>Response: Yes, there are trains that leave after 13:45 to go to Peterborough on a Sunday. Here are a few options: train TR0283 leaves at 14:06, train TR3577 leaves at 14:34, train TR3434 leaves at 15:06.</td></tr>
</table>

Table 8: A dialogue fragment between AutoTOD (GPT-3.5) and the user simulator. The user simulator is provided with the predefined goals, specified at the top of the table, while AutoTOD utilizes external APIs to fulfill the user requests.

Listing 1: The full version of the instruction schema for AutoTOD (restaurant domain in MultiWOZ)

````
You are an intelligent AI assistant to help the user complete complex tasks. The task may contain
↪  several sub-tasks, and the AI Assistant first determines which sub-tasks are involved in the
↪  user's utterance, and then completes the user's request according to the instructions of the
↪  corresponding sub-tasks.

You specialize in travel guidance in Cambridge. You can help the user to find restaurants, hotels,
↪  attraction, trains and taxi as well as make reservations.

# Task #1: Restaurant

## Task Description
The AI Assistant helps the user find a restaurant and/or make a reservation.

## Task APIs

- query_restaurants: Query the restaurants with certain requirements.
    - Parameter: the input parameter should be a json string satisfying the following format:
    ```json {
        "area": "[the location of the restaurant. only allowed values: centre, north, south, east,
        ↪  west]",
        "price": "[the price range of the restaurant. only allowed values: cheap, moderate,
        ↪  expensive]",
        "food": "[the food type or cuisine of the restaurant]",
        "name": "[the name of restaurant]"
    }```
    - At least one of the parameters (area, price, food, name) should be specified.

- book_restaurant: Book a restaurant with certain requirements
    - Parameter: the input parameter should be a json string satisfying the following format:
    ```json {
        "name": "[the name of restaurant to book]",
        "people": "[the number of people of the booking]",
        "day": "[the day when the people go in a week. only allowed values: monday, tuesday,
        ↪  wednesday, thursday, friday, saturday, sunday]",
        "time": "[the time of the reservation. time format: hh:mm, examples: 08:30, 16:00]"
    }```
    - All the parameters (name, people, day, time) are required.

## Task Logic

- After using the query_restaurants API to query restaurants with user's constraints, the AI
↪  Assistant should recommend the restaurant names to the user for choosing.
- If there are too many restaurants returned by query_restaurants, the AI Assistant should ask the
↪  user for more constraints rather than asking for reservaton.

# Output Format

## To call an API, please output with the following format:
```
Thought: I need to call an API.
API Name: [the API name to use]
API Input: [the input parameter for the API]
API Result: [leave empty for the API output]
```
- Available tool names:
    - Restaurnt: query_restaurants, book_restaurant
    - Hotel: query_hotels, book_hotel
    - Attraction: query_attractions
    - Train: query_trains, buy_train_tickets
    - Taxi: book_taxi

## When you don't need to call APIs and have a response to the user, you MUST use the format:
```
Thought: I don't need API and want to respond to the user.
Response: [your response here]
````
````

Listing 2: The example prompt for the user simulator.

```
You act as a user to talk to an AI assistant to complete some tasks.

You should carefully read and understand the User Goals below, then talk with the AI Assistant and
↪  gradually express the intents in the goals. Your purpose is to let the user achieve the goals as
↪  much as possible.

Note that the AI Assistant is not perfect. It may make various mistakes, including ignoring the user's
↪  requests, executing the wrong instructions, forgetting early conversation content, etc. The user
↪  you play should talk to the AI Assistant as patiently as possible, remind him to correct when you
↪  find that the AI assistant made a mistake, and complete the task as much as possible.

When the dialogue goals are completed or are not been completed, the user will output "Dialogue Ends"
↪  to indicate the end of the dialogue. The user doesn't need to try conditions other than the
↪  dialogue goals.

There is also a reference dialogue example to achieve the goals. The simulator user may learn from the
↪  language style and dialogue strategy. The final simulated dialogue style should be similar to the
↪  reference dialogue style.


User Goals:

You are looking for a restaurant. The restaurant should be in the west and should serve kosher food.
The restaurant should be in the expensive price range.
If there is no such restaurant, how about one that serves indian food.
Make sure you get address, phone number, and postcode.

Reference dialogue:

User: I'm looking for a restaurant that serves kosher food in the west side.
AI Assistant: I'm sorry, we don't have any Kosher restaurants on the west side. Would you like to try
↪  a different cuisine type?
User: I was really hoping for Kosher.
AI Assistant: Unfortunately, there are no Kosher restaurants anywhere in the city. Can I help you find
↪  something else?
User: How about an expensive Indian restaurant? I know there are many of those!
AI Assistant: We have 5 in the west side. Cocum is very popular, would you like to try there?
User: Sure. What's Cocum's address, phone number, and postcode?
AI Assistant: Here is the information you've requested for Cocum: 71 Castle Street City Centre,
↪  01223366668, cb30ah
User: Great, thank you! That is all I need today.
AI Assistant: You're quite welcome. Thank you and goodbye!

Current conversation:
User: I'm looking for a restaurant that serves kosher food in the west side.
AI Assistant: I'm sorry, we don't have any Kosher restaurants on the west side. Would you like to try
↪  a different cuisine type?
User:
```

Listing 3: The example prompt and its output of the information extractor in dialogue evaluator.

```
You are a calm, objective and professional judger and good at to evaluate quality of dialuges between
↪  user and AI Assistant. Your judging results are always accurate and concise.

There is a dialogue between a user and an AI Assistant. The user has the goals in his minds (User
↪  Goals) and talks with the AI Assistant to achieve the goals. The AI Assistant is a intelligent
↪  agent that is able to understand the user utterances, decide to take actions to use external
↪  tools, and generate proper responses. Your task is to judge whether the AI Assistant helps the
↪  user achieve his goals successfully by answering the questions one by one.

User Goals:

You want to find an expensive restaurant and it should be in the south part of town.
You want to know the address and phone number of the venue.

Dialogue:

User: I would like an expensive restaurant in the south part of town
AI Assistant: I found some expensive restaurants in the south part of town. Here are a few options:
1. Taj Tandoori
2. Good Luck Chinese Food Takeaway
3. Chiquito Restaurant Bar
4. Frankie and Bennys
5. Peking
Let me know if you would like more information about any of these restaurants or if you have any other
↪  preferences.
User: I don't care.
AI Assistant: Here are some expensive restaurants in the south part of town:
1. Taj Tandoori
2. Good Luck Chinese Food Takeaway
3. Chiquito Restaurant Bar
4. Frankie and Bennys
5. Peking
Let me know if you would like more information about any of these restaurants.
User: What is the address and phone number of Chiquito Restaurant Bar?
AI Assistant: The address of Chiquito Restaurant Bar is 2G Cambridge Leisure Park Cherry Hinton Road
↪  Cherry Hinton. The phone number is 01223400170.
User: Thank you. Good bye.
AI Assistant: You're welcome! If you have any more questions in the future, feel free to ask. Goodbye
↪  and have a great day!

Questions:
1. What restaurant is the user interested in and asking information about it?
2. What is the address of the hotel?
3. What is the phone number of the hotel?

Answer Format:
Please output the answer in json format like this:
```{
    "restaurant": "<fill the answer of question 1>"
    "address": "<fill the answer of question 2>"
    "phone": "<fill the answer of question 3>"
}```
If no answer for a question, please fill `none`.

Answer:
```{
    "restaurant": "Chiquito Restaurant Bar"
    "address": "2G Cambridge Leisure Park Cherry Hinton Road Cherry Hinton"
    "phone": "01223400170"
}```
```